UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

Uma ferramenta para a aprendizagem de desenvolvimento de sistemas

AUTOR: Rodrigo Gonçalves

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

ORIENTADOR: Prof. Antônio Carlos Mariani

Florianópolis, SC

2004/2

Rodrigo Gonçalves

Uma ferramenta para a aprendizagem de desenvolvimento de sistemas

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Antônio Carlos Mariani

Banca Examinadora

Prof. Frank Siqueira

Prof. Ronaldo dos Santos Mello

À minha família, sempre presente.

Agradecimentos

Agradeço em primeiro lugar à meus pais, Valdir José Gonçalves e Ivone Vieira Gonçalves. Não teria feito nem metade do que fiz até hoje se não fosse por seus esforços e atenção dada.

Agradeço ao meu orientador, Antônio Carlos Mariani, pela idéia original do projeto e pelas importantes orientações e dúvidas resolvidas durante a confecção deste.

Agradeço aos membros da banca examinadora, Frank Siqueira e Ronaldo dos Santos Mello, pelos valorosos comentários sobre o trabalho, que enriqueceram o mesmo.

Não posso deixar de agradecer a minha irmã, Renata Gonçalves, que serviu de “cobaia” ao final do trabalho durante a preparação da apresentação do mesmo.

Por último, mas não menos importante, agradeço aos amigos, que ou através de sugestões ou apenas me “aturando”, contribuíram para a realização deste trabalho.

Resumo

Neste trabalho apresenta-se uma ferramenta para auxiliar o processo de aprendizagem sobre desenvolvimento de sistemas computacionais. Esta permite a modelagem visual de sistemas computacionais, através de diagramas de classes segundo a notação UML. A partir deste, gera-se de forma automatizada o código correspondente (em ambiente *Smalltalk*), já incluindo o suporte a persistência dos dados envolvidos sobre um banco de dados relacional através de um *framework* de persistência.

PALAVRAS CHAVE: Ensino, Engenharia de Software, Banco de dados

Abstract

What is presented here is a tool to help the learning process of developing computational systems. This tool allows visual modeling of computational systems, through the use of class diagrams, following the UML notation. Based on this, it generates in an authomated manner the corresponding code (in the *Smalltalk* environment), already including suport for the persistence of the involved data over a relational database, using a persistence framework for that. KEYWORDS: Teaching, Software Engineering, Database systems

Lista de Figuras

2.1 Um exemplo de estrutura de um banco de dados relacional 7

3.1 Um exemplo de modelo de objetos, segundo a notação UML 12

4.1 Uma sugestão para o mapeamento genérico. Retirado de W. Ambler 2003 20

6.1 O diagrama de classes da representação em memória de um documento XMI 39

6.2 As classes responsáveis por representar uma hierarquia de classes segundo o paradigma OO 42

6.3 As classes que armazenam as informações sobre um modelo relacional . . 43

6.4 Os geradores de esquemas relacionais 44

6.5 O componente de geração de instruções DDL 45

6.6 O componente responsável pela geração do suporte à persistência 48

6.7 Um exemplo de interface gerada pelo sistema 51

6.8 O componente de geração de interfaces 51

6.9 O fluxograma de execução do Gerador de Sistemas 54

6.10 O sistema exemplo modelado - uma clínica simples 55

6.11 A configuração para geração do projeto exemplo 57

6.12 A geração do sistema em andamento 57

7.1 A interface do modelador de sistemas 60

7.2 A barra de açoes do modelador 61

7.3 A barra de elementos do modelador 61

7.4 Um classe no modelador 62
7.5 O menu da classe 63
7.6 A tela de adição de atributos à classe 63
7.7 Uma herança 64
7.8 Uma anotação 64
7.9 Uma associação 65
7.10 A tela de geração de sistema 66

Lista de Tabelas

6.1 Classes geradas pela ferramenta e seus respectivos métodos 56
6.2 Tabelas geradas pela ferramenta 58

Sumário

1 Introdução **1**

2 Bancos de dados relacionais **5**

2.1 Conceito 5

2.2 Modelo relacional 6

2.2.1 Estrutura 6

2.2.2 Integridade 7

2.2.3 Manipulação dos dados 8

3 Programação Orientada a Objetos **9**

3.1 Definição 9

3.2 Histórico 9

3.3 Conceitos básicos 10

3.4 Polimorfismo 11

3.5 Modelo de objetos 12

4 Mapeamento do paradigma OO para o modelo relacional **13**

4.1 Introdução 13

4.1.1 Correspondência entre os conceitos do paradigma OO e o modelo relacional 14

4.2 O processo de mapeamento 15

4.3 Mapeamento de classes, hierarquias e atributos 15

4.3.1 Atributos 15

4.3.2 Mapeamento de classes e hierarquias 16
4.3.3 Mapeamento de propriedades de classes 21
4.4 Mapeamento dos relacionamentos entre classes 22
4.5 Ferramentas de mapeamento OO-relacional 23

5 Características e tecnologias utilizadas na ferramenta desenvolvida 25
5.1 Visão geral 25
5.2 Linguagem adotada 26
5.2.1 Sobre o Smalltalk 26
5.2.2 Sobre o Squeak 27
5.3 Notação utilizada para modelagem de sistemas 27
5.3.1 Recursos da UML aceitos 28
5.4 Tipos de dados 29
5.5 Mapeamento do modelo OO para o relacional 30
5.6 GLORP 30
5.6.1 Persistência 32
5.6.2 Manipulando dados 35
5.6.3 Maiores informações 37
5.7 XMI 37

6 O Gerador de Sistemas 38
6.1 Visão geral 38
6.2 Documento XMI 38
6.3 Diagrama de classes 40
6.4 Representação do modelo relacional 41
6.5 Gerador de Esquema Relacional 43
6.6 Gerador de comandos DDL 45
6.7 Gerador de classes 46
6.8 Gerador da camada de persistência 47
6.9 Gerador de Interfaces 50

6.10 Gerador de projetos 52
6.11 Fluxograma da geração de um projeto 53
6.12 Um exemplo de geração de sistema 53

7 O Modelador de Sistemas 59
7.1 Visão geral 59
7.2 Interface 59
7.2.1 Diagrama de classes 60
7.2.2 Barra de ações 60
7.2.3 Barra de elementos 61
7.3 Elementos da modelagem 62
7.3.1 Classes 62
7.3.2 Heranças 64
7.3.3 Anotações 64
7.3.4 Associações 65
7.4 Gerando um sistema 66
7.5 Validações 66
7.6 Estrutura interna 67

8 Sobre o trabalho 70

9 Conclusão e trabalhos futuros 74

Referências Bibliográficas 78

10 Anexos I
10.1 Anexo I - Fonte do sistema exemplo gerado pela ferramenta I
10.2 Anexo II - Fontes da ferramenta desenvolvida XXV
10.3 Anexo III - Artigo CLXVI

Capítulo 1

Introdução

Um sistema computacional (também chamado de programa ou *software*) é uma entidade lógica que pode ter diversas finalidades: desde a solução de um problema complexo de cálculo estrutural até apenas puro entretenimento. Ele pode ter uma representação física (em um sistema para identificação de funcionários, o leitor de digitais) ou ser completamente abstrato (um agente de software rodando em redes de comunicação).

Pode-se considerar que um sistema é composto por várias camadas, cada uma com sua finalidade. Uma das possíveis composições pode incluir as seguintes camadas [Larman 2002]:

Apresentação - Interface com o usuário do sistema (telas, etc.).

Aplicação - Controle de sessões de usuários, estados do sistema, fluxos de execução, etc.

Domínio - As regras e serviços que o sistema deve respeitar e prover.

Infra-estrutura de negócio - Serviços de baixo nível disponibilizados e utilizados pelo sistema.

Serviços técnicos - Serviços de alto nível como persistência de dados[1] e segurança.

Fundação - Elementos como banco de dados, rede de comunicação, etc.

[1] Armazenar os dados em meio físico não-volátil

O desenvolvimento de sistemas computacionais é o processo de, dado um problema, analisá-lo, definir um sistema que o resolva (ou auxilie na sua resolução) e por fim implementá-lo. Esta atividade relaciona-se diretamente com capacidade de processamento dos computadores, a qual, à medida que evoluiu, permitiu o surgimento de ferramentas e recursos (abstrações como objetos[2]) nas linguagens de programação que simplificaram o processo.

Porém, mesmo com esta evolução, a atividade é considerada complexa para alguém começando a familiarizar-se com a mesma. Este fato é observado em alunos de disciplinas de programação de caráter introdutório, em cursos como Ciências da Computação e Sistemas de Informação. As dificuldades enfrentadas por estes são a motivação para este trabalho.

Desenvolveu-se neste trabalho uma ferramenta para auxiliar o aprendizado sobre desenvolvimento de sistemas. Esta destina-se àqueles não familiarizados com o processo. Logo deve ser simples de usar, com recursos suficientes para permitir o desenvolvimento de sistemas computacionais de baixa complexidade. Convém salientar que ela não visa ser utilizada sem a supervisão de alguém experiente no desenvolvimento de sistemas. Ela não implementa aspectos que visem ensinar aqueles que a utilizem sobre o desenvolvimento de sistemas, como caixas explicativas, demonstrações de como mapeiam-se elementos do mundo real para elementos em um sistema computacional, etc.

Ao falar-se em desenvolvimento de sistemas, diversos tópicos estão envolvidos, dependendo do processo (UP[3], XP[4], etc.) e do paradigma de programação (Orientado a Objetos[5], procedural, etc.) adotado.

A criação de uma ferramenta capaz de considerar todos os processos e paradigmas acabaria sendo muito extensa. Logo, este trabalho foca no paradigma orientado a objetos (OO) e nos seguintes tópicos relacionados ao desenvolvimento de software:

- Modelagem de sistemas através de Modelos de Objetos;

[2] Veja a seção 3.3

[3] Unified Process - um processo de desenvolvimento de software organizado em ciclos iterativos [Larman 2002]

[4] Extreme Programming - um processo de desenvolvimento de software orientado pelas necessidades do usuário, que busca adaptar-se facilmente à mudanças de requisitos [What is Extreme Programming? 2004]

[5] Veja a seção 3.1

- parte básica da implementação;
- persistência de dados.

O paradigma de programação OO foi adotado dada sua grande aceitação e popularidade, assim como sua capacidade em representar de forma simples entidades do mundo real e suas relações.

Os *Modelos de Objetos*[6] nada mais são que representações gráficas dos elementos do paradigma OO que compõe um sistema. Eles dão uma visão da parte estática deste, que inclui os elementos que fazem parte do sistema e as relações entre os mesmos.

A persistência de dados é um dos aspectos mais importantes em grande parte dos sistemas computacionais. Para a persistência de dados decidiu-se pelo uso de um banco de dados que segue o modelo relacional[7]. A utilização de um banco de dados deve-se ao interesse em apresentar e familiarizar (em caráter introdutório) o usuário a uma das tecnologias mais utilizadas na Computação.

A adoção de um banco de dados relacional em detrimento a um banco objeto-relacional (ou mesmo orientado a objetos) deve-se ao interesse em demonstrar a quem utiliza a ferramenta as diferenças na representação dos dados entre os dois paradigmas (OO e relacional), assim como familiarizar o usuário com o modelo de representação de dados mais utilizado pelos bancos de dados atuais.

Não adota-se um processo de desenvolvimento de software, deixando esta escolha a cargo do usuário e seu orientador.

A ferramenta desenvolvida pode ser definida em três componentes principais:

- Um modelador de sistemas segundo o conceito de Modelo de Objetos, utilizando um diagrama de classes conforme a notação UML[8].
- Um analisador e gerador que, a partir do diagrama de classes, gere o código básico de um sistema compatível com o diagrama, assim como um modelo relacional sobre

[6] Veja a seção 3.5
[7] Maiores informações na seção 2.2.
[8] Unified Modeling Language - Veja a seção 5.3

o qual os dados envolvidos pelos elementos da modelagem possam ser persistidos.

- Um sistema de suporte a persistência dos dados sobre o modelo relacional elaborado.

Este relatório compõe-se de três partes principais: uma revisão bibliográfica dos assuntos envolvidos; uma descrição detalhada da ferramenta desenvolvida; e por fim uma análise sobre o resultado do trabalho, levantando problemas, soluções e propostas para outros trabalhos.

A revisão bibliográfica está divida em três capítulos: nos capítulos 2 e 3 revisam-se conceitos relacionados ao modelo relacional e à Programação Orientada a Objetos. No quarto capítulo faz-se uma análise detalhada sobre as diversas formas e o processo de mapeamento entre os paradigmas OO e relacional, analisando as vantagens e desvantagens de cada abordagem.

A descrição da ferramenta é composta por um capítulo introdutório apresentando as principais características e descrevendo as tecnologias adotadas. O capítulo seguinte trata do “Gerador de Sistemas”, descrevendo sua estrutura e funcionamento. Por último descreve-se a ferramenta de modelagem (o “Modelador de Sistemas”), detalhando sua interface e estrutura interna.

A última parte divide-se em dois capítulos: no capítulo 8 discorre-se sobre o trabalho desenvolvido, comentando problemas encontrados, soluções para estes problemas, etc. No capítulo 9 trata-se das conclusões a respeito do trabalho e sugestões para trabalhos futuros.

Capítulo 2

Bancos de dados relacionais

2.1 Conceito

Um sistema gerenciador de banco de dados pode ser definido como: *"um sistema de armazenamento de dados baseado em computador; isto é, um sistema cujo objetivo global é registrar e manter informação"* [Date 1986]. Seu objetivo é servir como suporte a outros sistemas, fornecendo os dados necessários para que estes executem as tarefas às quais foram destinados.

A maneira como este sistema organiza e manipula o banco de dados (o conjunto de informações pelo qual é responsável) dá origem a diversos paradigmas (ou "modelos"), dentre os quais o mais difundido é o relacional. Sua popularidade deve-se à simplicidade, consistência, facilidade de uso e firmes fundações matemáticas que o validam [Stoimenov 1998].

Outros modelos surgiram mais recentemente, como o orientado a objetos, objeto-relacional e os bancos de dados XML[1].

[1] Extensible Markup Language

2.2 Modelo relacional

O modelo relacional preocupa-se com três aspectos principais em relação aos dados: estrutura, integridade e manipulação. Ele foi proposto por E. F. Codd em 1970, sendo o primeiro modelo formal para bancos de dados. Anterior a este existiam modelos como o hierárquico e de rede [Silberschatz, K. Korth e Sudarshan 1999].

2.2.1 Estrutura

O modelo relacional compõe-se de relações (também chamadas “tabelas”) que representam (de forma parcial ou completa) entidades (abstratas ou concretas) que possuam características em comum. As relações são consideradas matematicamente como conjuntos.

Cada entidade em uma relação é representada por uma tupla (ou “linha”) contendo atributos (ou “colunas”) que caracterizam a entidade.

Todas as tuplas de uma relação devem possuir os mesmos atributos. Uma mesma relação não deve possuir duas tuplas com os mesmos valores para todos os seus atributos, pois trata-se de um conjunto, que não admite valores repetidos por definição.

Os atributos de uma tupla têm tipo fixo e este deve ser simples (escalar ou “atômico”) como cadeias de caracteres, blocos de texto, números, etc. Uma tupla não pode ser composta por outras tuplas ou dados complexos (estruturas de dados). Um atributo de uma tupla pode ou não ser informado.

Um último elemento integrante do modelo relacional são as visões. Estas são tabelas que não possuem existência própria, sendo constituídas por outras tabelas ou mesmo visões [Date 1995]. Através delas é possível definir diferentes formas de visualizar as informações contidas nas tabelas do esquema relacional (como visualizar apenas tuplas que satisfaçam uma determinada condição).

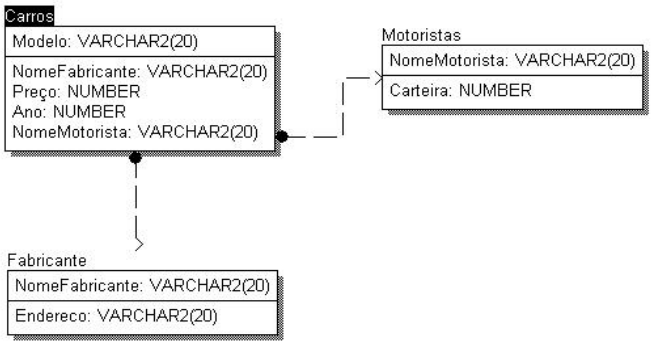

Figura 2.1: Um exemplo de estrutura de um banco de dados relacional

2.2.2 Integridade

Integridade em um banco de dados refere-se à capacidade do mesmo em manter a consistência dos dados que armazena [Silberschatz, K. Korth e Sudarshan 1999]. No modelo relacional a integridade dos dados baseia-se em dois conceitos principais: chaves primárias e chaves estrangeiras [Date 1995].

Para identificar uma tupla de uma relação, define-se uma chave primária. Esta é um valor único associado a cada tupla que a diferencia das demais na relação a que pertencem. Este valor é representado através de um ou mais atributos da tupla. Não é necessário que esta identificação seja única para o banco de dados como um todo.

Para relacionar as tabelas, o modelo relacional fornece as chaves estrangeiras. Através destas define-se um conjunto de atributos em uma relação que representam o valor da chave primária de uma tupla em outra (ou em alguns casos da mesma) relação. Com isto consegue-se estabelecer qualquer associações de grau 1:1, 1:N e N:N[2].

Outras formas de integridades também são fornecidas pelos bancos de dados, como validações de domínio, asserções, etc. Para informações sobre as mesmas, pode-se consultar [Date 1995].

[2]Veja um exemplo na figura 2.1

2.2.3 Manipulação dos dados

A manipulação dos dados no modelo relacional é definida pela álgebra relacional. Os detalhes a respeito do funcionamento da mesma não serão tratados aqui. Cabe apenas ressaltar que a manipulação resume-se a inserir, alterar, excluir e recuperar tuplas das relações.

Para determinar sob quais tuplas as operações irão operar utilizam-se predicados da álgebra relacional que permitem escolher determinadas tuplas com base nos valores dos seus atributos. É possível também operar sobre mais de uma relação ao mesmo tempo ao executar uma determinada manipulação de dados.

Nos bancos de dados não é comum utilizar-se a álgebra relacional, mas sim a SQL[3] para manipulação dos dados. Esta é uma linguagem declarativa com forma similar à algebra relacional porém sintaxe mais simples.

Um exemplo de consulta em algebra relacional que retorna o nome e idade dos pacientes que sejam do sexo feminino é:

$$\pi_{NOME,IDADE}(\sigma_{SEXO='F'}(PACIENTES)) \tag{2.1}$$

Já em SQL, a consulta ficaria na seguinte forma:

SELECT NOME, IDADE FROM PACIENTES WHERE SEXO='F'

Para mais informações a respeito da manipulação de dados em bancos de dados relacionais pode-se consultar [Date 1995] e [Silberschatz, K. Korth e Sudarshan 1999].

[3]SQL - Structured Query Language

Capítulo 3

Programação Orientada a Objetos

3.1 Definição

Programação Orientada a Objetos pode ser definida como *"a programação que é implementada pelo envio de mensagens a objetos"* [Pinson 1988].

Tendo como base esta definição, podemos afirmar que este estilo de programação consiste em identificar objetos (abstratos ou concretos) em um domínio e definir mensagens (métodos) para os mesmos. A combinação de mensagens a um ou mais objetos permite chegar a solução de um problema.

3.2 Histórico

Simula I (1962-65) e Simula 67 (1967) foram as primeiras linguagens orientadas a objeto(OO). Simula 67 introduziu conceitos como objetos e classes. Estas linguagens foram desenvolvidas por Ole-Johan Dahl e Kristen Nygaard partindo de uma necessidade de Nygaard, que precisava de uma ferramenta que permitisse uma descrição e simulação de sistemas homem-máquina complexos. Em 1961 surgiu a idéia de desenvolver uma linguagem que podesse ser utilizada tanto para descrever sistemas quanto comandar um computador [Nygaard e Dahl].

A partir de Simula, um grupo de pesquisadores da Xerox desenvolveu

o Smalltalk[1], estendendo os recursos de Simula pela integração de uma interface gráfica com o usuário e a execução interativa de programas. Mais tarde (na década de 80) Bjarne Strostrup começa a desenvolver C++, introduzindo conceitos de Simula na linguagem C [Nygaard e Dahl].

3.3 Conceitos básicos

Objeto É o conceito básico do paradigma OO. Representa uma entidade de um determinado domínio (problema). Tecnicamente é uma estrutura de dados que possue um comportamento e um estado definido. Ele abstrai e encapsula detalhes sobre si que não são pertinentes ao sistema (elementos privados), deixando visível apenas aquilo que o sistema requer (elementos públicos).

Atributos São características associadas aos objetos que definem um *estado* ao mesmo. [Rumbaugh 1991] define que um atributo deve ser um valor de dados puro, que não seja representado por um objeto. Esta definição, porém, não é completamente adequada a certas linguagens, como Smalltalk, onde o conceito de dados puros não existe (tudo é objeto).

Métodos Representam mensagens que um objeto pode receber, e em resposta executa uma seqüência de ações que podem alterar seu estado. Os métodos podem ou não conter parâmetros que guiem sua execução.

Classe É um conjunto de objetos que apresentam a mesma estrutura e comportamento, porém valores de atributos diferentes. Os objetos são sempre instâncias de alguma classe. Uma classe porém pode não possuir instâncias (classe abstrata). As classes também podem herdar a estrutura de outras classes através do conceito de *herança*.

Herança No paradigma OO, uma classe pode derivar uma ou mais classes. As classes derivadas (*subclasses*) herdam da classe que derivam (*superclasse*) os métodos e

[1]Maior informações sobre o Smalltalk na seção 5.2.1

atributos. É possível a uma classe herdar características de mais de uma classe utilizando o conceito de “herança múltipla” (este porém não é suportado pela maioria das linguagens OO).

Relacionamentos São a maneira pela qual os objetos que compõe um sistema relacionam-se. Podem ser associações (não há dependência de existência entre os objetos) ou composições (um objeto compõe-se de outros objetos, atrelando a existência destes a sua própria existência).

Encapsulamento Consiste na separação dos aspectos externos de um objeto, que são acessíveis por outros objetos, dos aspectos internos de implementação, escondidos dos outros objetos [Rumbaugh 1991]. O encapsulamento auxiliar a tornar os objetos independentes da forma como outros objetos são implementados, permitindo, por exemplo, que altere-se a implementação interna de um objeto sem afetar os demais objetos presentes no sistema.

3.4 Polimorfismo

Polimorfismo é um recurso das linguagens OO onde pode-se enviar uma mensagem a um objeto (desde que o mesmo a suporte) sem ter conhecimento da classe exata a que o mesmo pertence. Dessa forma, diferentes objetos de diferentes classes podem responder a uma mesma mensagem de diferentes formas [Pinson 1988].

Outro aspecto do polimorfismo é a possibilidade de passar objetos de diferentes tipos como parâmetros para outros objetos. Em Smalltalk isto fica claro pela possbilidade de um paramêtro poder ser um objeto de qualquer classe. Já em outras linguagens, como Java, o tipo do parâmetros deve respeitar a hierarquia de classes para poder haver polimorfismo (o objeto deve ser da classe ou alguma das subclasses do tipo especificado no parâmetro).

3.5 Modelo de objetos

Um modelo é uma abstração criada para entender um problema antes de implementar uma solução. Um *modelo de objetos* descreve a estrutura dos objetos que compõe um sistema - suas identidades, relacionamentos, atributos e operações (métodos) [Rumbaugh 1991].

O objetivo ao construir um modelo de objetos (figura 3.1) é capturar os conceitos do mundo real que são importantes para a aplicação da qual se está construindo o modelo. Ele é representado por um diagrama contendo as classes, arranjadas em hierarquias, compartilhando comportamentos e atributos. As classes também estão relacionadas entre si (associações, composições, etc.).

Este modelo não captura os aspectos dinâmicos (eventos e seqüências de operações no decorrer do tempo) e funcionais (as funções exercidas pelo sistema, independente da forma que são feitas internamente) [Rumbaugh 1991].

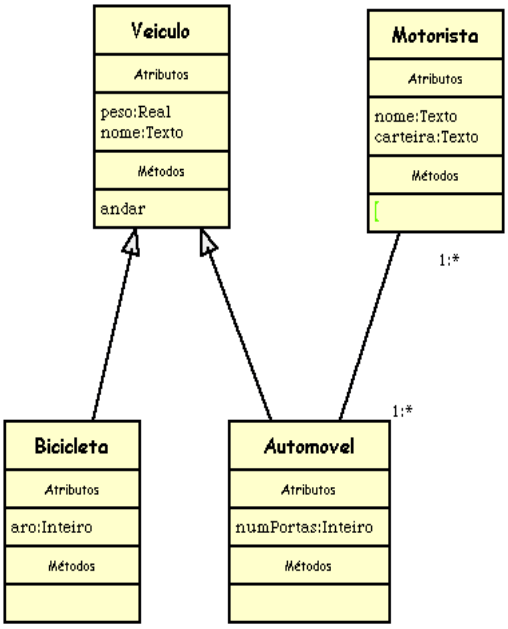

Figura 3.1: Um exemplo de modelo de objetos, segundo a notação UML

Capítulo 4

Mapeamento do paradigma OO para o modelo relacional

4.1 Introdução

"The act of determining how objects and their relationships are persisted in permanent data storage, in this case relational databases." [W. Ambler 2003]

Um dos paradigmas mais utilizados para o desenvolvimento de sistemas computacionais é o paradigma orientado a objetos. Este, introduzido no capítulo anterior, permite o desenvolvimento de sistemas robustos e com manutenção simplificada, graças a conceitos como encapsulamento[1] e herança.

Quando surgiu, como toda nova tecnologia, anunciou-se que ele seria capaz de sanar todos os problemas existentes no desenvolvimento de sistemas ("tudo é objeto"). Dizia-se que a persistência orientada a objetos em bancos de dados era apenas uma questão de tempo [Ambler 2003][2].

Porém a principal forma de persistência dos bancos de dados continua a ser o modelo relacional, por razões já citadas anteriormente (fundações matemáticas, simplicidade, consistência e facilidade de uso) [Stoimenov 1998]. Alternativas a estes

[1] Definido na seção 3.3

[2] No decorrer deste trabalho, os artigos de Scott Ambler são citados diversas vezes. Não foi possível, durante a confecção deste, obter o livro lançado pelo autor. A referência ao mesmo, porém, está presente na bibliografia ([W. Ambler 2003])

surgiram, como os bancos de dados orientados a objetos e os bancos de dados objeto-relacional. Estes, porém, ainda não alcançaram a difusão e popularidade dos bancos relacionais.

Face a isto, vê-se que há uma impedância no desenvolvimento dos sistemas: de um lado temos o paradigma OO e do outro o modelo relacional. Estes tratam as informações e sua organização de forma diferente. Portanto, a abordagem adotada para um em geral não pode ser aplicada diretamente ao outro.

Integrar as duas abordagens é um fator crucial para simplificar o trabalho dos desenvolvedores. Para tal, surgiu o conceito de *mapeamento objeto-relacional*. Neste, desenvolve-se uma camada de persistência (um componente de software) que permite a um sistema orientado a objetos armazenar suas informações em um modelo relacional.

4.1.1 Correspondência entre os conceitos do paradigma OO e o modelo relacional

Cabe em um primeiro momento discorrer como os conceitos do paradigma OO correspondem aos conceitos do modelo relacional:

Classes são representadas por tabelas[3] (uma ou mais tabelas, dependendo da abordagem adotada).

Objetos são as linhas das tabelas que representam as classes.

Atributos são as colunas das tabelas.

Relacionamentos estabelecem-se como relacionamentos baseados em chaves estrangeiras entre as tabelas.

Herança é um dos aspectos mais complexos de representar sob o modelo relacional, dado o conceito de polimorfismo presente no paradigma OO. Adiante no capítulo detalha-se o problema.

[3] A fim de simplificar o texto, os termos “tabela”, “coluna” e “linha” serão adotados quando for-se referir aos termos-equivalentes “relação”, “atributo” e “tupla”

4.2 O processo de mapeamento

O processo de mapeamento pode ser dividido em duas grandes etapas: o mapeamento de classes, hierarquias e atributos e o mapeamento das relações entre as classes.

4.3 Mapeamento de classes, hierarquias e atributos

4.3.1 Atributos

Os atributos das classes são mapeados como colunas no modelo relacional. Este mapeamento envolve atributos de tipo simples, como cadeias de caracteres, números, datas, horários, valores monetários, etc. Atributos que são outras classes do domínio ou tipos complexos são mapeados como outras tabelas, e o atributo torna-se um apontador (chave estrangeira).

No paradigma OO (Orientado a Objetos), todo objeto possui um identificador único e global no sistema, que não é parte de seus atributos, send intríseco à linguagem. No modelo relacional, porém, este conceito não existe. No mapeamento, costuma-se adotar (para simplificar o mapeamento e posterior persistência dos dados) um identificador único arbitrário gerado internamente para cada objeto (por comodidade, um valor inteiro), persistido no banco de dados, e este fica armazenado internamente no objeto para o suporte a persistência [W. Rahayu et al. 2000].

Um problema relacionado a estes identificadores arbitrários é como gerá-los. Diversas alternativas foram propostas [W. Ambler 2000], dentre as quais citam-se:

Obter o maior valor do identificador entre os objetos da classe e incrementá-lo.

Manter uma tabela responsável por armazenar o valor do último identificador de cada classe.

Ter uma tabela para cada classe persistida que contenha o último identificador gerado.

GUIDs / UUIDs - utiliza-se um hash[4] de algum identificador único do computador onde o sistema está rodando associado a data/hora para obter um valor que identifica de forma única o objeto.

Sistemas proprietários - aqui destacam-se os campos de auto-incremento e geradores de sequenciais presentes em bancos de dados como Oracle[5], Firebird[6], PostgreSQL[7].

HIGH / LOW IDs - supõe que o identificador de um objeto é composto por um valor superior e um inferior. O superior é gerado a cada sessão e o inferior a cada novo objeto persistido no banco de dados.

Sobre os atributos, convém salientar que nem todos precisam ser persistidos, pois muitas vezes relacionam-se a aspectos importantes apenas durante uma sessão da qual o objeto faz parte, não tendo importância para outras sessões [W. Ambler 2003].

4.3.2 Mapeamento de classes e hierarquias

O mapeamento das classes e suas hierarquias para tabelas é o aspecto mais complexo do mapeamento, dado que o modelo relacional não oferece suporte ao conceito de herança. Quatro alternativas principais [W. Ambler 2003] existem para representar as hierarquias de classes, cada qual com vantagens e desvantagens.

4.3.2.1 Uma tabela para cada classe concreta

Nesta abordagem considera-se cada classe concreta (não abstrata) a ser persistida como uma tabela. Esta abordagem é eficiente quando trabalha-se com consultas e atualização de objetos de forma individual ou limitada as “classes folhas” (classes que não possuem subclasses). Um sistema de persistência sobre esta abordagem também é relativamente simples.

[4] Um número curto gerado a partir de outra informação - uma cadeia de caracteres por exemplo - que a identifica de forma próxima à unica

[5] http://www.oracle.com

[6] http://firebird.sourceforge.net/

[7] http://www.postgresql.org

A busca das informações para um objeto nesta abordagem é a mais rápida e simples, pois não necessita relacionar tabelas - os dados estão contidos em uma única tabela. O desempenho também é favorecido pela tabela conter apenas as informações dos objetos da classe em consulta [W. Ambler 2003].

Porém esta abordagem também apresenta problemas [W. Ambler 2003] sob determinadas situações como:

- Caso inclua-se um novo atributo em uma classe, é necessário alterar todas as tabelas que representam as classes concretas abaixo da mesma para conter o novo atributo.
- Um objeto que muda de papel dentro do sistema - a alteração do mesmo é custosa, pois envolve copiar todas as informações de uma tabela para a outra.
- Ela viola os conceitos de normalização[8].
- Por último, ela não suporta diretamente o polimorfismo - o relacionamento entre tabelas baseia-se no conceito de chaves estangeiras e estas não podem ser estabelecidas com mais de uma tabela ao mesmo tempo (o que seria necessário para o caso de uma tabela relacionando-se com a tabela equivalente a superclasse de n-classes).

Esta abordagem tem seu principal uso em casos onde a sobreposição entre classes é limitada (hierarquias simples) e mudanças de tipos dos objetos são raras [W. Ambler 2003].

4.3.2.2 Uma tabela por classe

Nesta alternativa, cada classe do sistema (concreta ou abstrata) é mapeada para uma tabela no banco de dados. Cada tabela contém apenas os atributos relativos à classe que representa - os atributos herdados de outras classes ficam nas tabelas responsáveis por representar aquelas classes.

Entre as tabelas de uma hierarquia são estabelecidas relações (através de chaves estangeiras) que identificam quais registros nas tabelas representam um deter-

[8] [Date 1995] descreve o conceito de normalização

minado objeto. Utiliza-se o mesmo identificador para toda a hierarquia de classes (nomeado, por exemplo, como o nome da classe mais superior mais um complemento, como "ID"). Dessa forma fica simples relacionar as diversas classes da hierarquia para compor as informações dos objetos.

Um artifício nesta abordagem é utilizar um atributo na classe mais alta na hierarquia que defina a qual das classes descendentes aquele registro pertence. Dessa forma é possível identificar, a partir do registro na classe superior, de qual classe descendente ele faz parte.

Esta abordagem possui como vantagens [W. Ambler 2003]:

- excelente suporte ao polimorfismo, pois ao recuperar-se linhas de uma tabela inclui-se os objetos correspondente à classe desta tabela e todas as suas subclasses;
- facilidade de execução de operações envolvendo objetos de diferente classes, porém com mesmas classes superiores - uma alteração nos atributos de suas classes superiores é simplificada;
- a modificação das classes (adição, remoção de atributos) é simplificada, pois envolve alterar apenas uma única tabela;
- representa os dados de uma forma normalizada.

As desvantagens dessa abordagem estão relacionadas a profundidade e complexidade da hierarquia. Quanto mais classes, maior a quantidade de tabelas no sistema e mais custosas podem ser tornar as atualizações ou recuperação de dados dos objetos (muitos relacionamentos entre tabelas). Para tentar sanar este problema, pode-se adotar o uso de visões[9] no banco de dados, que encarregam-se de "camuflar" esta divisão em *n-tabelas* em uma única tabela para cada classe concreta [W. Ambler 2003].

4.3.2.3 Uma tabela por hierarquia

Aqui transforma-se toda uma hierarquia de classes em uma única tabela, em geral nomeada segundo a classe mais alta. Neste caso a tabela contém os atributos de

[9] Definidas na seção 2.2.1

todas as classes da hierarquia, além de conter uma coluna funcionando como identificador de qual classe pertence o objeto representado por aquela linha.

As vantagens desta abordagem são [W. Ambler 2003]:

- simplificação da estrutura do banco de dados, que fica reduzida a poucas tabelas;
- permite realizar a recuperação e atualização dos dados de forma simplificada, tanto de objetos individuais quanto de grupos de objetos (inclusive de atributos pertencentes a classes com subclasses).
- inclusão simplificada de novas classes - basta adicionar as colunas as tabelas.
- suporte ao polimorfismo

Porém ela peca ao ignorar os conceitos de normalização. Também pode levar a um gasto extra do espaço de armazenamento do banco de dados, pois cada linha da tabela possui todos os atributos de todas as classes da hieraquia. Por último, as tabelas podem crescer de forma muito rápida por representarem diversas classes do sistema [W. Ambler 2003].

4.3.2.4 Utilizar uma estrutura de tabelas genérica

Esta abordagem introduz um mapeamento onde as informações sobre a estrutura das classes ficam guardadas em meta-tabelas ao invés de dar origem a outras tabelas. Uma estrutura de tabelas é sugerida na figura 4.1.

Vê-se que cada classe é representa por um único registro na tabela *Class* e um ou mais registros na tabela *Attribute*. Os valores dos objetos armazenados ficam na tabela *Value*. Esta abordagem é diferenciada das demais porque pode tratar também os aspectos de relações entre classes, que não são considerados nas abordagens anteriores.

Esta alternativa é extremamente flexível, porém não comporta-se bem quando o volume de dados cresce e a manipulação de informações é complexa [W. Ambler 2003].

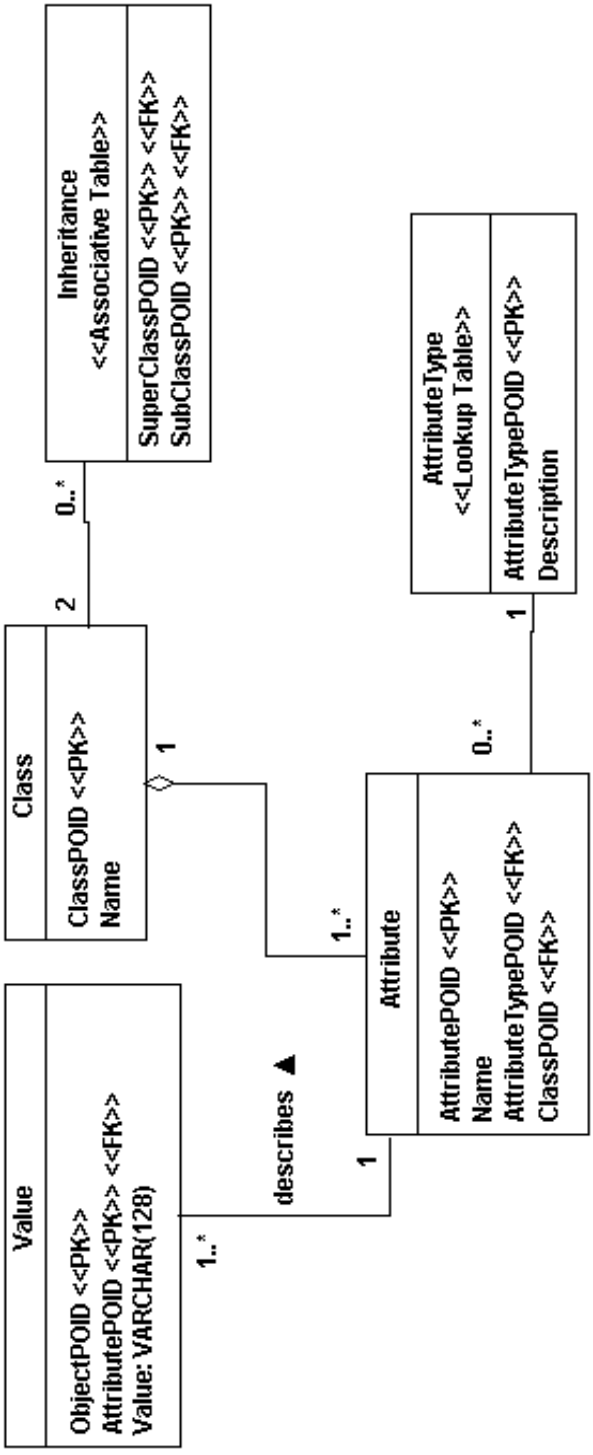

Figura 4.1: Uma sugestão para o mapeamento genérico. Retirado de W. Ambler 2003

4.3.2.5 Herança múltipla

O mapeamento de herança múltipla de um modelo OO para o modelo relacional dá-se de forma natural para as abordagens citadas, bastando seguir as regras definidas. A única questão que aparece refere-se ao mapeamento de uma tabela por classe, em relação ao estabelecimento das chaves estrangeiras indicando a herança. Neste caso, deve-se estabelecer mais de uma herança (chave-estrangeira) para garantir a integridade dos dados.

4.3.3 Mapeamento de propriedades de classes

Uma característica do paradigma OO é que uma classe pode conter atributos relacionados a ela, como entidade classe, não relacionados a suas instâncias (os objetos) e estes atributos muitas vezes precisam também ser persistidos.

Existem diversas maneiras [W. Ambler 2003] de mapear os atributos de classe para o modelo relacional, dentre os quais os mais utilizados são:

- Uma tabela contendo uma única linha, onde cada coluna corresponde a um atributo da classe - o acesso e atualização dos dados é rápido e simples, porém esta forma pode dar origem a muitas tabelas.

- Uma única tabela, contendo uma coluna para cada atributo de cada classe do sistema - neste caso, o número de tabelas adicionais é de apenas uma, porém problemas de concorrência (mais de um processo tentando alterar o valor das colunas) tem grande possibilidade de ocorrer.

- Uma única tabela genérica com múltiplas linhas - os atributos são guardados em tuplas (classe, atributo, valor). Esta forma evita os problemas de concorrência do modelo anterior e requer apenas uma tabela adicional.

4.4 Mapeamento dos relacionamentos entre classes

O mapeamento de relacionamentos entre classes para o modelo relacional não apresenta diversas maneiras como no caso do mapeamento das classes e sua hierarquia. Cada um dos tipos de relacionamento (um para um, um para vários, vários para vários) possui uma ou no máximo duas formas de serem implementados (o que pode diferenciar a forma de mapeamento é a necessidade da visibilidade no relacionamento ser uni ou bidirecional).

Relacionamento um para um Neste tipo de relacionamento, um atributo na tabela funciona como uma chave estrangeira para a outra tabela, estabelecendo o relacionamento entre as duas. Caso deseje-se um relacionamento bidirecional, ambas as tabelas possuirão colunas funcionando como chaves estrangeiras uma para a outra.

Relacionamento um para vários Aqui o fato de optar-se por um relacionamento uni ou bidirecional muda a forma como ele é representado. No caso unidirecional, basta definir uma coluna na tabela representante do lado não-unitário do relacionamento que aponte para a linha da tabela do lado unitário. Caso deseje-se uma visibilidade bidirecional, é necessário criar uma tabela que contenha as chaves primárias de cada uma das tabelas integrantes do relacionamento, onde cada linha representa um relacionamento as classes.

Relacionamento vários para vários Neste tipo, cria-se uma tabela auxiliar que conterá como colunas as chaves primárias das outras tabelas e onde cada linha representa um relacionamento entre as tabelas.

A diferença entre uma composição e uma associação no mapeamento dá-se pelas regras de integridade (chaves estrangeiras) que podem utilizar deleção em cascata, por exemplo[10]. Caso o mapeamento seja a um banco relacional que não suporte tais aspectos, cabe ao sistema de persistência adotado manter a integridade e consistência dos dados.

[10]Para maiores informações sobre o conceito de deleção em cascata, veja [Date 1995]

4.5 Ferramentas de mapeamento OO-relacional

Diversas ferramentas existem para o suporte ao mapeamento entre o paradigma OO e o relacional no mercado. Estas concentram-se em dois pontos principais: desempenho e flexibilidade.

Para dar flexibilidade ao mapeamento, ao ponto de permitir que modelos relacionais não projetados para persistências de sistemas OO sejam utilizados, as ferramentas deixam a cargo do usuário estabelecer o mapeamento. Em geral este mapeamento é definido através de um documento XML, onde basta alterar o mesmo para mudar o comportamento e regras de persistência do sistema.

As ferramentas não apresentam (salvo poucas exceções), mecanismos para um mapeamento automático entre o modelo OO e o relacional. O mais próximo de uma automatização que as ferramentas disponibilizam em geral é a capacidade de gerar as classes e tabelas de acordo com o esquema de mapeamento definido. Isso decorre do fato de buscarem o desempenho como um dos seus pontos chaves. E para tal, muitas vezes um mapeamento automatizado não garante o mesmo (dadas as particulares pertinentes a cada sistema).

Dentre as ferramentas analisadas, citam-se:

Hibernate É uma ferramenta de persistência para Java. Ela utiliza um arquivo XML para definir o mapeamento e pode gerar as classes e tabelas a partir do mesmo. Usa reflexão para recuperar os atributos a persistir, logo não necessita de programação adicional nas classes persistidas.

Castor Apresenta recursos semelhantes ao Hibernate (sendo também para Java), porém adiciona a capacidade de persistir as informações em documentos XML ao invés de um banco relacional.

InfoObjects Semelhante as anteriores, porém suporta C++ além de Java.

Torque - apresenta recursos equivalentes ao Hibernate.

CocoBase, intelliBO, Visual BSF são outras ferramentas existentes equivalentes as anteriores, porém proprietárias.

Capítulo 5

Características e tecnologias utilizadas na ferramenta desenvolvida

5.1 Visão geral

A ferramenta desenvolvida neste trabalho pode ser dividida em duas grandes partes: uma visual, cujo objetivo é permitir a modelagem de sistemas simples segundo o conceito de *Modelos de Objetos*[1], usando a notação UML. Esta será nomeada como *"Modelador de Sistemas"*.

A outra parte, denominada *"Gerador de Sistemas"* encarrega-se de, dada uma modelagem, gerar:

- O código básico da mesma (classes, *getters* e *setters*).
- Um esquema de banco de dados relacional sobre o qual os dados envolvidos podem ser persistidos - incluindo visões para simplificar a visualização dos dados.
- Uma camada de persistência (mapeando entre os paradigmas OO e relacional) para a persistência dos dados.
- Interfaces visuais para inclusão/remoção e alteração de entidades do sistema - utilizando a camada de persistência gerada pela própria ferramenta.

[1] Seção 5.3

5.2 Linguagem adotada

Como linguagem sobre a qual a ferramenta foi desenvolvida escolheu-se o Smalltalk. Dentre as várias implementações (Dolphin Smalltalk, VisualWorks, etc.) escolheu-se o Squeak[2].

5.2.1 Sobre o Smalltalk

> "Smalltalk foi projetado para ser fácil de aprender e usar. É uma linguagem expressiva que usa um subconjunto simples de linguagens, substantivos e verbos humanos. Algo que todos os humanos podem relacionar-se com. Isto facilita uma clara expressão de soluções que mapeiam-se muito bem com o pensamento humano. Esta é uma das razões que leva programas em Smalltalk a serem de um terço a um meio do tamanho de programa escritos em outras linguagens populares. Programas em Smalltalk tipicamente podem fazer duas a três vezes mais trabalho com a mesma quantidade de código. Em alguns casos programas em Smalltalk podem fazer muito mais com muito menos código." ([Lount 2004])

As características que levaram a escolha do Smalltalk como linguagem a ser utilizada foram:

- Em Smalltalk, *"Tudo é um objeto"*. Inteiros, *strings*, *booleans*, definições de classes e blocos de código são representado por objetos. Logo mantém-se uma maneira uniforme de interagir com os elementos da linguagem (através de mensagens enviadas para os objetos).

- Um objeto pode receber qualquer tipo de mensagem, sendo só verificado se ele pode recebê-la durante a execução do programa - isto evita "typecasts"[3] no código, como ocorre em outras linguagens.

[2] http://www.squeak.org
[3] Conversão de um objeto em outra classe

- Smalltalk é um ambiente integrado de desenvolvimento e execução - graças a isto pode-se programar e testar o código de forma simples (aos poucos, conforme ele é construído).

- Toda a linguagem está disponível no ambiente - para qualquer classe é possível olhar seu código e analisar seu funcionamento, além de também poder alterá-la se houver interesse.

5.2.2 Sobre o Squeak

> "Squeak é uma implementação aberta e portável do Smalltalk-80. Sua máquina virtual é escrita inteiramente em Smalltalk, tornando fácil depurá-la, analisá-la e modificá-la. Para obter uma performance prática, um tradutor produz um programa C equivalente cuja performance é comparável a outras implementações comerciais" [About Squeak 2004]

A escolha do Squeak deveu-se a:

- Está disponível para várias plataformas (Windows, BeOS, vários Unix - incluindo Linux, etc.);

- É uma implementação de código aberto e gratuita para uso tanto acadêmico quanto profissional;

- Uma grande quantidade de componentes (*pacotes* no linguajar do Squeak) disponíveis, alguns deles utilizados neste projeto[4].

5.3 Notação utilizada para modelagem de sistemas

Como a ferramenta desenvolvida dispõe de uma parte relacionada a modelagem de sistemas (utilizando o *Modelo de Objetos*), adotou-se a UML como linguagem de notação para a modelagem.

[4]Os pacotes disponíveis para o Squeak podem ser encontados no SqueakMAP, localizado em http://minnow.cc.gatech.edu/squeak/2726

A UML pode ser definida como *"Uma linguagem para especificação, visualização, construção e documentação de artefatos de sistemas de software, assim como modelagem de negócios e outros tipos de sistemas"*. As primeiras noções do que mais tarde tornariam-se a UML foram dadas em 1994 por Grady Booch e Jim Rumbaugh. Eles combinaram as notações de diagrama de seus métodos de desenvolvimento de software: OMT (Object Modeling Technique) e Booch. [Rumbaugh 1991]

Em 1997 a UML (após várias contribuições por outras pessoas) foi adotada pelo OMG[5] como um padrão para modelagem de sistemas OO.

Não vai entrar-se aqui em detalhes sobre a UML e sua notação. Apenas os aspectos importantes relativos à ferramenta desenvolvida serão citados à medida que esta for descrita. Para informações e a especificação da UML, pode-se consultar a documentação do OMG.

5.3.1 Recursos da UML aceitos

A especificação UML propõe uma série de recursos para criação de modelos de objetos, porém muitos não são pertinentes à modelagens simples (foco deste trabalho). Com este fato em mente, limitam-se os componentes da notação aceitos na modelagem aos seguintes: classes, heranças, atributos e métodos de objetos, relações de associação (com graus um para um, um para vários e vários para vários).

Quanto as heranças, suporta-se apenas heranças simples. Isto deve-se a dois motivos principais: a linguagem adotada não possui suporte ao conceito de heranças múltiplas; e este é considerado complexo e raramente utilizado dentro do modelo OO, logo não seria adequado ao usuário alvo da ferramenta.

No caso das relações, não trata-se a possibilidade de uma relação possuir um ou mais atributos associados a mesma. Na modelagem de sistemas, quando isto ocorre, ao passar-se para a fase de implementação em geral faz-se os atributos passarem a ser membros de um dos elementos da associação, ou cria-se uma terceira entidade res-

[5] *"O Object Management Group (OMG) é um consórcio aberto, sem fins lucrativos, que produz e mantém especificações para a indústria de informática para interoperabilidade entre aplicações empresariais"* - http://www.omg.org

ponsável por manter os dados. E isto já pode ser feito durante a própria modelagem do sistema.

5.4 Tipos de dados

Os atributos das classes deverão ser tipados[6] para que possam ser persistidos no banco de dados. Porém os tipos serão limitados a um conjunto específico, escolhido para representar mais fielmente os tipos de dados presentes no mundo real.

Estes tipos são:

Inteiro - um número pertence ao conjunto dos números inteiros;

Texto - uma seqüência qualquer de letras, números e símbolos, representada no banco de dados por algum tipo que o mesmo possua que não forneça limitações de tamanho;

Racional - um número que podem ser representado na forma de fração - armazenados como o tipo de dados de maior capacidade de armazenamento para ponto flutuante existente no banco de dados;

Real - números que não podem ser representados na forma de fração - armazenados como o tipo de dados de maior capacidade de armazenamento para ponto flutuante existente no banco de dados;

Data , Hora - momento no tempo;

ValorMonetario - representa valores monetários (decimais de duas casas);

Imagem - armazena imagens (fotos, desenhos, etc.) - mapeado como tipo *BLOB*[7] no banco de dados;

SimNao - representa um valor que pode ser verdadeiro (Sim) ou falso (Não) - o tipo *booleano*;

[6] Ter um tipo de dados definido
[7] Binary Large Object

Qualquer - pode conter dados arbitrários (arquivos binários, etc.) - mapeados como tipo *BLOB* no banco de dados.

5.5 Mapeamento do modelo OO para o relacional

O processo de mapeamento é feito de forma automática, a partir do diagrama de classes. O sistema foi projetado para que qualquer uma das quatro técnicas discutidas anteriormente possa ser utilizada, e daquelas discutidas três implementadas: uma tabela por classe concreta, uma tabela por hierarquia de classes e uma tabela por classe. A adotada no projeto é a de uma tabela por hierarquia de classes.

5.6 GLORP

Para este projeto adotou-se, para a persistência dos dados, uma ferramenta já existente, para poder concentrar o esforço em outros aspectos da ferramenta (como o mapeamento automático para o modelo relacional e a ferramenta visual para modelagem).

Duas alternativas foram encontradas para a persistência de objetos em bancos de dados relacionais no Smalltalk: *Tantalus* e *GLORP*. Inicialmente a idéia foi utilizar o Tantalus, pois era a única ferramenta conhecida e apresentava características que interessavam ao projeto (como não necessitar de código adicional nas classes para que fossem persistidas).

Porém o desenvolvedor do Tantalus decidiu parar de desenvolvê-lo, deixando-o inacabado (sem suporte por exemplo a relacionamentos muitos para muitos). Cogitou-se inicialmente tomar o mesmo e desenvolver os recursos que faltavam. Porém, após uma nova pesquisa, tomou-se conhecimento do GLORP (*Generic Lightweight Object-Relational Persistence*), outra ferramenta para persistência, que veio a ser adotada.

O GLORP pode ser definido como uma camada de mapeamento objeto-relacional para Smalltalk. Possui versões disponíveis para várias implementações do Smalltalk (VisualWorks, VisualAge, Dolphin Smalltalk, Squeak). É um projeto liderado

por Alan Knight e tem por objetivo prover um *framework*[8] simples, porém poderoso, para ler e gravar objetos de bancos de dados relacionais[9].

O GLORP apresenta diversas características que interessam ao projeto, como:

Baseado em transações Esta característica foi utilizada para criar o conceito de "sessão" na ferramenta desenvolvida.

Gravações transparentes Para o usuário basta registrar o objeto junto ao sistema de persistência para que o mesmo seja persistido. Uma vez registrado o objeto em uma sessão, não há necessidade de executar comandos específicos para gravar as alterações feitas nos dados.

Cache interno dos objetos Garante que se um objeto for recuperado duas vezes, a instância em memória nos dois casos será a mesma - logo não há problemas de consistência de dados.

Utilização de *proxies* - Os objetos relacionados a um objeto recuperado do banco de dados serão carregados à medida que for necessário. Para tal ele utiliza o conceito de "proxy", onde para os atributos não recuperados ainda ele instancia uma classe que ao ser acessada converte-se no atributo correto.

Gerência do mapeamento centralizada em uma única classe, *escondendo* do usuário a complexidade de funcionamento do framework de persistência.

Suporte a herança, polimorfismo e associações.

Como pontos negativos sobre o GLORP temos a persistência apenas sobre o PostgreSQL e Oracle; ser baseado em código - não apresenta uma forma de esta-

[8] *"Um esqueleto de implementação de uma aplicação ou de um subsistema de aplicação, em um domínio de problema particular. É composto de classes abstratas e concretas e provê um modelo de interação ou colaboração entre as instâncias de classes definidas pelo framework. Um framework é utilizado através de configuração ou conexão de classes concretas e derivação de novas classes concretas a partir das classes abstratas do framework" - A. Wirfs-Brock [WIA 91]*

[9] Maiores informações sobre o GLORP estão disponíveis em http://www.glorp.org

belecer o mapeamento que não seja através de código. Existe o interesse em desenvolver um suporte ao mapeamento dinâmico[10], porém não há nada concreto até o momento.

5.6.1 Persistência

No GLORP, toda configuração da persistência fica definida por uma instância de um *DescriptorSystem*. Esta classe deve ser especializada para cada esquema de persistência que se deseja estabelecer. Dentre os aspectos que precisam ser definidos, dois destacam-se: a definição das tabelas onde serão persistidos os dados e as classes que serão persistidas.

Para cada tabela sobre a qual os dados serão persistidos cria-se um método nomeado como "*tableFor<nomeDaTabela>*", onde *<nomeDaTabela>* corresponde ao nome da tabela no banco de dados. Este método recebe como parâmetro um descritor ao qual devem ser adicionados os campos, chaves primárias e chaves entrangeiras da tabela. Um exemplo (este código foi gerado pela própria ferramenta desenvolvida):

```
tableForCONSULTAS: umDescritor
        | intid tpCtrlInt medico paciente data |
        intid  (umDescritor
                createFieldNamed: 'intid'
                type: platform serial)
                bePrimaryKey.
        tpCtrlInt _ umDescritor
                createFieldNamed: 'tpCtrlInt'
                type: platform string.
        medico _ umDescritor
                createFieldNamed: 'medico'
                type: platform integer.
        paciente _ umDescritor
```

[10] Que poderia ser modificado durante a execução do sistema, sem alteração do código do mesmo, apenas de um modelo de mapeamento - como um documento XML, por exemplo

```
            createFieldNamed: ’paciente’
            type: platform integer.
    data _ umDescritor
            createFieldNamed: ’data’
            type: platform date.
    umDescritor
            addForeignKeyFrom: medico
                    to: ((self tableNamed: ’PESSOA’)
                    fieldNamed: ’intid’).
    umDescritor
            addForeignKeyFrom: paciente
                    to: ((self tableNamed: ’PESSOA’)
                    fieldNamed: ’intid’)
```

Da mesma forma, para cada classe a ser persistida, deve-se definir um método da forma *“descriptorFor<nomeDaClasse>”*, que recebe um descritor ao qual adicionam-se informações sobre a classe a persistir. Exemplo (também gerado pela ferramenta):

```
descriptorForConsultas: umDescritor
    | tabela |
    umDescritor
        table: (self tableNamed: ’CONSULTAS’).
    umDescritor
        addMapping: (DirectMapping
                from: #data
                to: ((self tableNamed: ’Consultas’)
                        fieldNamed: ’data’)).
    umDescritor
        addMapping: (DirectMapping
                from: #intid
```

```
                to: ((self tableNamed: 'Consultas')
                        fieldNamed: 'intid')).
    (self typeResolverFor: Consultas)
        register: umDescritor
        keyedBy: 'Consultas'
        field: ((self tableNamed: 'CONSULTAS')
                fieldNamed: 'tpCtrlInt').
    umDescritor addMapping: (OneToOneMapping new
        attributeName: #medico;
        referenceClass: Medico;
        mappingCriteria: (Join
                from: ((self tableNamed: 'CONSULTAS')
                        fieldNamed: 'medico')
                to: ((self tableNamed: 'PESSOA')
                        fieldNamed: 'intid'))).
    umDescritor addMapping: (OneToOneMapping new
        attributeName: #paciente;
        referenceClass: Paciente;
        mappingCriteria: (Join
                from: ((self tableNamed: 'CONSULTAS')
                        fieldNamed: 'paciente')
                to: ((self tableNamed: 'PESSOA')
                        fieldNamed: 'intid')))
```

O código sua maior parte é auto-explicativo, porém convém comentar o seguinte trecho:

```
 (self typeResolverFor: Consultas)
        register: umDescritor
        keyedBy: 'Consultas'
        field: ((self tableNamed: 'CONSULTAS')
```

```
fieldNamed: 'tpCtrlInt')
```

O código apresentado refere-se a uma persistência baseada no conceito de uma tabela por hierarquia de classes. Este trecho destacado define qual atributo e valor do mesmo identificam os elementos desta classe na tabela que contém a hierarquia da qual ela faz parte (para poder realizar a filtragem dos dados ao recuperá-los do banco de dados).

5.6.2 Manipulando dados

Para manipular dados com o GLORP, é preciso estabelecer uma *"Unit Of Work"*, que nada mais é que uma sessão onde existe uma conexão ao banco de dados e um *cache* dos objetos sendo manipulados na sessão. Neste *cache* ficam guardados tanto os objetos recuperados quanto os objetos novos que serão gravados no banco de dados quando a sessão terminar.

Para estabelecer a *"Unit of Work"*, basta instanciar uma *GlorpSession* para a qual passa-se uma instância do descritor de persistência (sub-classe de *DescriptorSystem*) e uma conexão ao banco de dados (*DatabaseAccessor*). Feito isto, com os métodos *"beginUnitOfWork"* e *"endUnitOfWork"* controla-se o início e fim de uma sessão no GLORP. Adicionalmente tem-se o método *"commitUnitOfWorkAndContinue"*, que permite persistir no banco de dados as alterações feitas durante a sessão e mantê-la aberta para uso posterior.

Para inserir ou excluir objetos estão disponíveis os métodos:

registerNew: registra um novo objeto (que será inserido no banco de dados posteriormente);

delete: remove um objeto da sessão e posteriormente do banco de dados.

Para alterar objetos, o GLORP assume que os mesmos devem ser recuperados e então alterados. Como ao ser recuperado ele já é registrado na sessão, ao terminá-la ele será automaticamente atualizado no banco de dados.

Para recuperar objetos o GLORP disponibiliza quatro métodos básicos:

readOneOf: umaClasse recupera um objeto (qualquer) de uma determinada classe;

readManyOf: umaClasse recupera todos os objetos de uma dada classe e subclasses;

readOneOf: umaClasse where: umaCondicao recupera um objeto de uma classe / subclasses para o qual a condição informada é valida;

readManyOf: umaClasse where: umaCondicao semelhante ao anterior porém retorna uma coleção de objetos.

"umaClasse" é a classe do objeto que deseja-se recuperar. *"umaCondicao"* é um bloco de código Smalltalk onde define-se uma condição que deve ser respeitada na hora de recuperar os objetos. Exemplo:

```
sessao readOneOf: Paciente
        where: [:paciente | paciente nome = 'João'].
```

Ao receber tal código, o GLORP converte o mesmo em uma cláusula SQL correspondente a condição estabelecida (exemplo: *"WHERE NOME="JOAO""*). Isto também é válido para relacionamentos, portanto:

```
sessao readManyOf: Consulta
        where: [:consulta | consulta paciente nome = 'João'].
```

também é uma condição de pesquisa válida. Pode-se também utilizar outros objetos dentro da condição de pesquisa, como:

```
pacienteRecuperado _ sessao readOneOf: Paciente
        where: [:paciente | paciente nome = 'João'].


sessao readManyOf: Consulta
        where: [:consulta | consulta paciente = pacienteRecuperado].
```

É importante destacar que o GLORP suporta polimorfismo na recuperação das entidades. Logo, ao recuperar objetos de uma classe também virão objetos de subclasses da mesma.

5.6.3 Maiores informações

Caso deseje-se mais informações a respeito do GLORP, pode-se acessar o site *http://www.glorp.org*. Entre outros documentos, estão disponíveis tutoriais sobre o mesmo.

5.7 XMI

No primeiro momento do projeto (quando ainda não havia a ferramenta de modelagem visual), foi necessário ter alguma forma de importar modelagens feitas em outras ferramentas para utilizar como testes para a ferramenta de geração.

Após uma pesquisa sobre qual a melhor forma de fazer isso, decidiu-se pela utilização do XMI para fazer a importação das modelagens, já que a maioria das ferramentas de modelagem existentes implementa alguma forma de exportação para o mesmo, além de ser um padrão estabelecido pelo OMG.

O XMI[11] é um padrão da OMG para troca simplificada de informações referentes a metamodelos, utilizando o formato XML. Através do XMI especificam-se diagramas de qualquer gênero (como diagramas de classe no caso da UML).

O XMI não é uma especificação de um formato de XML para metamodelos, mas sim um algoritimo de geração de especificações XML para metamodelos. Dessa forma ele suporta vários tipos de metamodelos (como os da UML). Porém ao mesmo tempo que isto leva a um ganho considerável de flexibilidade e recursos, provoca também uma tendência a incompatibilidade entre os documentos XMI gerados entre algumas aplicações (indo contra o propósito original de permitir a troca simples das informações entre as aplicações). Apesar disto, estas diferenças em geral podem ser contornadas pelos interpretadores dos documentos.

[11] XML Metadata Interchange

Capítulo 6

O Gerador de Sistemas

6.1 Visão geral

O *Gerador de Sistemas* é a parte da ferramenta responsável por tomar uma representação de um sistema segundo o conceito de *"Modelo de Objetos"* e a partir deste gerar: a estrutura básica de código (no ambiente Smalltalk) das classes do sistema (incluindo as relações entre as mesmas); uma camada de persistência compatível com a estrutura de classes geradas; um esquema de banco de dados relacional onde as classes possam ser persistidas; e por último interfaces que permitem a visualização e edição dos dados persistidos.

Ele é composto por vários componentes, que são: Documento XMI, Diagrama de Classes, Representação do Modelo Relacional, Gerador de Esquema Relacional, Gerados de comandos DDL[1], Gerador de Classes e o Gerador de Interfaces.

6.2 Documento XMI

Como em um momento inicial do projeto necessitou-se importar documentos XMI, desenvolveu-se um componente capaz de importá-los. Como um documento XMI tem a mesma estrutura sintática de um documento XML, pôde-se utilizar

[1]Data Definition Language - Linguagem utilizada para definir esquemas de bancos de dados

um *parser* já existente no Squeak para ler o arquivo XMI. Porém, desenvolveu-se toda uma estrutura para a representação em memória deste documento, de forma a poder obter informações sobre o mesmo (como classes, atributos, métodos, heranças e relacionamentos).

Esta representação não buscou satisfazer todas as características que um documento XMI pode apresentar mas sim aquelas necessárias ao projeto.

A hierarquia de classes que representa um documento XMI está demonstrada na figura 6.1.

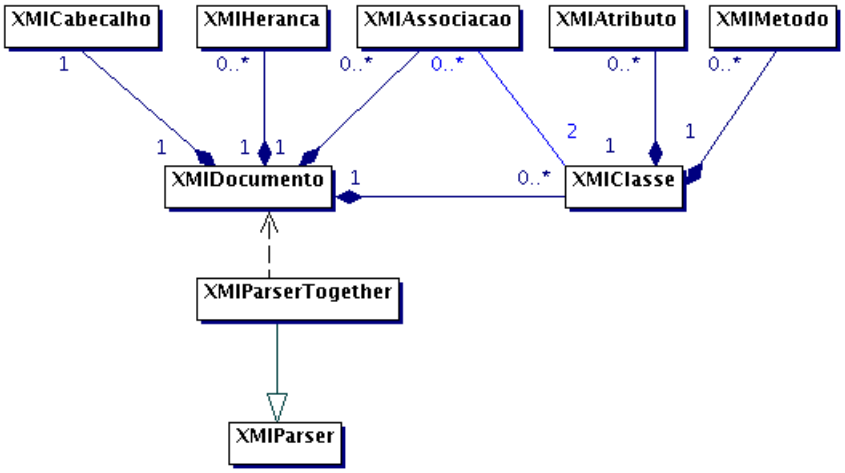

Figura 6.1: O diagrama de classes da representação em memória de um documento XMI

A utilização desta estrutura resume-se em criar um XMIDocumento e através de sua interface adicionar os elementos que o compõe (como classes, heranças, etc.).

Como o formato XMI acaba sendo específico da aplicação que o gera, desenvolveu-se parsers para o XMI gerado pelo Borland Together 6 e para o ArgoUML / Poseidon. Este último porém apresentou problemas na importação de determinadas modelagens devido a erros na geração do documento XMI pelo ArgoUML (determinados elementos não eram gerados) e acabou sendo descartado, ficando apenas o parser para documento exportados no formato *"XMI 1.1 for UML 1.4"* a partir do Together.

O parser para XMI desenvolvido funciona da seguinte forma:

1. Dado um *stream*[2] contendo um documento XMI, é estabelecido um XMLDOMParser, mantido internamente para acessar as informações do documento XMI.

2. O cabeçalho do documento é analisado e as informações contidas no mesmo são armazenadas em um XMICabecalho, anexo ao XMIDocumento sendo gerado.

3. As classes e tipos de dados são então recuperados (sem os atributos). Para cada um é gerado um XMIClasse[3] e são armazenados também os identificadores internos[4] presentes no documento XMI.

4. Os atributos das classes e tipos de dados são então analisados e inseridos nos respectivos XMIClasse. Através do identificador interno determina-se que XMIClasse corresponde ao tipo de dados definido para o atributo.

5. De forma semelhante aos atributos, os métodos das classes são recuperados e adicionados às XMIClasse correspondentes.

6. Após ter as classes recuperadas, as heranças são analisadas e adicionadas (XMIHeranca) ao XMIDocumento.

7. Por último as associações são recuperadas.

6.3 Diagrama de classes

Como todo o funcionamento da ferramenta baseia-se no diagrama de classes do sistema, desenvolveu-se uma estrutura (figura 6.2) para representar o mesmo em memória.

Para representá-lo adotou-se o uso de um grafo. Para tal, utilizou-se uma implementação disponível no sistema *Mundo dos Atores*[5]. Nesta implementação

[2]Fluxo de dados - no caso, um fluxo de caracteres

[3]Não diferencia-se aqui tipos de dados e classes pois para o projeto esta divisão não se fazia necessária

[4]Em um documento XMI, os tipos de dados e classes definidas apresentam um identificador interno - um literal *string* - que o identifica unicamente dentro do documento

[5]http://www.inf.ufsc.br/poo/atores/index.html

de grafo qualquer objeto pode ser adicionado como um vértice e as arestas podem ser "rotuladas" por um objeto qualquer.

No grafo cada vértice constitui uma classe e as arestas são os relacionamentos entre as classes (tanto associações quanto heranças). Para definir qual tipo de relação uma dada aresta constitui estabelece-se como rótulo da mesma um objeto que identifica seu tipo e no caso de associações também contém atributos como cardinalidade e papéis das classes envolvidas(*"roles"*).

Desenvolveu-se, utilizando esta representação, diversos métodos para facilitar a manipulação da hierarquia de classes, como recuperar as sub e superclasses de uma classe, métodos para estabelecer e recuperar os relacionamentos, para copiar atributos de uma classe a todas as suas subclasses, etc.

Esta representação também possui a capacidade de analisar um objeto da classe XMIDocumento e incorporar os elementos contidos no mesmo, gerando automaticamente a hierarquia de classes contida no documento. Assim a instrução:

```
(OORDBMHierarquiaDeClassesComParserXMI new xmi: umDocumentoXMI)
```

já retorna um objeto *OORDBMHierarquiaDeClasses* correspondente ao documento XMI informado.

6.4 Representação do modelo relacional

Desenvolveu-se uma representação (figura 6.3) simples de um esquema relacional baseada em três classes: uma representando tabelas, outra representando visões e uma última representando colunas. As duas primeiras contêm um conjunto de colunas e, no caso das tabelas, um subconjunto que representa sua chave primária.

Uma coluna pode possuir, além do nome e tipo (tipo este um dos tipos básicos definidos anteriormente - "Inteiro", "Texto", etc.), uma chave estrangeira, que é o objeto representante do campo da tabela com a qual a chave relaciona-se.

As visões são estabelecidas através da inclusão nas mesmas de campos

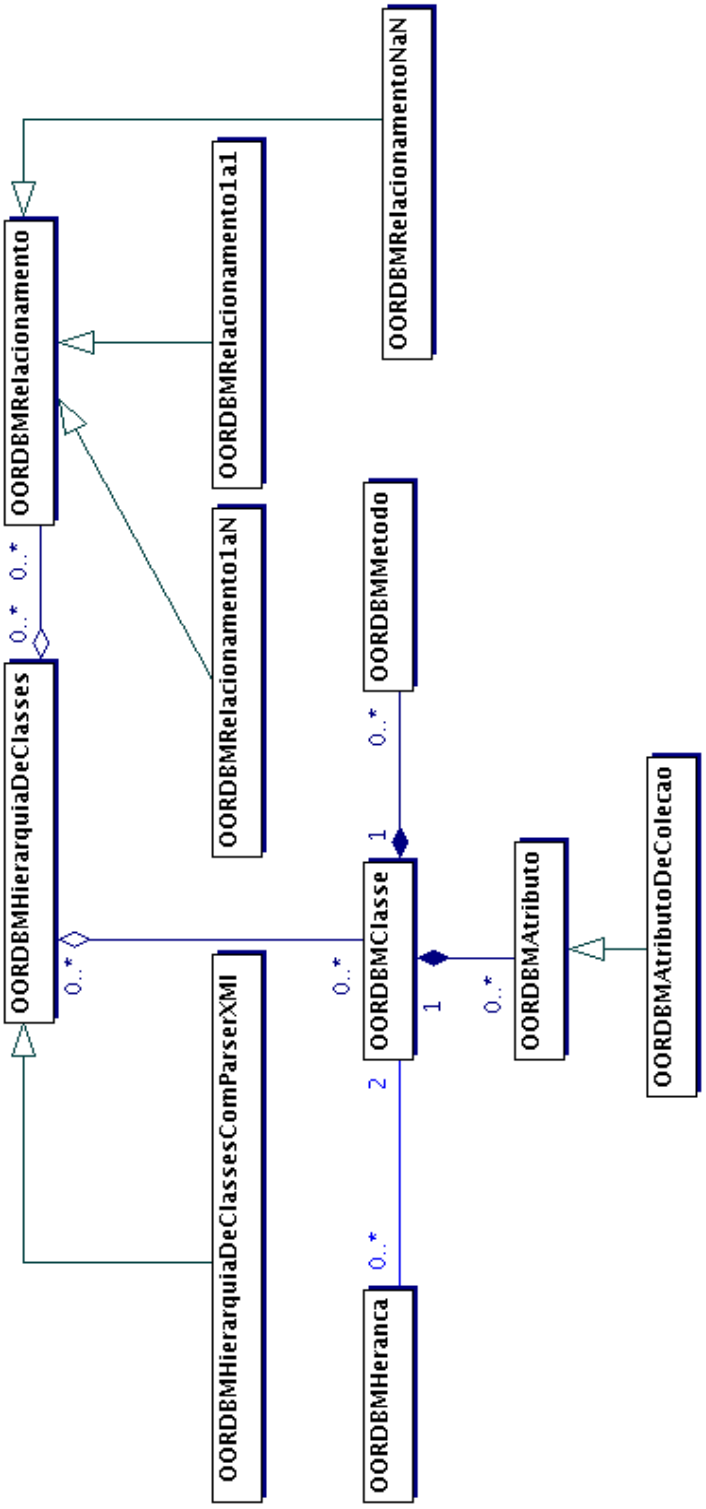

Figura 6.2: As classes responsáveis por representar uma hierarquia de classes segundo o paradigma OO

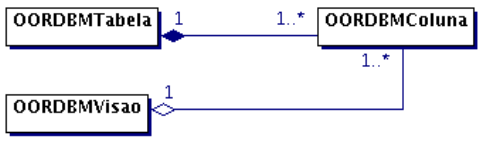

Figura 6.3: As classes que armazenam as informações sobre um modelo relacional

definidos em outras tabelas ou visões, além de uma cláusula[6] SQL que determina os critérios para o estabelecimento da visão. A abordagem mais adequada aqui seria utilizar uma linguagem ou mesmo uma representação em forma de objeto desta clásula, mas para o projeto desenvolvido optou-se por definir diretamente a cláusula SQL, pois satisfazia as necessidades do mesmo.

6.5 Gerador de Esquema Relacional

Com base na pesquisa feita sobre mapeamento entre os paradigmas OO e relacional, desenvolveu-se um componente (figura 6.4) responsável por, dada uma representação de um diagrama de classes (*OORDBMHierarquiaDeClasses*), gerar uma representação deste no modelo relacional.

O componente foi modelado de forma a poder ser utilizado para qualquer uma das quatro modalidades de persistência, das quais foram implementadas três formas: uma tabela por classe concreta, uma tabela por classe (incluindo abstratas) e uma tabela por hierarquia de classes.

O funcionamento do Gerador de Esquema Relacional pode ser descrito através da seguinte seqüência básica de passos:

1. Para cada classe superior da hierarquia (classes que não possuem superclasses), é gerada uma tabela de mesmo nome, contendo apenas uma coluna nomeada “intid”. Esta, de tipo inteiro, é utilizada como chave primária da tabela. Em seguida, para cada atributo destas classes, é verificado se o mesmo é um tipo simples (utilizando

[6]Do tipo “WHERE”

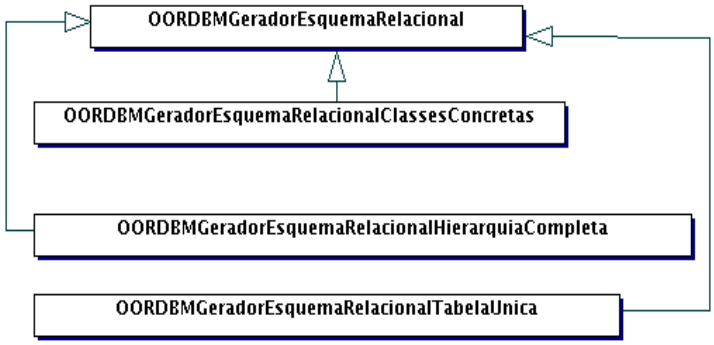

Figura 6.4: Os geradores de esquemas relacionais

a classe *OORDBMTiposBasicos*, que contém uma lista interna dos tipos básicos aceitos). Se for, uma coluna é adicionada a definição da tabela, com o nome e tipo iguais ao do atributo [7]

2. Um processo semelhante é desenvolvido então para as demais classes, variando conforme o tipo de abordagem de mapeamento OO-relacional adotado. No caso de uma tabela por classe concreta, o processo para as classes superiores é realizado de forma idêntica para as demais classes. Na abordagem de uma tabela por classe, além dos passos realizados para as classes superiores também é definida uma chave estrangeira entre o campo “intid” da tabela com o campo “intid” da tabela que representa sua classe superior. Por último, para a abordagem de uma tabela por hierarquia de classes, os atributos das classes descendentes são adicionados à tabela que representa a classe mais alta na hierarquia.

3. Mapeadas as classes e atributos, os relacionamentos são analisados. Para cada relacionamento são definidos os campos (e tabela, se for o caso) necessários para o mapeamento, conforme descrito no capítulo 4, além das devidas chaves estrangeiras.

4. Como último item, podem ser geradas visões sobre o esquema relacional elaborado (a fim de facilitar a visualização dos dados). No sistema desenvolvido este

[7] A conversão para um tipo existente no banco de dados - como de *Imagem* para *BLOB* - é feita pelo *Gerador de comandos DDL*, descrito na seção seguinte.

item ocorre apenas no caso do mapeamento de uma tabela por hierarquia, onde são criadas visões responsáveis por mostrar apenas os “objetos” e atributos de uma determinada classe.

Este componente não tem a capacidade de gerar os comandos DDL para criar o banco de dados. Isto é função de outro componente, o Gerador de comandos DDL, descrito a seguir.

6.6 Gerador de comandos DDL

Este componente (representado pela classe *OORDBMSQL* e apresentado na figura 6.5) gera, dado um conjunto de *OORDBMTabela*, os comandos de DDL do SQL equivalentes a representação de tabelas e campos gerada pelo *Gerador de Esquema Relacional*. Da mesma forma ele também pode gerar os comandos equivalentes para *OORDBMVisoes*.

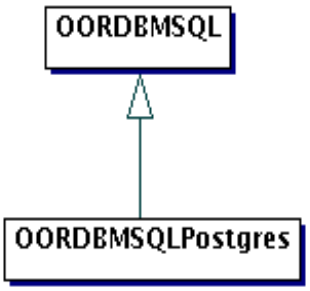

Figura 6.5: O componente de geração de instruções DDL

Ele pode gerar os comandos DDL de duas formas: uma onde toda a tabela é criada de uma única vez e outra onde cada campo é criado isoladamente. Dessa forma é possível atualizar (inserir novos campos) a estrutura de uma tabela.

Para evitar que na hora de gerar as tabelas ocorram problemas devido às chaves estrangeiras (criar uma sem existir ainda a tabela referenciada), ele gera de forma independente os comandos para as tabelas e relacionamentos.

Seu funcionamento interno pode ser descrito da seguinte forma: dada a representação de uma tabela ou visão, ele percorre a mesma montando o(s) comando(s) em SQL/DDL para gerá-la. Através de um dicionário interno determina as definições em SQL/DDL correspondentes aos tipos dos atributos (“Inteiro” para “INTEGER”, etc.). De forma semelhante, os comandos para estabelecimento das chaves estrangeiras também são gerados.

6.7 Gerador de classes

Representado por apenas uma única classe (*OORDBMGeradorDeClasses*), ele toma a representação em memória do diagrama de classes (*OORDBMDiagramaDeClasses*) e realiza todo o processo de geração das classes dentro do ambiente Smalltalk, utilizando-se de reflexão para analisar se as classes já existentes e em caso positivo, quais atributos/métodos já dispõe. Ele realiza este processo em quatro etapas:

1. Inicialmente ele verifica para cada classe se a mesma já existe ou não no sistema. Em caso negativo, ela é criada, sem atributos ou métodos. Em caso positivo, ele apenas verifica se a categoria e superclasse sob a qual ela está é a correta. Caso não seja, faz as devidas alterações.

2. Em seguida analisam-se os atributos de cada classe e geram-se as varíaveis de instância correspondentes aos mesmos (preservando variáveis definidas anteriormente, caso a classe já exista no ambiente). Feito isso, são criados os métodos para definir e recuperar estes atributos (*getters/setters*). Na geração destes, incorporam-se métodos de validação de tipo aos mesmos, de forma que se for passado um “Texto” para um atributo “Inteiro”, um erro será gerado. Os códigos destas validações ficam contidas na classe *OORDBMTiposBasicos*, em um dicionário cujas chaves são os tipos (“Inteiro”, etc.).

3. As associações entre as classe são analisadas e estabelecem-se os atributos correspondentes em cada classe. A existência ou não de um papel associado ao terminal

da relação determina se um atributo deve ou não ser criado (definindo dessa forma a visibilidade do relacionamento). No caso de terminais da associação com cardinalidade unitária, estabelece-se um *getter/setter* no formato de atributo simples, incluindo código para validar o tipo do objeto. Já em terminais com cardinalidade não-unitária, estabelece-se uma coleção como atributo para conter os objetos associados. Esta coleção pode ser da classe *TypedOrderedCollection* ou da classe *TypedLimitedOrderedCollection*, ambas desenvolvidas para o projeto. A primeira restringe o tipo de objeto que pode estar contido na coleção e a segunda restringe além do tipo a quantidade máxima de elementos que podem estar contidos.

4. Por último o gerador analisa a lista de métodos de cada classe. Caso os mesmos ainda não existam nas classes, ele os cria (só a definição, sem um corpo propriamente dito).

6.8 Gerador da camada de persistência

O gerador da camada de persistência (figura 6.6) é responsável por analisar a hieraquia de classes e gerar todo o suporte à persistência da mesma baseado no GLORP.

Da mesma forma que o Gerador de Esquema Relacional, existem três versões deste componente, correspondendo aos três tipos de mapeamento OO-relacional fornecidos pelo Gerador de Esquema Relacional.

O suporte a persistência gerado por este componente consiste em gerar a classe descendente de DescriptorSytem[8] que será responsável por informar ao GLORP como dar-se-á o mapeamento.

O gerador utiliza como fonte de informações uma instância de *OORDBMHierarquiaDeClasses* e uma coleção de *OORDBMTabela* e realiza o seguinte processo:

1. Ele gera a base da classe de persistência, que consiste nos métodos *constructAllClasses* e *allTableNames*, que definem os nomes das classes e tabelas que estarão

[8] Veja a seção 5.6.1

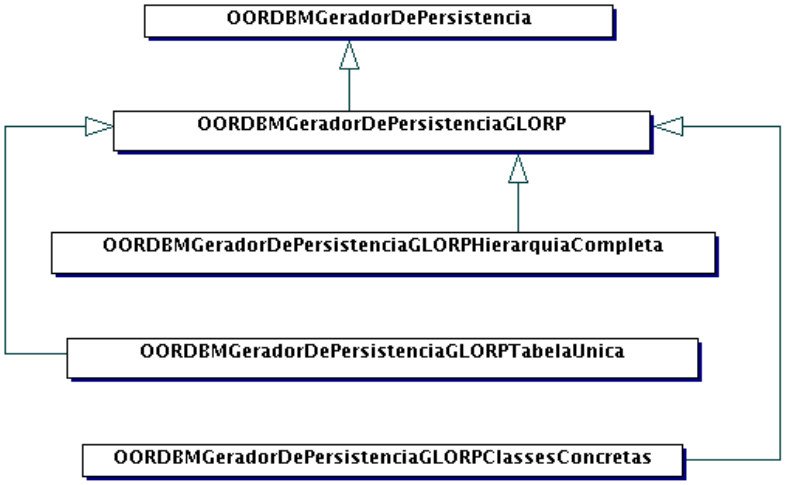

Figura 6.6: O componente responsável pela geração do suporte à persistência

envolvidas na persistência.

2. Em seguida, para cada *OORDBMTabela* ele gera um descritor para mesma, considerando os relacionamentos existentes.

3. Da mesma forma, para cada classe ele gera o descritor correspondente.

Este componente também é responsável por gerar uma classe que "camufla"o GLORP através de uma interface simples. Esta classe trabalha com dois conceitos chaves:

Sessão - tudo o que for relacionado à persistência deve estar contido em uma sessão, que deve ser iniciada e finalizada pelo usuário.

Registro - um objeto só torna-se persistente a partir do momento que é registrado dentro de uma sessão.

Ela contém os seguintes métodos:

- - recuperarEntidadesDoTipo: umaClasse segundo: umaCondicao
- - recuperarEntidadeDoTipo: umaClasse segundo: umaCondicao

- - recuperarEntidade: umaClasse segundo: umaCondicao
- - registrar: umObjeto
- - excluir: umObjeto
- - iniciarSessao
- - gravarEContinuarSessao
- - terminarSessao

Estes são mapeados para métodos de nome semelhante existentes no *GlorpSession* (descritos na seção 5.6.2). O diferencial fica no método iniciarSessao, mostrado a seguir:

```
iniciarSessao
    self conectarBancoDeDados.
    descritor _ DescritorPersistencia new.
    sessao _ GlorpSession new.
    sessao system: descritor.
    sessao accessor: accessor.
    sessao beginUnitOfWork
```

Além do processo de iniciar uma nova ‘‘*Unit of Work*” no Glorp, ele encarrega-se de instanciar a conexão e o descritor de persistência, simplificando a persistência dos dados a uma seqüência de comandos similar a:

```
incluirConsultas

| paciente sessao medico consulta |
sessao _ GerenciadorPesistencia new.
sessao iniciarSessao.
paciente _ sessao recuperarEntidadeDoTipo: Paciente
```

```
    segundo: [:pac | pac nome = 'Paciente1'].
medico _ sessao recuperarEntidadeDoTipo: Medico
    segundo: [:pac | pac nome = 'Rodrigo'].
consulta _ Consultas new paciente: paciente;
    medico: medico; data: Date today.
sessao registrar: consulta.
sessao terminarSessao
```

6.9 Gerador de Interfaces

A função deste componente do Gerador de Sistemas é, a partir do diagrama de classes, gerar interfaces gráficas através das quais os dados contidos nas classes possam ser editados, assim como instâncias das mesmas incluídas ou removidas do sistema.

A geração destas interfaces tem por objetivo facilitar a inclusão e edição de dados pelos usuários em um primeiro momento, pois "esconde" do mesmo os detalhes relacionados a persistência dos dados. Dessa forma, têm-se um contato inicial com o conceito de persistência (dois usuários, por exemplo, podem compartilhar o mesmo banco de dados e ver as alterações realizadas um do outro).

Além disto, ela funciona como uma ferramenta que pode ser incorporada pelo próprio usuário no desenvolvimento do sistema, pois disponibiliza uma forma de visualização e edição dos dados.

Para cada classe é gerada uma tela semelhante à figura 6.7. Ela disponibiliza as ações básicas comuns a sistemas que operam com dados persistidos (opções para navegação como "Próximo" e "Anterior", inclusão e exclusão de registros, etc.). Para gerar esta interface, o componente toma como base um diagrama de hierarquia de classes (*OORDBMHierarquiaDeClasses*).

Em um primeiro momento ele pede ao mesmo que expanda os atributos das classes superiores para as inferiores. Esta operação significa que, para cada atributo que existe em uma dada classe, o mesmo vai ser replicado em todas as suas subclasses.

Figura 6.7: Um exemplo de interface gerada pelo sistema

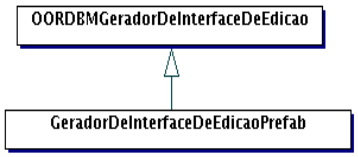

Figura 6.8: O componente de geração de interfaces

Dessa forma, cada classe passa a conter todos os atributos de suas classes superiores além dos atributos definidos nela própria. Esta operação é utilizada para que, quando for gerada a tela, estejam presentes na mesma todos os atributos da classe correspondente, ao invés de apenas os atributos definidos na classe.

O sistema então executa o seguinte processo para cada classe da hierarquia:

1. Gera a estrutura básica (definição) da classe que representa a interface gerada - aqui as subclasses de *OORDBMGeradorDeInterfaceDeEdicao* definem qual será a superclasse.

2. Cria métodos para navegação, edição, gravação e recuperação dos dados entre a interface e o objeto correspondente aos dados visualizados na mesma, considerando a lista de atributos da classe.

3. Estabelece os métodos para iniciar e terminar sessões de persistência, utilizando o Gerenciador de Persistência gerado anteriormente.

4. Elabora a interface gráfica, considerando a lista de atributos para determinar como gerar os componentes para edição dos dados na mesma. Para gerar a interface, ele identifica a quantidade de atributos e tamanho que cada um necessita, determinando assim o tamanho da tela criada. Feito isso, o código correspondente aos componentes que exibem os dados de um determinado atributo é gerado, considerando uma posição inicial calculada de acordo com sua posição na lista de atributos e o tipo de atributo.

O componente foi estruturado (conforme vê-se na figura 6.8) de forma a permitir a utilização de qualquer conjunto de componentes visuais. A classe *OORDBMGeradorDeInterfaceDeEdicao* contém toda a lógica para a geração das interfaces, porém abstrai os detalhes referentes a que classes de componentes visuais são utilizados, ficando isso a cargo das subclasses - no caso *OORDBMGeradorDeInterfaceDeEdicaoPrefab*. Essencialmente as subclasses ficam responsáveis por definir o código que será gerado, de acordo com parâmetros fornecidos às mesmas pelos métodos da classe *OORDBMGeradorDeInterfaceDeEdicao*.

Convém salientar que, como as interfaces geradas são com componentes do próprio Smalltalk, ficam disponíveis para qualquer plataforma na qual a máquina virtual do Squeak é capaz de executar. É preciso apenas que na plataforma desejada, a máquina virtual para a mesma possua suporte a um ambiente gráfico.

6.10 Gerador de projetos

Este é o componente principal do Gerador de Sistemas. É ele que coordena todos os outros componentes, permitindo a geração de um sistema “completo” (considerando o que a ferramenta desenvolvida propõe-se a fazer).

OORDBMProjeto é a principal classe do gerador de projetos. Ela possui três subclasses (*OORDBMProjetoClassesConcretas*, *OORDBMProjetoHierarquiaCompleta*, *OORDBMProjetoTabelaUnica*), conforme o modelo de mapeamento OO-relacional adotado. Estas limitam-se a definir qual tipo de classe deve ser utilizada para gerar as ta-

belas e a camada de persistência.

Em *OORDBMProjeto* é possível especificar os parâmetros utilizados na geração do projeto, que são: a categoria sob a qual as classes serão persistidas; o endereço do servidor de banco de dados onde será criado o banco de dados relacional; o nome do banco de dados a criar; o usuário e senha de acesso ao servidor de banco de dados.

O gerador também pode ser configurado para executar apenas determinadas etapas do processo, podendo combinar os processos de gerar classes, camada de persistência e banco de dados de qualquer forma desejada (gerar apenas as classes e o banco de dados, somente a camada de persistência, etc.).

6.11 Fluxograma da geração de um projeto

A fim de dar uma visão mais global do funcionamento da ferramenta durante a geração de um projeto, apresenta-se na figura 6.9 uma visão completa do processo, identificando as entidades do sistema responsáveis por cada uma das etapas.

Não entrou-se em detalhes sobre as etapas envolvidas em cada um dos subprocessos por estas estarem descritas detalhadamente na seções anteriores e para simplificar a visualização do fluxograma.

6.12 Um exemplo de geração de sistema

Para exemplificar o funcionamento gerador de sistemas, escolheu-se um sistema simples e fez-se a modelagem e posterior geração do mesmo. O sistema escolhido foi uma pequena clínica, da qual criou-se um modelo de objetos seguindo os seguintes critérios:

1. Considerou-se como entidades da clínica os médicos, pacientes e as consultas.
2. Um paciente tem um nome, um cpf e um código de cadastro que é definido pela clínica para identificá-lo.
3. Cada médico tem um nome, um cpf e um crm.

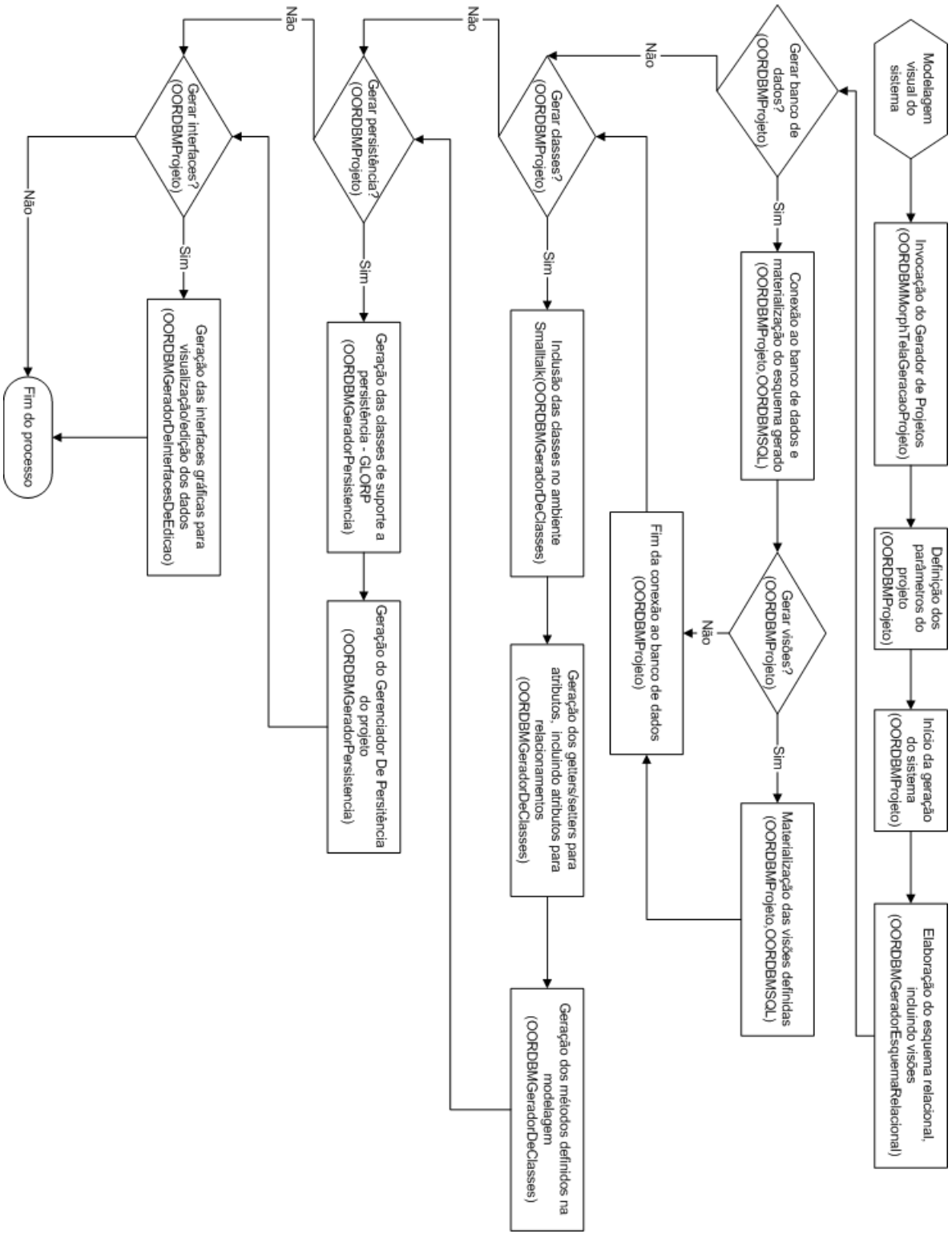

Figura 6.9: O fluxograma de execução do Gerador de Sistemas

4. Médicos e pacientes participam de consultas. Cada consulta é composta por um médico e um paciente, possuindo um momento de tempo (data e hora) e um valor associado.

5. Cada paciente tem consciência das consultas que já fez e pode calcular quanto já gastou com as mesmas.

6. Cada médico sabe que consultas realizou e pode saber quanto já ganhou.

A modelagem alcançada, com base nos critérios acima, é apresentada na figura 6.10. A partir dela foi requisitada a geração do sistema (figuras 6.11 e 6.12). Como resultado obtivemos as seguintes classes e métodos[9]:

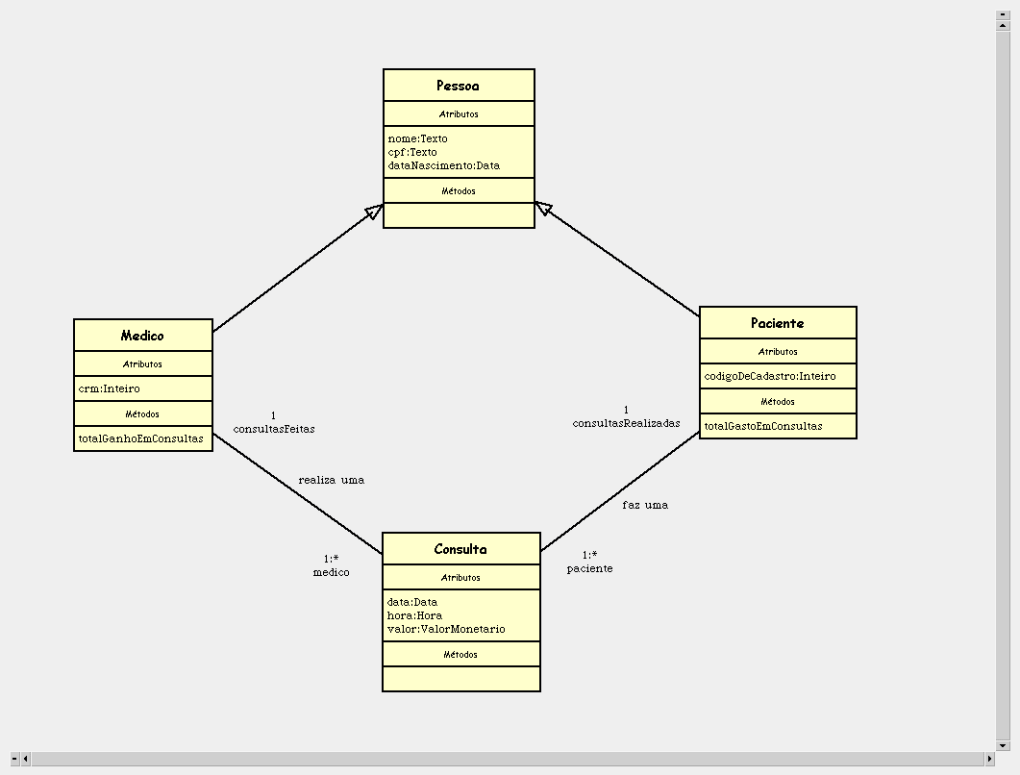

Figura 6.10: O sistema exemplo modelado - uma clínica simples

Já o modelo relacional gerado foi o seguinte:

[9]As classes com formato fixo (sem métodos variando de acordo com a modelagem) foram omitidos da listagem

Tabela 6.1: Classes geradas pela ferramenta e seus respectivos métodos

Classe	Métodos
CMConsulta	data , data: , hora , hora: , intid , intid: , medico , medico: , paciente , paciente: , valor , valor:
CMMedico	consultasFeitas , consultasFeitas: , crm , crm:
CMPaciente	codigoDeCadastro , codigoDeCadastro: , consultasRealizadas , consultasRealizadas: , totalGastoEmConsultas
CMPessoa	cpf , cpf: , nome , nome: , dataNascimento , dataNascimento
CMDescritorPersistencia	allTableNames , constructAllClasses , descriptorForCMMedico: , descriptorForCMMedico: , descriptorForCMConsulta: , descriptorForCMPessoa: , tableForCMCONSULTA: , tableForCMPESSOA: , typeResolverForCMConsulta , typeResolverForCMMedico , typeResolverForCMPaciente , typeResolverForCMPessoa
CMGerenciadorPesistencia	
CMNavegadorDeCMConsulta	
CMNavegadorDeCMMedico	
CMNavegadorDeCMPaciente	
CMNavegadorDeCMPessoa	

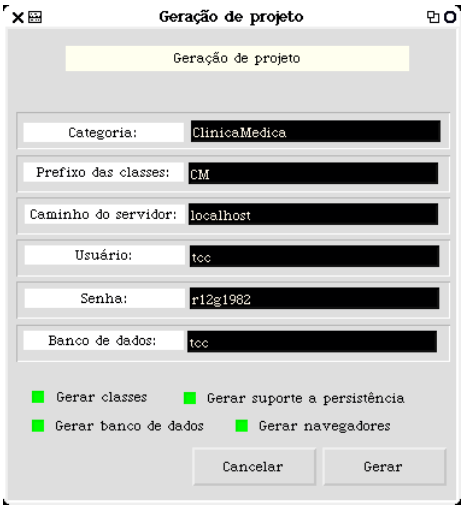

Figura 6.11: A configuração para geração do projeto exemplo

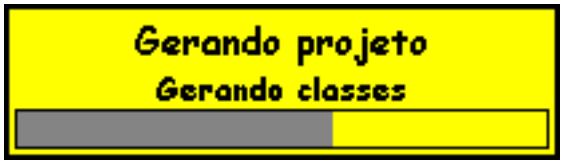

Figura 6.12: A geração do sistema em andamento

Tabela 6.2: Tabelas geradas pela ferramenta

Tabela	**Coluna**	**Tipo**	**Chave Estrangeira**
cmconsulta	intid	serial	
	tpctrlint	text	
	hora	time	
	data	date	
	paciente	integer	cmpessoa
	medico	integer	cmpessoa
	valor	numeric(15,2)	
cmpessoa	intid	serial	
	tpctrlint	text	
	nome	text	
	datanascimento	date	
	cpf	text	
	codigodecadastro	integer	
	crm	integer	

O código gerado para o exemplo seria por demasiado extenso para ser mostrado aqui. Dado este fato, ele está disponibilizado como o Anexo 1 deste trabalho.

Capítulo 7

O Modelador de Sistemas

7.1 Visão geral

A ferramenta visual de modelagem de sistemas desenvolvida neste projeto visa uma interface simples e de fácil uso para a definição da estrutura de um sistema (segundo o conceito de *"Modelo de Objetos"*).

Seguindo a notação UML, a ferramenta permite definir classes e relacionamentos entre estas classes - relacionamentos que podem ser tanto de herança quanto associações de cardinalidades diversas (um para um, um para vários, etc.).

O foco do desenvolvimento foi a facilidade de uso e não a completude da ferramenta em termos dos recursos disponibilizados pela UML. Caso deseje-se isto, já existem diversas ferramentas no mercado (Together e ArgoUML são exemplos). Como o usuário-alvo da ferramenta desenvolvida é leigo ou iniciante na modelagem de sistemas, não necessita de todo o "potencial" disponibilizado pela UML.

7.2 Interface

A interface com o usuário da ferramenta é apresentada na figura 7.1. Ela é composta pelo Diagrama de classes, Barra de elementos e Barra de ações.

Figura 7.1: A interface do modelador de sistemas

7.2.1 Diagrama de classes

É a área "vazia" da interface do modelador. Ali são dispostos os elementos da modelagem. Não há limitação quanto ao tamanho, sendo possível "rolar" o ambiente caso fique muito grande.

7.2.2 Barra de ações

Nela (em destaque na figura 7.2) estão presentes várias ações que podem ser feitas com o diagrama modelado. Estas opções incluem elementos tradicionais de uma ferramenta do gênero como: gravar e recuperar modelagens feitas para arquivos em disco; exportar o diagrama como uma imagem para posterior impressão; eliminar a modelagem atual e iniciar uma nova modelagem.

Porém, a que merece destaque é *"Gerar projeto"*, que funciona como *"ponte"* entre o modelador e o *Gerador de Sistemas*. Esta ação é descrita mais em detalhes adiante neste capítulo.

Figura 7.2: A barra de açoes do modelador

7.2.3 Barra de elementos

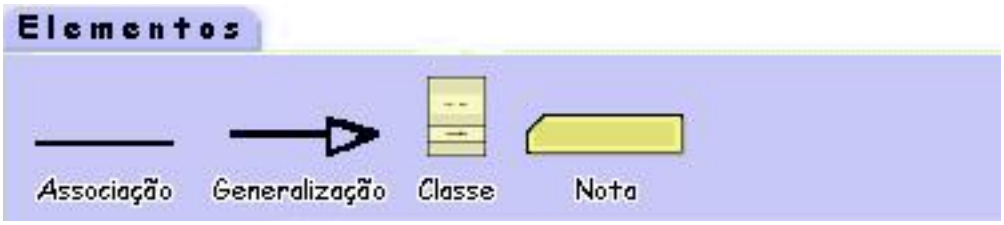

Figura 7.3: A barra de elementos do modelador

Apresentada na figura 7.3, está localizada na parte inferior da tela. Ela contém os elementos que podem fazer parte de uma modelagem. São eles: Classe, Associação, Generalização e Nota.

Para utilizar os elementos, basta selecioná-los e arrastá-los para o diagrama. No caso de elementos que unem outros elementos (heranças e associações), basta conectá-los aos elementos que se deseja unir. Se houver necessidade, pode-se trocar os elementos unidos por uma relação apenas selecionando o terminal desejado e arrastá-lo até o novo elemento.

7.3 Elementos da modelagem

7.3.1 Classes

As classes seguem uma representação gráfica tradicional (figura 7.4) para diagramas UML utilizada pela maioria das ferramentas existentes. Nesta representação, uma classe é composta essencialmente por três partes principais, que comportam: o nome da classe, seus atributos e por último seus métodos (interface).

Figura 7.4: Um classe no modelador

O nome da classe é definido de forma simples, bastando apenas digitá-lo no espaço reservado ao mesmo (por padrão uma classe é nomeada “NovaClasse”).

Os atributos são definidos também apenas digitando seus nomes em forma de lista no espaço destinado. Para especificar o tipo do atributo (seguindo a idéia de tipos simples explicada na seção 5.4) basta apenas adicionar “:” (dois pontos) ao final do nome do atributo e em seguida o tipo que o representa (“Inteiro”, “Texto”, etc.). Excluir um atributo significa simplesmente removê-lo desta listagem.

Os métodos seguem a mesma lógica que os atributos para serem definidos / removidos. Como um método entende-se a sua assinatura segundo a sintaxe do Smalltalk (ex: *“metodoQueSomaUmNumero: umNumero a: outroNumero”*).

A edição de métodos e atributos não sofre qualquer validação durante a modelagem, dando total liberdade ao usuário para modificá-los.

Figura 7.5: O menu da classe

Uma forma adicional de definir atributos é através da opção de menu (figura 7.5) "Incluir atributo"constante no menu do elemento Classe. Nesta opção é exibida uma tela (figura 7.6) auxiliar onde estão listados todos os tipos possíveis para um atributo (esta lista é montada dinamicamente, acessando a classe *OORDBMTiposBasicos* e requisitando a mesma a lista dos possíveis tipos). É possível deixar várias instâncias desta tela abertas (uma para cada classe do diagrama).

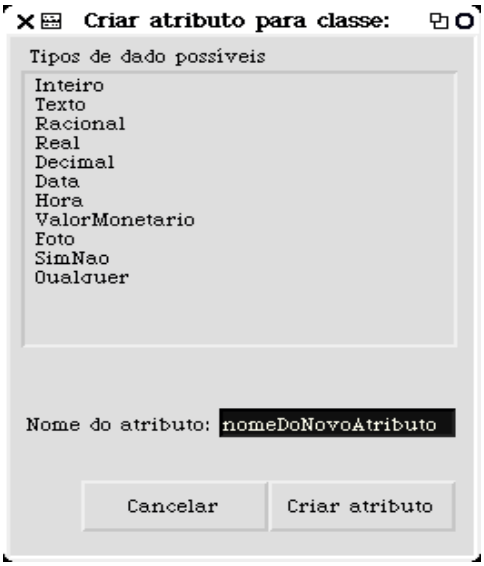

Figura 7.6: A tela de adição de atributos à classe

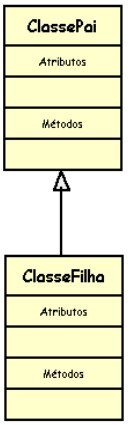

Figura 7.7: Uma herança

7.3.2 Heranças

As heranças (figura 7.7) não apresentam nenhum destaque especial quanto a sua funcionalidade. Apenas liga-se uma classe com a qual deseja-se estabelecer uma relação de herança.

7.3.3 Anotações

São elementos sem significância sintática ou semântica, que servem apenas para permitir ao usuário estabelecer anotações dentro do diagrama (para, por exemplo, saber o que precisa ser feito ainda naquele diagrama ou razão pela qual determinados relacionamentos/heranças foram estabelecidos). Um exemplo de anotação é apresentação na figura 7.8.

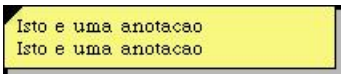

Figura 7.8: Uma anotação

7.3.4 Associações

As associações (figura 7.9) estabelecem-se da mesma forma que as heranças, apenas ligando-se as classes integrantes da associação. Porém dois aspectos diferenciam os relacionamentos: a possibilidade de definir uma cardinalidade e um papel (*"role"*) para os mesmos.

A definição da cardinalidade segue a sintaxe <*limiteInferior*> *:* <*limiteSuperior*> (similar a outras ferramentas de modelagem). O limite superior pode ser * para indicar que não há limite. A cardinalidade é definida no sentido "este terminal da relação aceita *x* elementos da classe conectada ao outro terminal da relação".

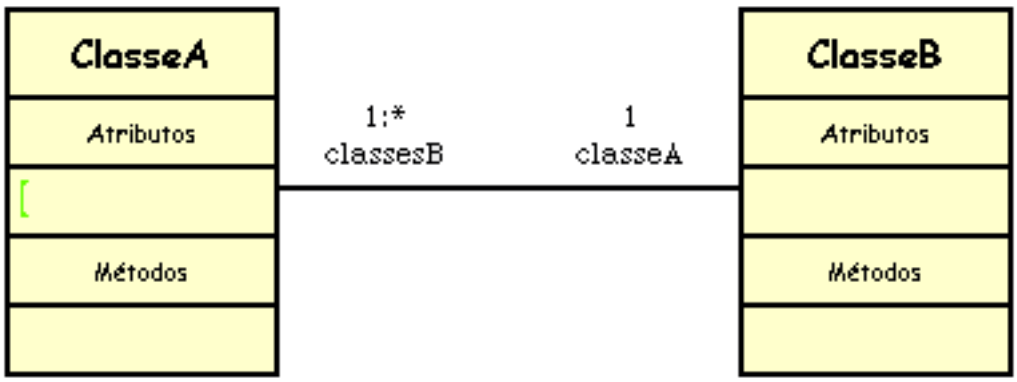

Figura 7.9: Uma associação

Os papéis do relacionamento definem como a entidade do terminal onde está definindo-se o papel enxerga os elementos do outro terminal (por exemplo, "Carro" enxergaria como "rodas" os elementos da outra ponta do relacionamento *Carro-Roda*).

Estes papéis posteriormente serão mapeados como atributos simples ou de coleção, dependendo da cardinalidade. Caso não seja informado um papel, não será criado um atributo e conseqüentemente estar-se-á limitando a visibilidade da relação.

Os papéis podem ser alterados tanto através do menu associado ao *OORDBMMorphAssociacao* quanto diretamente no próprio elemento gráfico (bastando alterar o texto que representa a cardinalidade/papel).

7.4 Gerando um sistema

No modelador existe a ação nomeada *"Gerar projeto"* a partir do qual é exibida a tela (figura 7.10) de geração de projeto.

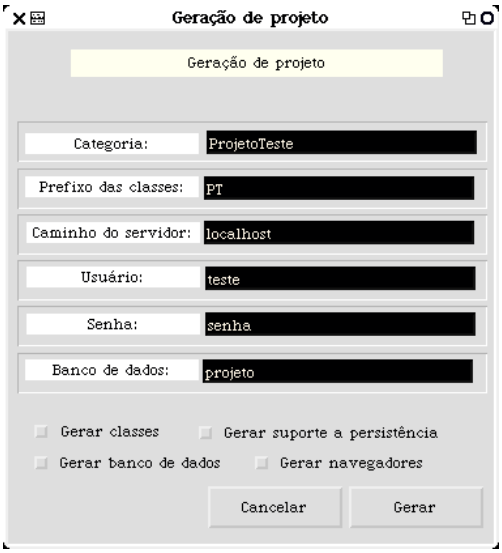

Figura 7.10: A tela de geração de sistema

Nesta é possível definir diversos parâmetros, sendo os dois primeiros relacionados a geração das classes: prefixo e categoria. *Prefixo* refere-se a um texto que será adicionado ao início do nome das classes para diferenciá-las de outras existentes já no ambiente, evitando choque de nomes. *Categoria* refere-se a categoria sob a qual as classes serão criadas dentro do ambiente do Smalltalk.

Os outros parâmetros referem-se à camada de persistência e geração do banco de dados para a persistência, como usuário e senha do banco de dados.

Pode-se escolher entre gerar as classes, a camada de persistência, o banco de dados e as interfaces de navegação/edição de forma independente.

7.5 Validações

Durante a modelagem, as validações realizadas sobre a mesma limitam-se às heranças, não permitindo a ligação entre classes que acabem gerando um ciclo no

grafo da hierarquia de classe ou a tentativa de estabelecer mais de uma herança para a mesma classe. Porém, ao requisitar a opção “Gerar projeto”, a ferramenta valida os seguintes aspectos da modelagem:

- Associações e heranças não conectadas.
- Nomes das classes - segundo a sintaxe do Smalltalk.
- Atributos - validando o nome segundo a sintaxe Smalltalk e o tipo conforme os tipos básicos que são disponibilizados ao usuário.
- Métodos - também segundo a sintaxe Smalltalk.
- Atributos e métodos na hierarquia de classes - evitando a duplicidade dos mesmos numa dada hieraquia (classe possui mesmos atributos/métodos que alguma de suas classes superiores).
- Relacionamentos - valida as cardinalidades dos mesmos e os papéis, tratando os papéis segundo a sintaxe para nomes de atributos do Smalltalk. A ferramenta também verifica casos onde o papel não foi informado e deve ser, para que o relacionamento possa concretizar-se (por exemplo, no caso de relacionamentos N:N, onde os dois lados do relacionamento devem ter um papel definido). Os relacionamentos em uma dada hierarquia também são validados tal qual métodos e atributos, evitando duplicidade dos mesmos. Para tal, utilizam-se os papéis para determinar se há duplicidade ou não dos relacionamentos.

7.6 Estrutura interna

Como base para a construção do Modelador de Sistemas foi utilizado o pacote Connectors[1], desenvolvido por Ned Konz. Este disponibiliza basicamente o suporte a entidades gráficas (“*Morphs*” no ambiente Squeak) capazes de conectarem-se umas às outras.

[1] http://minnow.cc.gatech.edu/squeak/1773

Ele provê diversos tipos de entidades, dentre as quais convém destacar *NCRectangleMorph* e o *NCConnectorMorph*. O primeiro provê uma entidade retangular a qual podem ser conectados vários *NCConnectorMorph*, em qualquer ponto de sua borda. O segundo é uma linha que conecta dois *Morphs*.

Além destes, foi utilizado também o *NCSmartLabel*, que acopla-se aos *NCConnectorMorph*, permitindo associar textos editáveis aos mesmos (utilizado para os papéis e cardinalidades das associações).

A partir do *NCTextRectangle* foi criada a classe *OORDBMMorphClasse*, que é a entidade gráfica correspondente ao elemento "Classe" no modelador. Ele foi expandido do formato original para o utilizado no modelador (com campos para nome da classe, atributos e métodos).

Além disso, foi estabelecida uma interface no mesmo para manipular o nome, atributos e métodos da classe que ele representa, além de rotinas para validar as informações nele contidas. Dessa forma, para determinar se uma definição de classe está correta, basta requisitar o método "validar".

A partir do *NCConnector* foi desenvolvida a *OORDBMorphHeranca*, que corresponde ao elemento Herança do modelador. Foi adicionada uma interface para recuperar a classe superior e inferior da herança.

Além disso também foram desenvolvidas rotinas que validam se o *OORDBMMorphHeranca* pode conectar-se a determinada classe durante a própria modelagem - dessa forma não é possível estabelecer uma herança inválida no diagrama. Por herança inválida entende-se aquelas que geram ciclos ou que estabeleceriam herança múltipla.

Foi desenvolvida também a classe *OORDBMMorphAnotacao*, que estende *NCNoteMorph* e representa as anotações no diagrama.

Outra classe criada foi *OORDBMMorphAssociacao*, que representa uma associação entre classes. Esta estende do *NCConnectorMorph* e implementa toda a parte relacionada a papéis e cardinalidade da relação. É possível definir e recuperar os papéis e cardinalidades de cada terminal da relação.

Por último temos *OORDBMMorphDiagrama*. Este corresponde ao am-

biente onde os elementos da modelagem ficam dispostos. Estende-se de um *NCWorld-Morph*, que é um *Morph* preparado para conter outros *Morphs*. Na extensão feita foram incluídas barras de rolagem.

Nele também foram desenvolvidos diversos métodos para validar a modelagem, assim como métodos para exportar a mesma para um XMIDocumento.

A validação do diagrama resume-se inicialmente a requistar a cada elemento contido no *OORDBMMorphDiagrama* que valide-se e analisar a resposta da validação para determinar se há algum problema. Em seguida as hierarquias de classes são validadas, evitando repetição de métodos e atributos (incluindo atributos originários das associações entre classes - evitar papéis de associação repetidos em uma hierarquia).

Capítulo 8

Sobre o trabalho

Neste trabalho, algumas dificuldades foram encontradas, tanto de ordem técnica quanto relacionadas ao projeto - quais recursos disponibilizar, linguagem e tecnologias a utilizar, etc.

Uma das dificuldades iniciais foi limitar o escopo do projeto. Considerando as etapas comuns no desenvolvimento de sistemas OO, chegou-se à conclusão que tratar a modelagem do sistema (seguindo o conceito de "Modelo de Objetos") seria uma boa alternativa. Esta englobaria tanto elementos relacionados à análise do sistema (determinação das entidades componentes do mesmo) quanto aspectos relacionados à implementação (as classes a serem criadas).

A escolha da linguagem na qual o sistema seria desenvolvido recaiu sobre o Smalltalk, devido às caraterísticas da linguagem que tornaram seu uso atraente para o projeto (citadas na seção 5.2.1). Mas isto não significa que outra linguagem (como Java) não pudesse ter sido utilizada (com as devidas adaptações).

Sobre a persistência dos dados, inicialmente cogitou-se a utilização do mapeamento OO-relacional segundo a técnica de uma tabela por classe concreta. Esta abordagem, apesar de simples e de fácil compreensão (para quem analisa o banco de dados), apresentou um problema nos relacionamentos entre classes: dado um relacionamento entre classes com subclasses, como saber se o mesmo é com a classe ou alguma de suas subclasses? A ferramenta utilizada para persistência (GLORP) não disponibiliza

nenhum recurso para este caso, logo esta abordagem acabou descartada.

Analisou-se então a técnica de uma tabela por classe do sistema. Esta alternativa esbarrou em um problema de implementação na ferramenta de persistência, que não persistiu corretamente os dados de objetos divididos em várias tabelas. Entrou-se em contato com os desenvolvedores, porém segundo estes o suporte a este tipo de mapeamento, apesar de possível, ainda não era considerado suportado e uma versão com suporte não seria desenvolvida logo.

Por fim adotou-se a abordagem de uma tabela por hierarquia. Para compensar o fato de não ter mais uma tabela por classe concreta (como na primeira abordagem), que torna a visualização do banco de dados mais simples, foi desenvolvido um suporte para criação de visões no banco de dados. Estas representam as classes do sistema. Desse forma, chega-se a algo semelhante à primeira abordagem de mapeamento adotada em termos de facilidade para visualizar o banco de dados.

Um aspecto a comentar sobre a forma de mapeamento adotada refere-se à filtragem dos registros. Para gerar os identificadores utilizados na filtragem dos dados das tabelas (recuperando apenas elementos da classe desejada), utiliza-se o nome das próprias classes como identificador. Dessa forma, mesmo que altere-se a modelagem e regere-se o sistema, os dados persistidos para a versão anterior continuarão válidos.

Outro aspecto sobre o mapeamento OO-relacional refere-se às chaves estrangeiras. Apesar de o código estar desenvolvido, atualmente as chaves estrangeiras não estão sendo geradas. Isto deve-se ao sistema de persistência adotado (GLORP) não conseguir tratar referências cíclicas durante a inserção de dados. Isto é, imagine a situação onde temos duas classes, Médico e Clínica:

- Medico(nome,crm,idade,clinicaOndeTrabalha (FK[1] para Clinica)[2])
- Clinica(nome,localizacao,medicoResponsalvel (FK para Medico))

Suponha-se agora que se insira um médico e uma clínica, e associem-se os dois cliclicamente (o médico trabalhar na clínica e é também o médico responsável pela

[1]Foreign Key - chave estrangeira

[2]Considera-se aqui que um médico só pode trabalhar para uma determinada clínica no domínio do exemplo

mesma). Dessa forma, ao gravar os dados no banco de dados, o sistema de persistência encontrará o seguinte problema devido as chaves estrangeiras: não pode gravar o médico porque a Clínica ainda não está no banco e não pode gravar a Clínica pelo médico ainda não constar no banco de dados.

Para solucionar isso, ele deveria fazer a gravação de forma a contornar a referência cíclica (deixando os atributos com chave estrangeira para serem gravados posteriormente em uma atualização de registro, por exemplo). Como o GLORP não trata este caso, desativou-se a geração das chaves estrangeiras, o que não prejudica o uso da ferramenta ou visualização dos dados no banco de dados, pois trata-se de um aspecto interno ao mesmo.

A utilização da persistência foi um aspecto que tentou-se simplificar. Através da classe de gerência de persistência[3] gerada pela ferramenta, consegue-se o uso da persistência com poucos e simples passos: inicia-se uma sessão, manipulam-se os objetos e termina-se a sessão. Não ficou ao encargo do usuário cuidar de aspectos como iniciar e terminar a conexão com o banco de dados, etc.

A falta de um formato único para troca de modelagens UML entre ferramentas também foi um fato que verificou-se no início do projeto. Como não existia a ferramenta de modelagem, foi necessário encontrar uma forma de importar modelagens de hierarquias de classes de outras ferramentas para testar o *Gerador de Sistemas*. A melhor alternativa que chegou-se foi o XMI. Porém este não adota um modelo XML único, logo necessita de um parser específico para cada modelo que as ferramentas podem gerar (ArgoUML, Together Control Center, etc.). Dessa forma, limitou-se a importação às modelagens feitas pelo Together Control Center.

Outra fato a comentar foi a documentação de alguns componentes utilizados, em especial o GLORP. A documentação em geral estava desatualizada ou era inexistente. No caso do GLORP, ela resumia-se a aspectos básicos de uso e não entrava em detalhes sobre o funcionamento interno. Dessa forma, detectar alguns problemas e tentar buscar soluções acabou tornando-se trabalhoso, pois houve a necessidade de aplicar-se

[3]Descrita na seção 6.8

procedimentos de “engenharia reversa[4]” no código do mesmo (na maior parte sem qualquer comentário a respeito de seu funcionamento). Ficou clara a importância de uma boa documentação para a manutenção de um sistema.

Sobre a ferramenta de modelagem não houve grandes problemas em seu desenvolvimento. O que buscou-se na mesma foi ser o mais simples possível de usar. Dessa forma, a edição no próprio espaço de definição dos atributos e métodos evitou a necessidade do usuário de abrir uma tela adicional para alterar os atributos e métodos de uma classe, assim como os papéis e cardinalidades de um relacionamento.

O uso dos papéis nos terminais da relação teve como objetivo tornar aquilo que foi modelado mais próximo do código gerado, já que os mesmos tornam-se atributos nas respectivas classes. Dessa forma, dada uma modelagem e o código gerado fica clara a origem dos atributos que não faziam parte da definição da classe na modelagem.

A tela para geração do sistema foi um aspecto que visou simplificar o processo, centralizando-o. Dessa forma não são necessárias ferramentas ou sistemas adicionais. A tela também buscou ser simples, tendo apenas os aspectos mínimos necessários para a geração do projeto.

Sobre a estrutura da implementação, além do que já foi comentado em capítulos anteriores, cabe salientar que a mesma buscou ser o mais flexível possível. Dessa forma é possível estendê-la para utilizar outro banco de dados (ou mesmo outro sistema de persistência) caso haja interesse, sem a necessidade de alterar o código das classes existentes.

Um último ponto a comentar sobre sobre o sistema desenvolvido é o fato que ele não necessita de nada além do máquina virtual do Squeak e do banco de dados para executar. Não é necessário instalar bibliotecas ou programas adicionais.

[4]Obter uma representação de nível mais alto que outra existente de um determinado sistema - como um diagrama de classes a partir do código de uma aplicação

Capítulo 9

Conclusão e trabalhos futuros

Neste projeto desenvolveu-se uma ferramenta para modelagem de sistemas computacionais segundo o paradigma de programação orientado a objetos. Ela possui recursos para gerar a estrutura básica de um sistema computacional em Smalltalk (Squeak), assim como um esquema de banco de dados relacional, sobre o qual os dados são persistidos através de um framework.

Este porém é o "ponto de partida" de uma ferramenta mais complexa, que visa tratar todo o processo de desenvolvimento de software. Este "processo" poderia incluir etapas comuns à maioria dos processos tradicionais existentes ou fixar-se em um processo específico (como UP). Completa, a ferramenta serviria como apoio para o ensino de desenvolvimento de sistemas de uma forma completa. É claro que ela cobriria apenas os aspectos básicos de cada etapa, já que destina-se a projetos simples, desenvolvidos por pessoas sem ou com pouco conhecimento sobre o assunto.

Esta idéia de apresentar todo um processo de desenvolvimento de software a alguém iniciante no assunto (por exemplo, na primeira fase de um curso de Ciências da Computação) é uma abordagem diferente da aplicada na grande maioria dos cursos. Nestes comumente inicia-se com o ensino de aspectos isolados (algorítimos, programação orientada a objetos, estruturas de dados) e adiante trabalha-se aspectos como engenharia de software, considerando sistemas computacionais mais próximos dos reais.

A idéia sugerida aqui (baseada nos artigos de [Meyer 1993] e [J. La

Salle 1997], segue o princípio do “currículo invertido”, onde logo de início apresenta-se ao aluno “o todo” (no caso de Ciências da Computação, um sistema computacional completo) e começa-se a desmontar este “todo”. Dessa forma, à medida que é decomposto em partes menores, os conceitos e técnicas envolvidas começam a ficar mais claros, pois o aluno já teve uma visão do todo e pode perceber onde eles aplicam-se.

Sobre a ferramente desenvolvida, conseguiu-se alcançar o objetivo inicial (uma ferramenta simples de modelagem com suporte a persistência de dados). Porém isto não significa que a mesma está completa e sem possiblidades de expansões. Pelo contrário, diversas sugestões podem ser feitas para melhorar tanto seu funcionamento quanto sua utilização.

Como primeira sugestão, sugere-se uma análise pegagógica e ergonômica da ferramenta. Esta análise (que envolveria testes com alguns alunos) permitiria determinar se as decisões tomadas e a estrutura da ferramenta estão adequadas ao objetivo da mesma.

Uma outra sugestão, que vai de encontro ao que foi comentado anteriomente, é desenvolver o suporte a outras etapas do desenvolvimento de sistemas, como casos de usos, etc. Sobre isto, apenas recomenda-se que as mesmas não devem buscar possuir muitos recursos além do mínimo necessário para construção de sistemas simples. Para sistemas complexos existem ferramentas de alta qualidade já disponíveis.

Uma alternativa, porém, seria implementar recursos mais avançados (para poder ser utilizada para projetos mais avançados) mas de forma que estes estejam disponíveis conforme configuração. Dessa forma, poderia-se trabalhar com a ferramenta em diversos “níveis” de aprendizado, conforme o momento em que a mesma fosse aplicada.

Ainda sobre a parte de interface, poderia-se desenvolver um sistema simples para a criação de telas - nada muito avançado, apenas com elementos simples. Para tal, pode-se pensar em utilizar o pacote “Prefab”, disponível para o Squeak, que permite construir telas através do posicionamento dos componentes diretamente, sem nenhuma codificação.

Em relação ao componente de modelagem desenvolvido, alguns aspec-

tos podem ser melhorados. Um deles é a possibilidade de incluir classes já existentes no Smalltalk na modelagem (atualmente não pode-se referenciar uma classe que não esteja presente no diagrama). Isto poderia ser feito para qualquer classe presente no ambiente do Squeak, exibindo por exemplo, uma listagem das mesmas e permitindo ao usuário arrastá-las para o diagrama. Outra sugestão é uma forma para dispor as classes e relacionamentos de forma automática (pode-se pensar em utilizar alguns dos algoritimos existentes para organização de grafos, por exemplo).

Sobre a ferramenta como um todo, uma sugestão é implementar relacionamentos de agregação e composição. O controle poderia ser feito pela ferramenta de persistência (GLORP), através de uma derivação dos tipos de mapeamentos existentes atualmente. Outra alternativa seria manter o controle sobre as dependências de existência entre as classes à parte e verificando as mesmas ao receber um pedido de exclusão de um item (isto poderia ser feito no próprio gerenciador de persistência).

Sugere-se também implementar aspectos didáticos à ferramenta, como textos explicativos sobre os elementos do Modelador de Sistemas (Classe, Associação, etc.), demonstrações de como executar determinar ações - como mapear os elementos da descrição de um sistema no mundo para para elementos de um sistema computacional - , etc. Para isto, seria interessante, por exemplo, acompanhar a utilização da ferramenta por um grupo reduzido de usuários para determinar aonde surgem as dúvidas e quais explicações/demonstrações tendem a sanar de forma mais eficiente as mesmas.

Poderia-se também desenvolver um “tutorial”, através do qual o usuário utilizasse a ferramenta em uma seqüência de passos pré-definida, que envolveria todo o processo de desenvolvimento de um sofware. Assim ele teria um “guia” para a utilização da ferramenta.

Como última sugestão fica a tradução das interfaces (métodos) das classes básicas do Smalltalk (como coleções, “Number”, “String”, etc.). Assim, ficaria ainda mais simples para o usuário desenvolver os sistemas, já que tornaria-se mais natural a programação dos métodos.

Para tal poderia-se desenvolver uma ferramenta que através de reflexão determine os métodos de uma dada classe e exiba-os em uma tela onde seja possível

traduzí-los de forma que gere-se uma subclasse para a classe traduzida, contendo a interface traduzida.

Referências Bibliográficas

[About Squeak 2004]ABOUT Squeak. 2004. Disponível em: <http://www.squeak.org/about/index.html>. Acesso em: 28/09/2004.

[Ambler 2003]AMBLER, S. W. The object-relational impedance mismatch. 2003. Disponível em: <http://www.agiledata.org/essays/impedanceMismatch.html>. Acesso em: 20/05/2004.

[Date 1986]DATE, C. J. *Introdução a Sistemas de Bancos de Dados*. 4ª. ed. [S.l.]: Editora Campus, 1986.

[Date 1995]DATE, C. J. *An Introduction to Database Systems*. 6ª. ed. [S.l.]: Addison-Wesley Publishing Company, 1995. (Addison-Wesley systems programming series).

[J. La Salle 1997]J. La Salle, A. An inverted computing curriculum: Preparing graduates to build quality systems. 1997.

[Larman 2002]Larman, C. *Applying UML and patterns: an introduction to object-oriented analysis and design and the Unified Process*. 2ª. ed. [S.l.]: Prentice Hall, 2002.

[Lount 2004]LOUNT, P. W. A brief introduction to smalltalk. 2004. Disponível em: <http://www.smalltalk.org/articles/article_20040000_11.html>. Acesso em: 28/09/2004.

[Meyer 1993]MEYER, B. Towards an object oriented curriculum. 1993.

[Nygaard e Dahl]NYGAARD, K.; Dahl, O.-J. How object-oriented programming started. Disponível em:

<http://heim.ifi.uio.no/∼kristen/FORSKNINGSDOK_MAPPE/F_OO_start.html>. Acesso em: 20/05/2004.

[Pinson 1988]PINSON, L. *An Introduction to Object-Oriented Programming and Smalltalk*. [S.l.]: Addison-Wesley Publishing Company, 1988.

[Rumbaugh 1991]RUMBAUGH, J. *Object-Oriented Modeling and Design*. [S.l.]: Prentice-Hall International, Inc., 1991.

[Silberschatz, K. Korth e Sudarshan 1999]SILBERSCHATZ, A.; K. Korth, H.; Sudarshan, S. *Sistema de Banco de Dados*. 3ª. ed. [S.l.]: Makron Books, 1999.

[Stoimenov 1998]STOIMENOV, L. Bridging objects and relations: a mediator for an oo front-end to rdbmss. Elsevier Science, 1998.

[W. Ambler 2000]W. Ambler, S. Mapping objects to relational databases. 2000.

[W. Ambler 2003]W. Ambler, S. *Agile Database Techniques : Effective Strategies for the Agile Software Developer*. [S.l.]: Wiley, 2003.

[W. Ambler 2003]W. Ambler, S. The fundamentals of mapping objects to relational databases. 2003. Disponível em: <http://www.agiledata.org/essays/mappingObjects.html>. Acesso em: 20/05/2004.

[W. Rahayu et al. 2000]W. Rahayu, J. et al. A methodology for transforming inheritance relationships in an object-oriented conceptual model to relational tables. Elsevier Science, 2000.

[What is Extreme Programming? 2004]WHAT is Extreme Programming? 2004. Disponível em: <http://www.extremeprogramming.org/what.html>. Acesso em: 02/10/2004.

Capítulo 10

Anexos

10.1 Anexo I - Fonte do sistema exemplo gerado pela ferramenta

Object subclass: CMConsulta
Category: ClinicaMedica
Instance variable names: hora valor data intid paciente medico

Instance protocol

Methods for category: atributos

data
↑ data.

data: *umData*
[[:valorTMPSET | valorTMPSET asDate] value: umData] ifError: [self error: 'Valor passado incompatível!'].
data ← [:valorTMPSET | valorTMPSET asDate] value: umData

hora
↑ hora.

hora: *umHora*
[[:valorTMPSET | valorTMPSET asTime] value: umHora] ifError: [self error: 'Valor passado incompatível!'].
hora ← [:valorTMPSET | valorTMPSET asTime] value: umHora

intid
↑ intid.

intid: *umIntid*
[[:valorTMPSET | valorTMPSET asInteger] value: umIntid] ifError: [self error: 'Valor passado incompatível!'].
intid ← [:valorTMPSET | valorTMPSET asInteger] value: umIntid

medico
↑ medico.

medico: *umMedico*
(umMedico isKindOf: CMMedico) ifFalse: [↑ self error: 'Valor nao e do tipo esperado'] ifTrue: [
medico ← umMedico
]

paciente
↑ paciente.

paciente: *umPaciente*
(umPaciente isKindOf: CMPaciente) ifFalse: [↑ self error: 'Valor nao e do tipo esperado'] ifTrue: [
paciente ← umPaciente
]

valor
↑ valor.

valor: *umValor*
[[:valorTMPSET | valorTMPSET asScaledDecimal: 2] value: umValor] ifError: [self error: 'Valor passado incompatível!'].
valor ← [:valorTMPSET | valorTMPSET asScaledDecimal: 2] value: umValor

DescriptorSystem subclass: CMDescritorPersistencia
Category: ClinicaMedica

Instance protocol

Methods for category: descritores

allTableNames
↑ #('CMConsulta' 'CMPessoa').

constructAllClasses
↑ ((super constructAllClasses) add: CMConsulta; add: CMMedico; add: CMPessoa; add: CMPaciente; yourself).

descriptorForCMConsulta: *umDescritor* | *tabela* |
umDescritor table: (self tableNamed: 'CMCONSULTA').
umDescritor addMapping: (DirectMapping from: #hora to: ((self tableNamed: 'CMConsulta') fieldNamed: 'hora')).
umDescritor addMapping: (DirectMapping from: #valor to: ((self tableNamed: 'CMConsulta') fieldNamed: 'valor')).
umDescritor addMapping: (DirectMapping from: #data to: ((self tableNamed: 'CMConsulta') fieldNamed: 'data')).
umDescritor addMapping: (DirectMapping from: #intid to: ((self tableNamed: 'CMConsulta') fieldNamed: 'intid')).
(self typeResolverFor: CMConsulta) register: umDescritor keyedBy: 'CMConsulta' field: ((self tableNamed: 'CMCONSULTA') fieldNamed: 'tpCtrlInt').
umDescritor addMapping: ((OneToOneMapping new) attributeName: #paciente; referenceClass: CMPaciente; mappingCriteria: (Join from: ((self tableNamed: 'CMCONSULTA') fieldNamed: 'paciente') to: ((self tableNamed: 'CMPESSOA') fieldNamed: 'intid'))).
umDescritor addMapping: ((OneToOneMapping new) attributeName: #medico; referenceClass: CMMedico; mappingCriteria: (Join from: ((self tableNamed: 'CMCONSULTA') fieldNamed: 'medico') to: ((self tableNamed: 'CMPESSOA') fieldNamed: 'intid'))).

descriptorForCMMedico: *umDescritor* | *tabela* |
umDescritor table: (self tableNamed: 'CMPESSOA').
umDescritor addMapping: (DirectMapping from: #dataNascimento to: ((self tableNamed: 'CMPessoa') fieldNamed: 'dataNascimento')).
umDescritor addMapping: (DirectMapping from: #nome to: ((self tableNamed: 'CMPessoa') fieldNamed: 'nome')).
umDescritor addMapping: (DirectMapping from: #cpf to: ((self tableNamed: 'CMPessoa') fieldNamed: 'cpf')).
umDescritor addMapping: (DirectMapping from: #crm to: ((self tableNamed: 'CMPessoa') fieldNamed: 'crm')).
umDescritor addMapping: (DirectMapping from: #intid to: ((self tableNamed: 'CMPessoa') fieldNamed: 'intid')).
(self typeResolverFor: CMPessoa) register: umDescritor keyedBy: 'CMMedico' field: ((self tableNamed: 'CMPESSOA') fieldNamed: 'tpCtrlInt').
umDescritor addMapping: ((OneToManyMapping new) attributeName: #consultasFeitas; referenceClass: CMConsulta; mappingCriteria: (Join from: ((self tableNamed: 'CMPESSOA') fieldNamed: 'intid') to: ((self tableNamed: 'CMCONSULTA') fieldNamed: 'medico'))).

descriptorForCMPaciente: *umDescritor* | *tabela* |
umDescritor table: (self tableNamed: 'CMPESSOA').
umDescritor addMapping: (DirectMapping from: #dataNascimento to: ((self tableNamed: 'CMPessoa') fieldNamed: 'dataNascimento')).
umDescritor addMapping: (DirectMapping from: #nome to: ((self tableNamed: 'CMPessoa') fieldNamed: 'nome')).
umDescritor addMapping: (DirectMapping from: #cpf to: ((self tableNamed: 'CMPessoa') fieldNamed: 'cpf')).
umDescritor addMapping: (DirectMapping from: #codigoDeCadastro to: ((self tableNamed: 'CMPessoa') fieldNamed: 'codigoDeCadastro')).
umDescritor addMapping: (DirectMapping from: #intid to: ((self tableNamed: 'CMPessoa') fieldNamed: 'intid')).
(self typeResolverFor: CMPessoa) register: umDescritor keyedBy: 'CMPaciente' field: ((self tableNamed: 'CMPESSOA') fieldNamed: 'tpCtrlInt').
umDescritor addMapping: ((OneToManyMapping new) attributeName: #consultasRealizadas; referenceClass: CMConsulta; mappingCriteria: (Join from: ((self tableNamed: 'CMPESSOA') fieldNamed: 'intid') to: ((self tableNamed: 'CMCONSULTA') fieldNamed: 'paciente'))).

descriptorForCMPessoa: *umDescritor* | *tabela* |
umDescritor table: (self tableNamed: 'CMPESSOA').
umDescritor addMapping: (DirectMapping from: #dataNascimento to: ((self tableNamed: 'CMPessoa') fieldNamed: 'dataNascimento')).
umDescritor addMapping: (DirectMapping from: #nome to: ((self tableNamed: 'CMPessoa') fieldNamed: 'nome')).
umDescritor addMapping: (DirectMapping from: #cpf to: ((self tableNamed: 'CMPessoa') fieldNamed: 'cpf')).
umDescritor addMapping: (DirectMapping from: #intid to: ((self tableNamed: 'CMPessoa') fieldNamed: 'intid')).
(self typeResolverFor: CMPessoa) register: umDescritor keyedBy: 'CMPessoa' field: ((self tableNamed: 'CMPESSOA') fieldNamed: 'tpCtrlInt').

Methods for category: tabelas

tableForCMCONSULTA: *umDescritor* | *hora* **intid** *tpctrlint* **medico** *valor* **paciente** *data* |
hora ← (umDescritor createFieldNamed: 'hora' type: (platform time)) .
intid ← (umDescritor createFieldNamed: 'intid' type: (platform serial)) bePrimaryKey.
tpctrlint ← (umDescritor createFieldNamed: 'tpCtrlInt' type: (platform varchar)) .

```
medico ← (umDescritor createFieldNamed: 'medico' type: (platform integer)) .
valor ← (umDescritor createFieldNamed: 'valor' type: (platform numeric)) .
paciente ← (umDescritor createFieldNamed: 'paciente' type: (platform integer)) .
data ← (umDescritor createFieldNamed: 'data' type: (platform date)) .
umDescritor addForeignKeyFrom: medico to: ((self tableNamed: 'CMPESSOA') fieldNamed: 'intid' ).
umDescritor addForeignKeyFrom: paciente to: ((self tableNamed: 'CMPESSOA') fieldNamed: 'intid' ).
```

tableForCMPESSOA: *umDescritor* | *datanascimento* **nome** *intid* **tpctrlint** *codigodecadastro* **cpf** *crm* |

```
datanascimento ← (umDescritor createFieldNamed: 'dataNascimento' type: (platform date)) .
nome ← (umDescritor createFieldNamed: 'nome' type: (platform varchar)) .
intid ← (umDescritor createFieldNamed: 'intid' type: (platform serial)) bePrimaryKey.
tpctrlint ← (umDescritor createFieldNamed: 'tpCtrlInt' type: (platform varchar)) .
codigodecadastro ← (umDescritor createFieldNamed: 'codigoDeCadastro' type: (platform integer)) .
cpf ← (umDescritor createFieldNamed: 'cpf' type: (platform varchar)) .
crm ← (umDescritor createFieldNamed: 'crm' type: (platform integer)) .
```

Methods for category: typeResolvers

typeResolverForCMConsulta

```
↑ FilteredTypeResolver forRootClass: CMConsulta.
```

typeResolverForCMMedico

```
↑ FilteredTypeResolver forRootClass: CMPessoa.
```

typeResolverForCMPaciente

```
↑ FilteredTypeResolver forRootClass: CMPessoa.
```

typeResolverForCMPessoa

```
↑ FilteredTypeResolver forRootClass: CMPessoa.
```

Object subclass: CMGerenciadorPesistencia
Category: ClinicaMedica
Instance variable names: accessor sessao descritor usuarioBD senhaBD enderecoBD nomeBD

Class protocol

Methods for category: initialization

new
↑ super new initialize.

Instance protocol

Methods for category: initialization

initialize *usuarioBD ← 'tcc'.*
senhaBD ← 'r12g1982'.
enderecoBD ← 'localhost'.
nomeBD ← 'tcc'.

Methods for category: registro

excluir: *umObjeto*
sessao isNil ifTrue: [
self error: 'Voce nao esta em uma sessao com o banco de dados'.
].
sessao delete: umObjeto.

recuperarEntidadeDoTipo: *umaClasse* **segundo:** *umaCondicao*
sessao isNil ifTrue: [
self error: 'Voce nao esta em uma sessao com o banco de dados'.
].
↑ sessao readOneOf: umaClasse where: umaCondicao.

recuperarEntidadesDoTipo: *umaClasse*
sessao isNil ifTrue: [
self error: 'Voce nao esta em uma sessao com o banco de dados'.
].
↑ sessao readManyOf: umaClasse.

recuperarEntidadesDoTipo: *umaClasse* **segundo:** *umaCondicao*
sessao isNil ifTrue: [
self error: 'Voce nao esta em uma sessao com o banco de dados'.
].
↑ sessao readManyOf: umaClasse where: umaCondicao.

registrar: *umObjeto*
sessao isNil ifTrue: [
self error: 'Voce nao esta em uma sessao com o banco de dados'.
].
sessao register: umObjeto.
sessao refresh: umObjeto

registrarNovo: *umObjeto*
sessao isNil ifTrue: [
self error: 'Voce nao esta em uma sessao com o banco de dados'.
].
sessao registerAsNew: umObjeto.
sessao refresh: umObjeto

Methods for category: sessao

conectarBancoDeDados
accessor ← DatabaseAccessor forLogin: (Login new database: PostgreSQLPlatform new; username: usuarioBD; password: senhaBD; connectString: enderecoBD , ':' , '5432' , '←' , nomeBD).
accessor login.

gravarEContinuarSessao
sessao isNil ifTrue: [
self error: 'Voce nao esta em uma sessao com o banco de dados'.
].
sessao commitUnitOfWorkAndContinue.

iniciarSessao
self conectarBancoDeDados.
descritor ← CMDescritorPersistencia new.

```
  sessao ← GlorpSession new.
  sessao system: descritor.
  sessao accessor: accessor.
  sessao beginUnitOfWork.
terminarSessao
  sessao isNil ifTrue: [
     self error: 'Voce nao esta em uma sessao com o banco de dados'.
  ].
  sessao commitUnitOfWork.
  accessor logout.
  sessao ← nil.
  accessor ← nil.
```

PrefabManager subclass: CMNavegadorDeCMConsulta
Category: ClinicaMedica
Instance variable names: sessao elementos inserindo corpo

Instance protocol

Methods for category: private

atualizarDadosDoRegistro

```
elementos elementAtCursor isNil ifFalse: [
elementos elementAtCursor hora: (corpo clientWithTag: 'hora') text
elementos elementAtCursor valor: (corpo clientWithTag: 'valor') text
elementos elementAtCursor data: (corpo clientWithTag: 'data') text
]
```

atualizarDadosNaTela

```
elementos elementAtCursor isNil ifFalse: [
(elementos elementAtCursor hora) isNil ifTrue: [ (corpo clientWithTag: 'hora') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'hora') text: (elementos elementAtCursor hora asString).].
(elementos elementAtCursor valor) isNil ifTrue: [ (corpo clientWithTag: 'valor') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'valor') text: (elementos elementAtCursor valor asString).].
(elementos elementAtCursor data) isNil ifTrue: [ (corpo clientWithTag: 'data') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'data') text: (elementos elementAtCursor data asString).].
] ifTrue: [
(corpo clientWithTag: 'hora') text: ' '.
(corpo clientWithTag: 'valor') text: ' '.
(corpo clientWithTag: 'data') text: ' '.
]
```

buildOn: *aManager*

```
aManager
    extent: 579@150;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    minSize: 10@10;
    maxSize: 1073741823@1073741823.
  aManager addClient: (PrefabFrame new
    extent: 536@35;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    style: #inset;
    yourself)
    atRelPosition: 13@10.
aManager addClient: (PrefabButton new
    extent: 64@28;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    action: [self irParaPrimeiroRegistro];
    tag: #botaoPrimeiro;
    text: ' Primeiro ';
    contentOrientation: #centered;
    yourself)
    atRelPosition: 19@13.
  aManager addClient: (PrefabButton new
    extent: 79@27;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    action: [self cancelarAlteracoes];
    tag: #botaoCancelar;
    text: 'Cancelar alt.';
    contentOrientation: #centered;
    yourself)
    atRelPosition: 462@14.
  aManager addClient: (PrefabButton new
    extent: 56@28;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
```

```
        action: [self irParaProximoRegistro];
        tag: #botaoProximo;
        text: 'Proximo';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 147@13.
    aManager addClient: (PrefabButton new
        extent: 43@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self irParaUltimoRegistro];
        tag: #botaoUltimo;
        text: 'Ultimo';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 204@13.
    aManager addClient: (PrefabButton new
        extent: 68@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self gravarRegistro];
        tag: #botaoSalvar;
        text: 'Salvar alt.';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 394@14.
    aManager addClient: (PrefabButton new
        extent: 62@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self irParaRegistroAnterior];
        tag: #botaoAnterior;
        text: 'Anterior';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 84@13.
    aManager addClient: (PrefabButton new
        extent: 46@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self excluirRegistro];
        tag: #botaoExcluir;
        text: 'Excluir';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 327@13.
    aManager addClient: (PrefabButton new
        extent: 54@26;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self inserirRegistro];
        tag: #botaoInserir;
        text: 'Inserir';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 273@13.
aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'hora';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@60.
    aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
```

```
        tag: 'hora';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@60.
    aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'valor';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@90.
      aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'valor';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@90.
    aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'data';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@120.
      aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'data';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@120.
```

cancelarAlteracoes

```
inserindo ifTrue: [elementos removeLast. elementos adjustCursor. self atualizarDadosNaTela] ifFalse: [
    self atualizarDadosNaTela.]
```

excluirRegistro

```
inserindo ifTrue: [elementos removeLast. elementos cursorToLast. self atualizarDadosNaTela. inserindo ← false.] ifFalse: [
    elementos elementAtCursor isNil ifFalse: [sessao excluir: (elementos elementAtCursor). sessao gravarEContinuarSessao. elementos remove: (elementos elementAtCursor). elementos adjustCursor. self atualizarDadosNaTela.] ]
```

gravarRegistro

```
inserindo ifTrue: [sessao registrarNovo: (elementos elementAtCursor). self atualizarDadosDoRegistro. sessao gravarEContinuarSessao. inserindo ← false.] ifFalse: [
    self atualizarDadosDoRegistro. sessao gravarEContinuarSessao.]
```

iniciarSessao

```
sessao ← CMGerenciadorPesistencia new. sessao iniciarSessao. elementos ← OrderedCollectionWithCursor withAll: (sessao recuperarEntidadesDoTipo: CMConsulta ). self atualizarDadosNaTela.
```

initialize

```
super initialize. self buildOn: self. corpo ← self. inserindo ← false.
```

inserirRegistro

```
elementos addLast: (CMConsulta new). elementos cursorToLast. self atualizarDadosNaTela. inserindo ← true.
```

irParaPrimeiroRegistro

```
elementos cursorToFirst.
self atualizarDadosNaTela.
```

irParaProximoRegistro

```
elementos advanceCursor.
self atualizarDadosNaTela.
```

irParaRegistroAnterior

```
elementos retrocedeCursor.
```

```
    self atualizarDadosNaTela.
irParaUltimoRegistro
    elementos cursorToLast.
    self atualizarDadosNaTela.
openInSystemWindow
    | nm |
    self world
        ifNotNil: [self delete].
    (nm ← Prefab managerFrameClass new) client: self;
        setLabel: title;
        openInWorld.
    nm
        color: (Color
                r: 0.851
                g: 0.851
                b: 0.851).
    nm setLabel: 'Navegador de entidades CMConsulta'.
    self iniciarSessao.   ↑ self
terminarSessao
    sessao terminarSessao. elementos ← OrderedCollectionWithCursor new.
```

Methods for category: system

```
abrir
    self openInSystemWindow.
```

PrefabManager subclass: CMNavegadorDeCMMedico
Category: ClinicaMedica
Instance variable names: sessao elementos inserindo corpo

Instance protocol

Methods for category: private

atualizarDadosDoRegistro

```
elementos elementAtCursor isNil ifFalse: [
elementos elementAtCursor nome: (corpo clientWithTag: 'nome') text
elementos elementAtCursor cpf: (corpo clientWithTag: 'cpf') text
elementos elementAtCursor dataNascimento: (corpo clientWithTag: 'dataNascimento') text
elementos elementAtCursor crm: (corpo clientWithTag: 'crm') text
]
```

atualizarDadosNaTela

```
elementos elementAtCursor isNil ifFalse: [
(elementos elementAtCursor nome) isNil ifTrue: [ (corpo clientWithTag: 'nome') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'nome') text: (elementos elementAtCursor nome asString).].
(elementos elementAtCursor cpf) isNil ifTrue: [ (corpo clientWithTag: 'cpf') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'cpf') text: (elementos elementAtCursor cpf asString).].
(elementos elementAtCursor dataNascimento) isNil ifTrue: [ (corpo clientWithTag: 'dataNascimento') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'dataNascimento') text: (elementos elementAtCursor dataNascimento asString).].
(elementos elementAtCursor crm) isNil ifTrue: [ (corpo clientWithTag: 'crm') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'crm') text: (elementos elementAtCursor crm asString).].
] ifTrue: [
(corpo clientWithTag: 'nome') text: ' '.
(corpo clientWithTag: 'cpf') text: ' '.
(corpo clientWithTag: 'dataNascimento') text: ' '.
(corpo clientWithTag: 'crm') text: ' '.
]
```

buildOn: *aManager*

```
aManager
    extent: 579@180;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    minSize: 10@10;
    maxSize: 1073741823@1073741823.
  aManager addClient: (PrefabFrame new
    extent: 536@35;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    style: #inset;
    yourself)
    atRelPosition: 13@10.
aManager addClient: (PrefabButton new
    extent: 64@28;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    action: [self irParaPrimeiroRegistro];
    tag: #botaoPrimeiro;
    text: ' Primeiro ';
    contentOrientation: #centered;
    yourself)
    atRelPosition: 19@13.
  aManager addClient: (PrefabButton new
    extent: 79@27;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    action: [self cancelarAlteracoes];
    tag: #botaoCancelar;
    text: 'Cancelar alt.';
    contentOrientation: #centered;
    yourself)
    atRelPosition: 462@14.
```

```
    aManager addClient: (PrefabButton new
        extent: 56@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self irParaProximoRegistro];
        tag: #botaoProximo;
        text: 'Proximo';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 147@13.
    aManager addClient: (PrefabButton new
        extent: 43@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self irParaUltimoRegistro];
        tag: #botaoUltimo;
        text: 'Ultimo';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 204@13.
    aManager addClient: (PrefabButton new
        extent: 68@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self gravarRegistro];
        tag: #botaoSalvar;
        text: 'Salvar alt.';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 394@14.
    aManager addClient: (PrefabButton new
        extent: 62@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self irParaRegistroAnterior];
        tag: #botaoAnterior;
        text: 'Anterior';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 84@13.
    aManager addClient: (PrefabButton new
        extent: 46@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self excluirRegistro];
        tag: #botaoExcluir;
        text: 'Excluir';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 327@13.
    aManager addClient: (PrefabButton new
        extent: 54@26;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self inserirRegistro];
        tag: #botaoInserir;
        text: 'Inserir';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 273@13.
aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'nome';
        contentOrientation: #centered;
        yourself)
```

```
        atRelPosition: 10@60.
    aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'nome';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@60.
  aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'cpf';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@90.
    aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'cpf';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@90.
  aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'dataNascimento';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@120.
    aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'dataNascimento';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@120.
  aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'crm';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@150.
    aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'crm';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@150.
```

cancelarAlteracoes

```
  inserindo ifTrue: [elementos removeLast. elementos adjustCursor. self atualizarDadosNaTela] ifFalse: [
    self atualizarDadosNaTela.]
```

excluirRegistro

```
  inserindo ifTrue: [elementos removeLast. elementos cursorToLast. self atualizarDadosNaTela. inserindo ← false.] ifFalse: [
    elementos elementAtCursor isNil ifFalse: [sessao excluir: (elementos elementAtCursor). sessao gravarEContinuarSessao. elementos remove: (elementos elementAtCursor). elementos adjustCursor. self atualizarDadosNaTela.] ]
```

gravarRegistro

```
  inserindo ifTrue: [sessao registrarNovo: (elementos elementAtCursor). self atualizarDadosDoRegistro. sessao gravarEContinuarSessao. inserindo ← false.] ifFalse: [
    self atualizarDadosDoRegistro. sessao gravarEContinuarSessao.]
```

iniciarSessao
sessao ← CMGerenciadorPesistencia new. sessao iniciarSessao. elementos ← OrderedCollectionWithCursor withAll: (sessao recuperarEntidadesDoTipo: CMMedico). self atualizarDadosNaTela.

initialize
super initialize. self buildOn: self. corpo ← self. inserindo ← false.

inserirRegistro
elementos addLast: (CMMedico new). elementos cursorToLast. self atualizarDadosNaTela. inserindo ← true.

irParaPrimeiroRegistro
elementos cursorToFirst.
self atualizarDadosNaTela.

irParaProximoRegistro
elementos advanceCursor.
self atualizarDadosNaTela.

irParaRegistroAnterior
elementos retrocedeCursor.
self atualizarDadosNaTela.

irParaUltimoRegistro
elementos cursorToLast.
self atualizarDadosNaTela.

openInSystemWindow
```
| nm |
self world
   ifNotNil: [self delete].
(nm ← Prefab managerFrameClass new) client: self;
   setLabel: title;
   openInWorld.
nm
   color: (Color
         r: 0.851
         g: 0.851
         b: 0.851).
nm setLabel: 'Navegador de entidades CMMedico'.
self iniciarSessao.  ↑ self
```

terminarSessao
sessao terminarSessao. elementos ← OrderedCollectionWithCursor new.

Methods for category: system

abrir
self openInSystemWindow.

PrefabManager subclass: CMNavegadorDeCMPaciente
Category: ClinicaMedica
Instance variable names: sessao elementos inserindo corpo

Instance protocol

Methods for category: private

atualizarDadosDoRegistro

```
elementos elementAtCursor isNil ifFalse: [
elementos elementAtCursor nome: (corpo clientWithTag: 'nome') text
elementos elementAtCursor cpf: (corpo clientWithTag: 'cpf') text
elementos elementAtCursor codigoDeCadastro: (corpo clientWithTag: 'codigoDeCadastro') text
elementos elementAtCursor dataNascimento: (corpo clientWithTag: 'dataNascimento') text
]
```

atualizarDadosNaTela

```
elementos elementAtCursor isNil ifFalse: [
(elementos elementAtCursor nome) isNil ifTrue: [ (corpo clientWithTag: 'nome') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'nome') text: (elementos elementAtCursor nome asString).].
(elementos elementAtCursor cpf) isNil ifTrue: [ (corpo clientWithTag: 'cpf') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'cpf') text: (elementos elementAtCursor cpf asString).].
(elementos elementAtCursor codigoDeCadastro) isNil ifTrue: [ (corpo clientWithTag: 'codigoDeCadastro') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'codigoDeCadastro') text: (elementos elementAtCursor codigoDeCadastro asString).].
(elementos elementAtCursor dataNascimento) isNil ifTrue: [ (corpo clientWithTag: 'dataNascimento') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'dataNascimento') text: (elementos elementAtCursor dataNascimento asString).].
] ifTrue: [
(corpo clientWithTag: 'nome') text: ' '.
(corpo clientWithTag: 'cpf') text: ' '.
(corpo clientWithTag: 'codigoDeCadastro') text: ' '.
(corpo clientWithTag: 'dataNascimento') text: ' '.
]
```

buildOn: *aManager*

```
aManager
    extent: 579@180;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    minSize: 10@10;
    maxSize: 1073741823@1073741823.
  aManager addClient: (PrefabFrame new
    extent: 536@35;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    style: #inset;
    yourself)
    atRelPosition: 13@10.
aManager addClient: (PrefabButton new
    extent: 64@28;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    action: [self irParaPrimeiroRegistro];
    tag: #botaoPrimeiro;
    text: ' Primeiro ';
    contentOrientation: #centered;
    yourself)
    atRelPosition: 19@13.
  aManager addClient: (PrefabButton new
    extent: 79@27;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    action: [self cancelarAlteracoes];
    tag: #botaoCancelar;
    text: 'Cancelar alt.';
    contentOrientation: #centered;
    yourself)
    atRelPosition: 462@14.
```

```
    aManager addClient: (PrefabButton new
        extent: 56@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self irParaProximoRegistro];
        tag: #botaoProximo;
        text: 'Proximo';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 147@13.
    aManager addClient: (PrefabButton new
        extent: 43@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self irParaUltimoRegistro];
        tag: #botaoUltimo;
        text: 'Ultimo';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 204@13.
    aManager addClient: (PrefabButton new
        extent: 68@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self gravarRegistro];
        tag: #botaoSalvar;
        text: 'Salvar alt.';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 394@14.
    aManager addClient: (PrefabButton new
        extent: 62@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self irParaRegistroAnterior];
        tag: #botaoAnterior;
        text: 'Anterior';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 84@13.
    aManager addClient: (PrefabButton new
        extent: 46@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self excluirRegistro];
        tag: #botaoExcluir;
        text: 'Excluir';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 327@13.
    aManager addClient: (PrefabButton new
        extent: 54@26;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self inserirRegistro];
        tag: #botaoInserir;
        text: 'Inserir';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 273@13.
aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'nome';
        contentOrientation: #centered;
        yourself)
```

```
        atRelPosition: 10@60.
    aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'nome';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@60.
  aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'cpf';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@90.
    aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'cpf';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@90.
  aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'codigoDeCadastro';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@120.
    aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'codigoDeCadastro';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@120.
  aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'dataNascimento';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@150.
    aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'dataNascimento';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@150.
```

cancelarAlteracoes

```
inserindo ifTrue: [elementos removeLast. elementos adjustCursor. self atualizarDadosNaTela] ifFalse: [
    self atualizarDadosNaTela.]
```

excluirRegistro

```
inserindo ifTrue: [elementos removeLast. elementos cursorToLast. self atualizarDadosNaTela. inserindo ← false.] ifFalse: [
    elementos elementAtCursor isNil ifFalse: [sessao excluir: (elementos elementAtCursor). sessao gravarEContinuarSessao. elementos remove: (elementos elementAtCursor). elementos adjustCursor. self atualizarDadosNaTela.] ]
```

gravarRegistro

```
inserindo ifTrue: [sessao registrarNovo: (elementos elementAtCursor). self atualizarDadosDoRegistro. sessao gravarEContinuarSessao. inserindo ← false.] ifFalse: [
    self atualizarDadosDoRegistro. sessao gravarEContinuarSessao.]
```

iniciarSessao
sessao ← CMGerenciadorPesistencia new. sessao iniciarSessao. elementos ← OrderedCollectionWithCursor withAll: (sessao recuperarEntidadesDoTipo: CMPaciente). self atualizarDadosNaTela.

initialize
super initialize. self buildOn: self. corpo ← self. inserindo ← false.

inserirRegistro
elementos addLast: (CMPaciente new). elementos cursorToLast. self atualizarDadosNaTela. inserindo ← true.

irParaPrimeiroRegistro
elementos cursorToFirst.
self atualizarDadosNaTela.

irParaProximoRegistro
elementos advanceCursor.
self atualizarDadosNaTela.

irParaRegistroAnterior
elementos retrocedeCursor.
self atualizarDadosNaTela.

irParaUltimoRegistro
elementos cursorToLast.
self atualizarDadosNaTela.

openInSystemWindow
```
| nm |
self world
   ifNotNil: [self delete].
(nm ← Prefab managerFrameClass new) client: self;
   setLabel: title;
   openInWorld.
nm
   color: (Color
         r: 0.851
         g: 0.851
         b: 0.851).
nm setLabel: 'Navegador de entidades CMPaciente'.
```
self iniciarSessao. ↑ self

terminarSessao
sessao terminarSessao. elementos ← OrderedCollectionWithCursor new.

Methods for category: system

abrir
self openInSystemWindow.

PrefabManager subclass: CMNavegadorDeCMPessoa
Category: ClinicaMedica
Instance variable names: sessao elementos inserindo corpo

Instance protocol

Methods for category: private

atualizarDadosDoRegistro

```
elementos elementAtCursor isNil ifFalse: [
elementos elementAtCursor nome: (corpo clientWithTag: 'nome') text
elementos elementAtCursor cpf: (corpo clientWithTag: 'cpf') text
elementos elementAtCursor dataNascimento: (corpo clientWithTag: 'dataNascimento') text
]
```

atualizarDadosNaTela

```
elementos elementAtCursor isNil ifFalse: [
(elementos elementAtCursor nome) isNil ifTrue: [ (corpo clientWithTag: 'nome') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'nome') text: (elementos elementAtCursor nome asString).].
(elementos elementAtCursor cpf) isNil ifTrue: [ (corpo clientWithTag: 'cpf') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'cpf') text: (elementos elementAtCursor cpf asString).].
(elementos elementAtCursor dataNascimento) isNil ifTrue: [ (corpo clientWithTag: 'dataNascimento') text: ' ' ] ifFalse: [ (corpo clientWithTag: 'dataNascimento') text: (elementos elementAtCursor dataNascimento asString).].
] ifTrue: [
(corpo clientWithTag: 'nome') text: ' '.
(corpo clientWithTag: 'cpf') text: ' '.
(corpo clientWithTag: 'dataNascimento') text: ' '.
]
```

buildOn: *aManager*

```
aManager
    extent: 579@150;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    minSize: 10@10;
    maxSize: 1073741823@1073741823.
  aManager addClient: (PrefabFrame new
    extent: 536@35;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    style: #inset;
    yourself)
    atRelPosition: 13@10.
aManager addClient: (PrefabButton new
    extent: 64@28;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    action: [self irParaPrimeiroRegistro];
    tag: #botaoPrimeiro;
    text: ' Primeiro ';
    contentOrientation: #centered;
    yourself)
    atRelPosition: 19@13.
  aManager addClient: (PrefabButton new
    extent: 79@27;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
    action: [self cancelarAlteracoes];
    tag: #botaoCancelar;
    text: 'Cancelar alt.';
    contentOrientation: #centered;
    yourself)
    atRelPosition: 462@14.
  aManager addClient: (PrefabButton new
    extent: 56@28;
    color: (Color r: 0.851 g: 0.851 b: 0.851);
    foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
```

```
        action: [self irParaProximoRegistro];
        tag: #botaoProximo;
        text: 'Proximo';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 147@13.
    aManager addClient: (PrefabButton new
        extent: 43@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self irParaUltimoRegistro];
        tag: #botaoUltimo;
        text: 'Ultimo';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 204@13.
    aManager addClient: (PrefabButton new
        extent: 68@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self gravarRegistro];
        tag: #botaoSalvar;
        text: 'Salvar alt.';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 394@14.
    aManager addClient: (PrefabButton new
        extent: 62@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self irParaRegistroAnterior];
        tag: #botaoAnterior;
        text: 'Anterior';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 84@13.
    aManager addClient: (PrefabButton new
        extent: 46@27;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self excluirRegistro];
        tag: #botaoExcluir;
        text: 'Excluir';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 327@13.
    aManager addClient: (PrefabButton new
        extent: 54@26;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        action: [self inserirRegistro];
        tag: #botaoInserir;
        text: 'Inserir';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 273@13.
aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'nome';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@60.
    aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
```

```
        tag: 'nome';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@60.
    aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'cpf';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@90.
      aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'cpf';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@90.
    aManager addClient: (PrefabLabel new
        extent: 49@28;
        color: (Color r: 0.851 g: 0.851 b: 0.851);
        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
        text: 'dataNascimento';
        contentOrientation: #centered;
        yourself)
        atRelPosition: 10@120.
      aManager addClient: (PrefabEdit new
        extent: 473@20;
        color: (Color r: 0.0 g: 0.0 b: 0.0);
        tag: 'dataNascimento';
        foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);
        yourself)
        atRelPosition: 64@120.
```

cancelarAlteracoes

```
inserindo ifTrue: [elementos removeLast. elementos adjustCursor. self atualizarDadosNaTela] ifFalse: [
    self atualizarDadosNaTela.]
```

excluirRegistro

```
inserindo ifTrue: [elementos removeLast. elementos cursorToLast. self atualizarDadosNaTela. inserindo ← false.] ifFalse: [
    elementos elementAtCursor isNil ifFalse: [sessao excluir: (elementos elementAtCursor). sessao gravarEContinuarSessao. elementos remove: (elementos elementAtCursor). elementos adjustCursor. self atualizarDadosNaTela.] ]
```

gravarRegistro

```
inserindo ifTrue: [sessao registrarNovo: (elementos elementAtCursor). self atualizarDadosDoRegistro. sessao gravarEContinuarSessao. inserindo ← false.] ifFalse: [
    self atualizarDadosDoRegistro. sessao gravarEContinuarSessao.]
```

iniciarSessao

```
sessao ← CMGerenciadorPesistencia new. sessao iniciarSessao. elementos ← OrderedCollectionWithCursor withAll: (sessao recuperarEntidadesDoTipo: CMPessoa ). self atualizarDadosNaTela.
```

initialize

```
super initialize. self buildOn: self. corpo ← self. inserindo ← false.
```

inserirRegistro

```
elementos addLast: (CMPessoa new). elementos cursorToLast. self atualizarDadosNaTela. inserindo ← true.
```

irParaPrimeiroRegistro

```
elementos cursorToFirst.
self atualizarDadosNaTela.
```

irParaProximoRegistro

```
elementos advanceCursor.
self atualizarDadosNaTela.
```

irParaRegistroAnterior

```
elementos retrocedeCursor.
```

```
  self atualizarDadosNaTela.
irParaUltimoRegistro
  elementos cursorToLast.
  self atualizarDadosNaTela.
openInSystemWindow
    | nm |
    self world
      ifNotNil: [self delete].
    (nm ← Prefab managerFrameClass new) client: self;
      setLabel: title;
      openInWorld.
    nm
      color: (Color
          r: 0.851
          g: 0.851
          b: 0.851).
    nm setLabel: 'Navegador de entidades CMPessoa'.
  self iniciarSessao.  ↑ self
terminarSessao
  sessao terminarSessao. elementos ← OrderedCollectionWithCursor new.
```

Methods for category: system

```
abrir
  self openInSystemWindow.
```

Object subclass: CMPessoa

Category: ClinicaMedica
Instance variable names: dataNascimento nome cpf intid

Instance protocol

Methods for category: atributos

```
cpf
    ↑ cpf.
cpf: umCpf
  [ [:valorTMPSET | valorTMPSET asString ] value: umCpf] ifError: [self error: 'Valor passado incompativel!'].
    cpf ← [:valorTMPSET | valorTMPSET asString] value: umCpf
dataNascimento
    ↑ dataNascimento.
dataNascimento: umDataNascimento
  [ [:valorTMPSET | valorTMPSET asDate ] value: umDataNascimento] ifError: [self error: 'Valor passado incompativel!'].
    dataNascimento ← [:valorTMPSET | valorTMPSET asDate] value: umDataNascimento
intid
    ↑ intid.
intid: umIntid
  [ [:valorTMPSET | valorTMPSET asInteger ] value: umIntid] ifError: [self error: 'Valor passado incompativel!'].
    intid ← [:valorTMPSET | valorTMPSET asInteger] value: umIntid
nome
    ↑ nome.
nome: umNome
  [ [:valorTMPSET | valorTMPSET asString ] value: umNome] ifError: [self error: 'Valor passado incompativel!'].
    nome ← [:valorTMPSET | valorTMPSET asString] value: umNome
```

CMPessoa subclass: CMMedico
Category: ClinicaMedica
Instance variable names: crm consultasFeitas

Instance protocol

Methods for category: atributos

consultasFeitas
```
consultasFeitas isNil ifTrue: [ consultasFeitas ← TypedOrderedCollection new tipo: CMConsulta
].
↑ consultasFeitas.
```
consultasFeitas: *osConsultasFeitas*
```
consultasFeitas ← TypedOrderedCollection new tipo: CMConsulta.
consultasFeitas addAll: osConsultasFeitas.
```
crm
```
↑ crm.
```
crm: *umCrm*
```
[ [:valorTMPSET | valorTMPSET asInteger ] value: umCrm] ifError: [self error: 'Valor passado incompativel!'].
crm ← [:valorTMPSET | valorTMPSET asInteger] value: umCrm
```
Methods for category: metodos

totalGanhoEmConsultas!

CMPessoa subclass: CMPaciente
Category: ClinicaMedica
Instance variable names: codigoDeCadastro consultasRealizadas

Instance protocol

Methods for category: atributos

codigoDeCadastro
```
↑ codigoDeCadastro.
```
codigoDeCadastro: *umCodigoDeCadastro*
```
[ [:valorTMPSET | valorTMPSET asInteger ] value: umCodigoDeCadastro] ifError: [self error: 'Valor passado incompativel!'].
codigoDeCadastro ← [:valorTMPSET | valorTMPSET asInteger] value: umCodigoDeCadastro
```
consultasRealizadas
```
consultasRealizadas isNil ifTrue: [ consultasRealizadas ← TypedOrderedCollection new tipo: CMConsulta
].
↑ consultasRealizadas.
```
consultasRealizadas: *osConsultasRealizadas*
```
consultasRealizadas ← TypedOrderedCollection new tipo: CMConsulta.
consultasRealizadas addAll: osConsultasRealizadas.
```
Methods for category: metodos

totalGastoEmConsultas!
```
nt [ [:valorTMPSET | valorTMPSET asInteger ] value: umCodigoDeCadastro] ifError: [self error: 'Valor passado incompati-
vel!'].
codigoDeCadastro ← [:valorTMPSET | valorTMPSET asInteger] value: umCodigoDeCadastro
```

consultasRealizadas
```
consultasRealizadas isNil ifTrue: [ consultasRealizadas ← TypedOrderedCollection new tipo: CMConsulta
].
↑ consultasRealizadas.
```
consultasRealizadas: *osConsultasRealizadas*
```
consultasRealizadas ← TypedOrderedCollection new tipo: CMConsulta.
consultasRealizadas addAll: osConsultasRealizadas.
```
Methods for category: metodos

totalGastoEmConsultas!
```
bmitted for class: CMPaciente.
```

CMPessoa subclass: CMPaciente
Category: ClinicaMedica
Instance variable names: codigoDeCadastro consultasRealizadas

Instance protocol

Methods for category: atributos

codigoDeCadastro
↑ codigoDeCadastro.

codigoDeCadastro: *umCodigoDeCadastro*
[[:valorTMPSET | valorTMPSET asInteger] value: umCodigoDeCadastro] ifError: [self error: 'Valor passado incompativel!'].

codigoDeCadastro ← [:valorTMPSET | valorTMPSET asInteger] value: umCodigoDeCadastro

consultasRealizadas
consultasRealizadas isNil ifTrue: [consultasRealizadas ← TypedOrderedCollection new tipo: CMConsulta

].
↑ consultasRealizadas.

consultasRealizadas: *osConsultasRealizadas*
consultasRealizadas ← TypedOrderedCollection new tipo: CMConsulta.

consultasRealizadas addAll: osConsultasRealizadas.

Methods for category: metodos

totalGastoEmConsultas!

10.2 Anexo II - Fontes da ferramenta desenvolvida

Object subclass: XMIAssociacao
Category: XMI
Instance variable names: classeOrigem classeDestino multiplicidadeOrigem multiplicidadeDestino papelOrigem papelDestino

Esta classe representa uma associacao entre entidades em um documento XMI.

Class protocol

Methods for category: as yet unclassified

new
↑ super new initialize.

Instance protocol

Methods for category: accessing

classeDestino
↑ classeDestino!

classeDestino: *umXMIClasse*
classeDestino ← umXMIClasse!

classeOrigem
↑ classeOrigem
classeOrigem: *umXMIClasse*
classeOrigem ← umXMIClasse!

multiplicidadeDestino
↑ multiplicidadeDestino
multiplicidadeDestino: *umInteiro*
multiplicidadeDestino ← umInteiro!

multiplicidadeOrigem
↑ multiplicidadeOrigem
multiplicidadeOrigem: *umInteiro*
multiplicidadeOrigem ← umInteiro!

papelDestino
↑ papelDestino
papelDestino: *umTexto*
papelDestino ← umTexto!

papelOrigem
↑ papelOrigem
papelOrigem: *umTexto*
papelOrigem ← umTexto!

Methods for category: initialization

initialize
self papelDestino: ' '.
self papelOrigem: ' '.
self multiplicidadeOrigem: 1.
self multiplicidadeDestino: 1.!

Object subclass: XMIAtributo
Category: XMI
Instance variable names: nome tipo

Esta classe representa um atributo de uma entidade de um documento XMI.

Instance protocol

Methods for category: acessing

nome
↑ nome!

nome: *umTexto*
nome ← umTexto!

tipo
↑ tipo!

tipo: *umXMIClasse*
"O tipo de um atributo deve ser um XMIClasse. Ele pode ser tanto uma classe presente no documento XMI como uma classe ou como um datatype"
tipo ← umXMIClasse!

Object subclass: XMICabecalho
Category: XMI
Instance variable names: metamodel versao

Esta classe contem as informacoes de um cabecalho de documento XMI

Instance protocol

Methods for category: acessing

metamodel
↑ metamodel

metamodel: *umTextoDescrevendoOModelo*
"Indica qual tipo de meta−modelo foi utilizado para confeccionar o
documento "
metamodel ← umTextoDescrevendoOModelo!

versao
↑ versao

versao: *umTexto*
"Indica qual versao da notacao XMI foi utilizada na composicao do
documento "
versao ← umTexto!

Object subclass: XMIClasse
Category: XMI
Instance variable names: nome abstrata identificadorInterno atributos metodos

Esta classe representa uma classe em um documento XMI. Contem um conjunto de atributos e metodos.

Class protocol

Methods for category: as yet unclassified

new
↑ super new initialize.

Instance protocol

Methods for category: accessing

abstrata
"Se esta classe e abstrata"
↑ abstrata!

abstrata: *umBoolean*
abstrata ← umBoolean!

atributos
"Colecao de XMIAtributo que representam os atributos desta classe"
↑ atributos!

identificadorInterno
"O identificador deste elemento(classe) interno no documento XMI"
↑ identificadorInterno!

identificadorInterno: *umTextoIdentificando*
identificadorInterno ← umTextoIdentificando!

metodos
"Colecao de XMIMetodo que representam os metodos desta classe"
↑ metodos!

nome
"Nome da classe"
↑ nome!

nome: *umTexto*
nome ← umTexto!

Methods for category: comparing

= *outraClasse*
↑ outraClasse nome = self nome.

Methods for category: initialization

initialize
atributos ← OrderedCollection new.
metodos ← OrderedCollection new!

Object subclass: XMIDocumento
Category: XMI
Instance variable names: classes cabecalho herancas tiposDeDados associacoes

Esta classe representa um documento XMI. Ela contem um certo numero de classes, herancas e associacoes.

Class protocol

Methods for category: as yet unclassified

new

```
↑ super new initialize.
```

Instance protocol

Methods for category: accessing

associacoes

```
"Colecao de XMIAssociacao que representam as associacoes entre as classes do documento"
↑ associacoes!
```

cabecalho

```
"XMICabecalho deste documento"
↑ cabecalho!
```

cabecalho: *umXMICabecalho*

```
cabecalho ← umXMICabecalho!
```

classes

```
↑ classes.
```

herancas

```
"Colecao de XMIHeranca contendo as herancas do documento"
↑ herancas!
```

tiposDeDados

```
"Colecao de XMIClasse que representam os tipos de dados utilizados(dataTypes) no documento"
↑ tiposDeDados!
```

Methods for category: initialization

initialize

```
classes ← OrderedCollection new.
herancas ← OrderedCollection new.
tiposDeDados ← OrderedCollection new.
associacoes ← OrderedCollection new!
```

Methods for category: resolver

classeParaIdentificadorInterno: *umIdentificador*

```
"Dado umIdentificador, procura pela classe (ou tipo de dados) correspondente e a/o retorna"
classes
    do: [:classe | classe identificadorInterno = umIdentificador
        ifTrue: [↑ classe] ].
tiposDeDados
    do: [:dt | dt identificadorInterno = umIdentificador
        ifTrue: [↑ dt] ].
↑ nil!
```

classeParaNome: *umNome*

```
"Dado um nome de classe, retorna o XMIClasse correspondente ao mesmo
contido no documento"
classes
    do: [:classe | classe nome = umNome
        ifTrue: [↑ classe] ].
tiposDeDados
    do: [:dt | dt nome = umNome
        ifTrue: [↑ dt] ].
↑ nil!
```

criarClasseParaNome: *umNome*

```
"Instancia um novo XMIClasse sob o nome informado e o adiciona a
colecao de classes deste documento"
| ret |
```

```
ret ← self classeParaNome: umNome.
ret isNil
   ifTrue: [classes add: (ret ← XMIClasse new nome: umNome)].
↑ ret!
```

Object subclass: XMIHeranca
Category: XMI
Instance variable names: classeSuperior classeDescendente

Representa uma heranca entre classes de um documento XMI

Class protocol

Methods for category: as yet unclassified

new
```
↑ super new.
```
new: *classe* **herdaDe:** *superClasse*
```
↑ super new initialize: classe herdaDe: superClasse!
```

Instance protocol

Methods for category: accessing

classeDescendente
```
↑ classeDescendente!
```
classeDescendente: *umXMIClasse*
```
classeDescendente ← umXMIClasse!
```
classeSuperior
```
↑ classeSuperior!
```
classeSuperior: *umXMIClasse*
```
classeSuperior ← umXMIClasse!
```

Methods for category: initialization

initialize: *class* **herdaDe:** *superClasse*
```
self classeDescendente: class.
self classeSuperior: superClasse!
```

Object subclass: XMIMetodo
Category: XMI
Instance variable names: assinatura

Guarda um metodo de um XMIClasse

Instance protocol

Methods for category: accessing

assinatura
```
"Literal (string) contendo a assinatura do metodo"
↑ assinatura!
```
assinatura: *umTextoRepresentandoAAssinatura*
```
assinatura ← umTextoRepresentandoAAssinatura!
```

Object subclass: XMIMultiplicidade
Category: XMI
Instance variable names: superior inferior

Guarda os limites (inferior e superior) de um terminal de uma associacao

Class protocol

Methods for category: as yet unclassified

de: *inferior* **ate:** *superior*
↑ super new initialize: inferior ate: superior!

Instance protocol

Methods for category: accessing

inferior
↑ inferior!

inferior: *umInteiro*
"Limite inferior da multiplicidade do terminal da associacao"
inferior ← umInteiro!

superior
↑ superior!

superior: *umInteiro*
"Limite inferior da multiplicidade do terminal da associacao"
superior ← umInteiro!

Methods for category: initalization

initialize: *limiteInferior* **ate:** *limiteSuperior*
self inferior: limiteInferior.
self superior: limiteSuperior!

Object subclass: XMIParser
Category: XMI
Instance variable names: documentoXML documentoXMI

Esta classe abstrata define o que eh e o q faz um parser de um documento XMI

Class protocol

Methods for category: accessing

Methods for category: as yet unclassified

Instance protocol

Methods for category: accessing

documentoXMI
"Deve retornar o XMIDocumento originiario do parsing do documento XMI"
↑ documentoXMI!

Methods for category: initialization

stream: *umStream*
"Deve receber e processar o stream contendo o documento XMI"
↑ self subClassResponsibility!

XMIParser subclass: XMIParserTogether
Category: XMI

Realiza o parsing de um documento XMI exportado pelo Together Control Center.

Instance protocol

Methods for category: initialization

stream: *aStream*

```
  “Passa a trabalhar em cima do stream informado"
  documentoXML ← XMLDOMParser parseDocumentFrom: aStream.
  documentoXMI ← XMIDocumento new.
  self parseCabecalho.
  self parseClasses.
  self parseAssociacoes.
  self parseHerancas.
  self parseMetodos.
!
```

Methods for category: private

parseAssociacao: *nodo*

```
  | terminais associacao terminal multXMI |
  terminais ← OrderedCollection new addAll: (nodo elementAt: 'UML:Association.connection') elements.
  associacao ← XMIAssociacao new.
  terminal ← terminais at: 1.
  associacao
    papelOrigem: (terminal
        attributeAt: 'name'
        ifAbsent: [' ']).
  associacao
    classeOrigem: (documentoXMI
        classeParaIdentificadorInterno: (((terminal elementAt: 'UML:AssociationEnd.participant')
            elementAt: 'UML:Classifier')
            attributeAt: 'xmi.idref')).
  multXMI ← (((terminal elementAt: 'UML:AssociationEnd.multiplicity')
        elementAt: 'UML:Multiplicity')
        elementAt: 'UML:Multiplicity.range')
        elementAt: 'UML:MultiplicityRange'.
  associacao
    multiplicidadeOrigem: (XMIMultiplicidade
        de: (multXMI attributeAt: 'lower')
        ate: (multXMI attributeAt: 'upper')).
  terminal ← terminais at: 2.
  associacao
    papelDestino: (terminal
        attributeAt: 'name'
        ifAbsent: [' ']).
  associacao
    classeDestino: (documentoXMI
        classeParaIdentificadorInterno: (((terminal elementAt: 'UML:AssociationEnd.participant')
            elementAt: 'UML:Classifier')
            attributeAt: 'xmi.idref')).
  multXMI ← (((terminal elementAt: 'UML:AssociationEnd.multiplicity')
        elementAt: 'UML:Multiplicity')
        elementAt: 'UML:Multiplicity.range')
        elementAt: 'UML:MultiplicityRange'.
  associacao
    multiplicidadeDestino: (XMIMultiplicidade
        de: (multXMI attributeAt: 'lower')
        ate: (multXMI attributeAt: 'upper')).
  self documentoXMI associacoes add: associacao.
  ↑ associacao!
```

parseAssociacoes

```
  | nodoPrincipal |
  nodoPrincipal ← (((documentoXML elementAt: 'XMI')
        elementAt: 'XMI.content')
        elementAt: 'UML:Model')
```

```
            elementAt: 'UML:Namespace.ownedElement'.
    (nodoPrincipal elements
        select: [:element | element name = 'UML:Association'])
        do: [:nodo | self parseAssociacao: nodo].
!
```

parseAtributos: *nodoDeAtributos*

```
    | atributo atributos idInternoClasse |
    atributos ← OrderedCollection new.
    (nodoDeAtributos elements
        select: [:elemento | elemento name = 'UML:Attribute'])
        do: [:nodo | ((nodo attributeAt: 'name')
                beginsWith: 'lnk')
            ifFalse: [atributo ← XMIAtributo new.
                atributo
                    nome: (nodo attributeAt: 'name').
                idInternoClasse ← ((((nodo elementAt: 'UML:StructuralFeature.type')
                        elementAt: 'UML:Classifier')
                        elementAt: 'UML:Namespace.ownedElement')
                        elementAt: 'UML:DataType')
                        attributeAt: 'xmi.idref'.
                atributo
                    tipo: (documentoXMI classeParaIdentificadorInterno: idInternoClasse).
                atributos add: atributo.
                ] ].
    ↑ atributos!
```

parseCabecalho

```
    | nodo cabecalho |
    cabecalho ← XMICabecalho new.
    nodo ← documentoXML elementAt: 'XMI'.
    nodo isNil
        ifFalse: [nodo ← nodo elementAt: 'XMI.header'].
    nodo isNil
        ifFalse: [nodo ← nodo elementAt: 'XMI.metamodel'].
    nodo isNil
        ifFalse: [cabecalho
            metamodel: (nodo
                attributeAt: 'xmi.name'
                ifAbsent: [ ]).
          cabecalho
            versao: (nodo
                attributeAt: 'xmi.version'
                ifAbsent: [ ])].
    documentoXMI cabecalho: cabecalho.
!
```

parseClasse: *nodo*

```
    | classe |
    classe ← XMIClasse new.
    classe
        nome: (nodo attributeAt: 'name').
    classe abstrata: (nodo attributeAt: 'isAbstract')
          = 'yes'.
    classe
        identificadorInterno: (nodo attributeAt: 'xmi.id').
    documentoXMI classes add: classe.
!
```

parseClasses

```
    | nodoPrincipal |
    nodoPrincipal ← (((documentoXML elementAt: 'XMI')
            elementAt: 'XMI.content')
            elementAt: 'UML:Model')
            elementAt: 'UML:Namespace.ownedElement'.
    (nodoPrincipal elements
        select: [:nodo | nodo name = 'UML:Class'])
```

```
      do: [:nodo | self parseClasse: nodo].
   (nodoPrincipal elements
      select: [:nodo | nodo name = 'UML:DataType'])
      do: [:nodo | self parseTipoDeDados: nodo].
(nodoPrincipal elements
      select: [:nodo | nodo name = 'UML:Class'])
      do: [:nodo || atributos |
         atributos ← nodo elements
               detect: [:elemento | elemento name = 'UML:Classifier.feature']
               ifNone: [ ].
         atributos isNil
            ifFalse: [(documentoXMI
               classeParaIdentificadorInterno: (nodo attributeAt: 'xmi.id')) atributos
               addAll: (self parseAtributos: atributos)] ]!
```

parseHeranca: *nodo*

```
   | classeSuperior classeDescendente nodoGeneralizacao |
   (nodo elementAt: 'UML:Namespace.ownedElement') isNil
      ifFalse: [(nodoGeneralizacao ← (nodo elementAt: 'UML:Namespace.ownedElement')
                  elementAt: 'UML:Generalization') isNil
            ifFalse: [
               classeSuperior ← documentoXMI
                     classeParaIdentificadorInterno: (((nodoGeneralizacao elementAt: 'UML:Generalization.parent')
                           elementAt: 'UML:GeneralizableElement')
                           attributeAt: 'xmi.idref').
               classeDescendente ← documentoXMI
                     classeParaIdentificadorInterno: (((nodoGeneralizacao elementAt: 'UML:Generalization.child')
                           elementAt: 'UML:GeneralizableElement')
                           attributeAt: 'xmi.idref').
               documentoXMI herancas
                  add: (XMIHeranca new: classeDescendente herdaDe: classeSuperior)].
]!
```

parseHerancas

```
   | nodoPrincipal |
   nodoPrincipal ← (((documentoXML elementAt: 'XMI')
            elementAt: 'XMI.content')
            elementAt: 'UML:Model')
            elementAt: 'UML:Namespace.ownedElement'.
   (nodoPrincipal elements
      select: [:nodo | nodo name = 'UML:Class'])
      do: [:nodo | self parseHeranca: nodo].
!
```

parseMetodos

```
   "A definicao de metodos do squeak nao eh compativel com o Together,
   portanto nao eh feito o parsing dos mesmos"
   ↑ nil!
```

parseTipoDeDados: *nodo*

```
   | tipoDeDados |
   tipoDeDados ← XMIClasse new.
   tipoDeDados
      nome: (nodo attributeAt: 'name').
   (documentoXMI classeParaNome: tipoDeDados nome) isNil
      ifTrue: [tipoDeDados
            identificadorInterno: (nodo attributeAt: 'xmi.id').
         documentoXMI tiposDeDados add: tipoDeDados]!

     t xmi.idref = ' 'S.30' '/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ' 'S.32' '
name = ' 'ModelosDeMoto' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isAbstract = ' 'false' ' isActive = ' 'false' '>
```

<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ' 'G.22' '
name = ' '' ' visibility = ' 'public' ' isSpecification = ' 'false' '
discriminator = ' '' '>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ' 'S.32' '/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ' 'S.30' '/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ' 'S.30' '
name = ' 'ModelosDeAutomoveis' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isAbstract = ' 'false' ' isActive = ' 'false' '>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ' 'S.33' '
name = ' 'nome' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.1' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
</UML:Class>
<UML:Class xmi.id = ' 'S.34' '
name = ' 'Documentacao' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isAbstract = ' 'false' ' isActive = ' 'false' '>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ' 'S.35' '
name = ' 'CPF' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.1' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ' 'S.36' '
name = ' 'RG' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>

</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ''G.1''/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
</UML:Class>
<!--From Class Pessoa to Class Documentacao-->
<UML:Association xmi.id = ''G.0''
name = ''{Pessoa-Documentacao}'' visibility = ''private'' isSpecification = ''false''
isAbstract = ''false''>
<UML:Association.connection>
<UML:AssociationEnd name="papelOrigem"xmi.id = ''G.23'' visibility = ''public'' isSpecification = ''false''
isNavigable = ''true'' ordering = ''unordered'' aggregation = ''none''
targetScope = ''instance'' changeability = ''changeable''>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ''1'' upper = ''1''/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ''S.4''/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd name="papelDestino"xmi.id = ''G.24'' visibility = ''public'' isSpecification = ''false''
isNavigable = ''true'' ordering = ''unordered'' aggregation = ''none''
targetScope = ''instance'' changeability = ''changeable''>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ''1'' upper = ''1''/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ''S.34''/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<!--From Class Motorista to Class Automovel-->
<UML:Association xmi.id = ''G.4''
name = ''{Motorista-Automovel}'' visibility = ''private'' isSpecification = ''false''
isAbstract = ''false''>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = ''G.25'' visibility = ''public'' isSpecification = ''false''
isNavigable = ''true'' ordering = ''unordered'' aggregation = ''none''
targetScope = ''instance'' changeability = ''changeable''>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ''0'' upper = ''-1''/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ''S.8''/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id = ''G.26'' visibility = ''public'' isSpecification = ''false''

isNavigable = ’ ’true’ ’ ordering = ’ ’unordered’ ’ aggregation = ’ ’none’ ’
targetScope = ’ ’instance’ ’ changeability = ’ ’changeable’ ’>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’0’ ’ upper = ’ ’−1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ’ ’S.11’ ’/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<!−−From Class Automovel to Class Fabricante−−→
<UML:Association xmi.id = ’ ’G.8’ ’
name = ’ ’{Automovel−Fabricante}’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = ’ ’G.27’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isNavigable = ’ ’true’ ’ ordering = ’ ’unordered’ ’ aggregation = ’ ’none’ ’
targetScope = ’ ’instance’ ’ changeability = ’ ’changeable’ ’>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’−1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ’ ’S.11’ ’/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id = ’ ’G.28’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isNavigable = ’ ’true’ ’ ordering = ’ ’unordered’ ’ aggregation = ’ ’none’ ’
targetScope = ’ ’instance’ ’ changeability = ’ ’changeable’ ’>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ’ ’S.27’ ’/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<!−−From Class Carro to Class ModelosDeCarro−−→
<UML:Association xmi.id = ’ ’G.11’ ’
name = ’ ’{Carro−ModelosDeCarro}’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = ’ ’G.29’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isNavigable = ’ ’true’ ’ ordering = ’ ’unordered’ ’ aggregation = ’ ’none’ ’
targetScope = ’ ’instance’ ’ changeability = ’ ’changeable’ ’>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’−1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ’ ’S.17’ ’/>

</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id = ''G.30'' visibility = ''public'' isSpecification = ''false''
isNavigable = ''true'' ordering = ''unordered'' aggregation = ''none''
targetScope = ''instance'' changeability = ''changeable''>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ''1'' upper = ''1''/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ''S.29''/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<!−−From Class Moto to Class ModelosDeMoto−→
<UML:Association xmi.id = ''G.14''
name = ''{Moto−ModelosDeMoto}'' visibility = ''private'' isSpecification = ''false''
isAbstract = ''false''>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = ''G.31'' visibility = ''public'' isSpecification = ''false''
isNavigable = ''true'' ordering = ''unordered'' aggregation = ''none''
targetScope = ''instance'' changeability = ''changeable''>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ''1'' upper = ''−1''/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ''S.20''/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id = ''G.32'' visibility = ''public'' isSpecification = ''false''
isNavigable = ''true'' ordering = ''unordered'' aggregation = ''none''
targetScope = ''instance'' changeability = ''changeable''>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ''1'' upper = ''1''/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ''S.32''/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<!−−From Class Caminhao to Class ModelosDeCaminhao−→
<UML:Association xmi.id = ''G.17''
name = ''{Caminhao−ModelosDeCaminhao}'' visibility = ''private'' isSpecification = ''false''
isAbstract = ''false''>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = ''G.33'' visibility = ''public'' isSpecification = ''false''
isNavigable = ''true'' ordering = ''unordered'' aggregation = ''none''
targetScope = ''instance'' changeability = ''changeable''>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ''1'' upper = ''−1''/>
</UML:Multiplicity.range>
</UML:Multiplicity>

```
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ''S.23''/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id = ''G.34'' visibility = ''public'' isSpecification = ''false''
isNavigable = ''true'' ordering = ''unordered'' aggregation = ''none''
targetScope = ''instance'' changeability = ''changeable''>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ''1'' upper = ''1''/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ''S.31''/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<UML:DataType xmi.id = ''G.1''
name = ''Texto'' visibility = ''public'' isSpecification = ''false''/>
<UML:DataType xmi.id = ''G.2''
name = ''Data'' visibility = ''public'' isSpecification = ''false''/>
<UML:DataType xmi.id = ''G.3''
name = ''Documentacao'' visibility = ''public'' isSpecification = ''false''/>
<UML:DataType xmi.id = ''G.5''
name = ''Automovel'' visibility = ''public'' isSpecification = ''false''/>
<UML:DataType xmi.id = ''G.6''
name = ''Inteiro'' visibility = ''public'' isSpecification = ''false''/>
<UML:DataType xmi.id = ''G.9''
name = ''Fabricante'' visibility = ''public'' isSpecification = ''false''/>
<UML:DataType xmi.id = ''G.10''
name = ''Valor'' visibility = ''public'' isSpecification = ''false''/>
<UML:DataType xmi.id = ''G.12''
name = ''ModelosDeCarro'' visibility = ''public'' isSpecification = ''false''/>
<UML:DataType xmi.id = ''G.15''
name = ''ModelosDeMoto'' visibility = ''public'' isSpecification = ''false''/>
<UML:DataType xmi.id = ''G.18''
name = ''ModelosDeCaminhao'' visibility = ''public'' isSpecification = ''false''/>
</UML:Namespace.ownedElement>
</UML:Model>
<UML:TaggedValue xmi.id = ''XX.1''
name = ''clientCardinality''
modelElement = ''S.7''>
<UML:TaggedValue.dataValue>
  1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ''XX.2''
name = ''supplierCardinality''
modelElement = ''S.7''>
<UML:TaggedValue.dataValue>
  1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ''XX.4''
name = ''clientCardinality''
modelElement = ''S.9''>
<UML:TaggedValue.dataValue>
  0..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ''XX.5''
name = ''supplierCardinality''
modelElement = ''S.9''>
```

<UML:TaggedValue.dataValue>
0..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.7’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’S.12’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.8’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’S.12’ ’>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.10’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’S.18’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.11’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’S.18’ ’>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.13’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’S.21’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.14’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’S.21’ ’>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.16’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’S.24’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.17’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’S.24’ ’>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.27’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’G.0’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>

<UML:TaggedValue xmi.id = ’ ’XX.28’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’G.0’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.32’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’G.4’ ’>
<UML:TaggedValue.dataValue>
0..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.33’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’G.4’ ’>
<UML:TaggedValue.dataValue>
0..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.37’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’G.8’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.38’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’G.8’ ’>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.42’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’G.11’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.43’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’G.11’ ’>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.47’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’G.14’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.48’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’G.14’ ’>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.52’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’G.17’ ’>
<UML:TaggedValue.dataValue>

1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ''XX.53''
name = ''clientCardinality''
modelElement = ''G.17''>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
</XMI.content>
</XMI>'!

testParseXMI

xmi ← XMIParserTogether new stream: stream.
xmi explore. !

Object subclass: XMIAssociacao

Category: XMI
Instance variable names: classeOrigem classeDestino multiplicidadeOrigem multiplicidadeDestino papelOrigem papelDestino

Esta classe representa uma associacao entre entidades em um documento XMI.

Class protocol

Methods for category: as yet unclassified

new
↑ super new initialize.

Instance protocol

Methods for category: accessing

classeDestino
↑ classeDestino!

classeDestino: *umXMIClasse*
classeDestino ← umXMIClasse!

classeOrigem
↑ classeOrigem
classeOrigem: *umXMIClasse*
classeOrigem ← umXMIClasse!

multiplicidadeDestino
↑ multiplicidadeDestino
multiplicidadeDestino: *umInteiro*
multiplicidadeDestino ← umInteiro!

multiplicidadeOrigem
↑ multiplicidadeOrigem
multiplicidadeOrigem: *umInteiro*
multiplicidadeOrigem ← umInteiro!

papelDestino
↑ papelDestino
papelDestino: *umTexto*
papelDestino ← umTexto!

papelOrigem
↑ papelOrigem
papelOrigem: *umTexto*
papelOrigem ← umTexto!

Methods for category: initialization

initialize
self papelDestino: ' '.
self papelOrigem: ' '.
self multiplicidadeOrigem: 1.
self multiplicidadeDestino: 1.!

Object subclass: XMIAtributo
Category: XMI
Instance variable names: nome tipo

Esta classe representa um atributo de uma entidade de um documento XMI.

Instance protocol

Methods for category: acessing

nome
↑ nome!

nome: *umTexto*
nome ← umTexto!

tipo
↑ tipo!

tipo: *umXMIClasse*
"O tipo de um atributo deve ser um XMIClasse. Ele pode ser tanto uma classe presente no documento XMI como uma classe ou como um datatype"
tipo ← umXMIClasse!

Object subclass: XMICabecalho
Category: XMI
Instance variable names: metamodel versao

Esta classe contem as informacoes de um cabecalho de documento XMI

Instance protocol

Methods for category: acessing

metamodel
↑ metamodel

metamodel: *umTextoDescrevendoOModelo*
"Indica qual tipo de meta−modelo foi utilizado para confeccionar o documento "
metamodel ← umTextoDescrevendoOModelo!

versao
↑ versao

versao: *umTexto*
"Indica qual versao da notacao XMI foi utilizada na composicao do documento "
versao ← umTexto!

Object subclass: XMIClasse
Category: XMI
Instance variable names: nome abstrata identificadorInterno atributos metodos

Esta classe representa uma classe em um documento XMI. Contem um conjunto de atributos e metodos.

Class protocol

Methods for category: as yet unclassified

new
↑ super new initialize.

Instance protocol

Methods for category: accessing

abstrata
"Se esta classe e abstrata"
↑ abstrata!

abstrata: *umBoolean*
abstrata ← umBoolean!

atributos
"Colecao de XMIAtributo que representam os atributos desta classe"
↑ atributos!

identificadorInterno
"O identificador deste elemento(classe) interno no documento XMI"
↑ identificadorInterno!

identificadorInterno: *umTextoIdentificando*
identificadorInterno ← umTextoIdentificando!

metodos
"Colecao de XMIMetodo que representam os metodos desta classe"
↑ metodos!

nome
"Nome da classe"
↑ nome!

nome: *umTexto*
nome ← umTexto!

Methods for category: comparing

= *outraClasse*
↑ outraClasse nome = self nome.

Methods for category: initialization

initialize
atributos ← OrderedCollection new.
metodos ← OrderedCollection new!

Object subclass: XMIDocumento
Category: XMI
Instance variable names: classes cabecalho herancas tiposDeDados associacoes

Esta classe representa um documento XMI. Ela contem um certo numero de classes, herancas e associacoes.

Class protocol

Methods for category: as yet unclassified

new
↑ super new initialize.

Instance protocol

Methods for category: accessing

associacoes
"Colecao de XMIAssociacao que representam as associacoes entre as classes do documento"
↑ associacoes!

cabecalho
"XMICabecalho deste documento"
↑ cabecalho!

cabecalho: *umXMICabecalho*
cabecalho ← umXMICabecalho!

classes
↑ classes.

herancas
"Colecao de XMIHeranca contendo as herancas do documento"
↑ herancas!

tiposDeDados
"Colecao de XMIClasse que representam os tipos de dados utilizados(dataTypes) no documento"
↑ tiposDeDados!

Methods for category: initialization

initialize
```
classes ← OrderedCollection new.
herancas ← OrderedCollection new.
tiposDeDados ← OrderedCollection new.
associacoes ← OrderedCollection new!
```

Methods for category: resolver

classeParaIdentificadorInterno: *umIdentificador*
```
"Dado umIdentificador, procura pela classe (ou tipo de dados) correspondente e a/o retorna"
classes
    do: [:classe | classe identificadorInterno = umIdentificador
        ifTrue: [↑ classe] ].
tiposDeDados
    do: [:dt | dt identificadorInterno = umIdentificador
        ifTrue: [↑ dt] ].
↑ nil!
```

classeParaNome: *umNome*
```
"Dado um nome de classe, retorna o XMIClasse correspondente ao mesmo
contido no documento"
classes
    do: [:classe | classe nome = umNome
        ifTrue: [↑ classe] ].
tiposDeDados
    do: [:dt | dt nome = umNome
        ifTrue: [↑ dt] ].
```

```
    ↑ nil!

criarClasseParaNome: umNome
    "Instancia um novo XMIClasse sob o nome informado e o adiciona a
    colecao de classes deste documento"
    | ret |
    ret ← self classeParaNome: umNome.
    ret isNil
        ifTrue: [classes add: (ret ← XMIClasse new nome: umNome)].
    ↑ ret!
```

Object subclass: XMIHeranca
Category: XMI
Instance variable names: classeSuperior classeDescendente

Representa uma heranca entre classes de um documento XMI

Class protocol

Methods for category: as yet unclassified

```
new
    ↑ super new.
new: classe herdaDe: superClasse
    ↑ super new initialize: classe herdaDe: superClasse!
```

Instance protocol

Methods for category: accessing

```
classeDescendente
    ↑ classeDescendente!

classeDescendente: umXMIClasse
    classeDescendente ← umXMIClasse!

classeSuperior
    ↑ classeSuperior!

classeSuperior: umXMIClasse
    classeSuperior ← umXMIClasse!
```

Methods for category: initialization

```
initialize: class herdaDe: superClasse
    self classeDescendente: class.
    self classeSuperior: superClasse!
```

Object subclass: XMIMetodo
Category: XMI
Instance variable names: assinatura

Guarda um metodo de um XMIClasse

Instance protocol

Methods for category: accessing

assinatura
"Literal (string) contendo a assinatura do metodo"
↑ assinatura!

assinatura: *umTextoRepresentandoAAssinatura*
assinatura ← umTextoRepresentandoAAssinatura!

Object subclass: XMIMultiplicidade
Category: XMI
Instance variable names: superior inferior

Guarda os limites (inferior e superior) de um terminal de uma associacao

Class protocol

Methods for category: as yet unclassified

de: *inferior* **ate:** *superior*
↑ super new initialize: inferior ate: superior!

Instance protocol

Methods for category: accessing

inferior
↑ inferior!

inferior: *umInteiro*
"Limite inferior da multiplicidade do terminal da associacao"
inferior ← umInteiro!

superior
↑ superior!

superior: *umInteiro*
"Limite inferior da multiplicidade do terminal da associacao"
superior ← umInteiro!

Methods for category: initalization

initialize: *limiteInferior* **ate:** *limiteSuperior*
self inferior: limiteInferior.
self superior: limiteSuperior!

Object subclass: XMIParser
Category: XMI
Instance variable names: documentoXML documentoXMI

Esta classe abstrata define o que eh e o q faz um parser de um documento XMI

Class protocol

Methods for category: accessing

Methods for category: as yet unclassified

Instance protocol

Methods for category: accessing

documentoXMI
"Deve retornar o XMIDocumento originirio do parsing do documento XMI"
↑ documentoXMI!

Methods for category: initialization

stream: *umStream*
"Deve receber e processar o stream contendo o documento XMI"
↑ self subClassResponsibility!

XMIParser subclass: XMIParserTogether
Category: XMI

Realiza o parsing de um documento XMI exportado pelo Together Control Center.

Instance protocol

Methods for category: initialization

stream: *aStream*

```
"Passa a trabalhar em cima do stream informado"
documentoXML ← XMLDOMParser parseDocumentFrom: aStream.
documentoXMI ← XMIDocumento new.
self parseCabecalho.
self parseClasses.
self parseAssociacoes.
self parseHerancas.
self parseMetodos.
```

!

Methods for category: private

parseAssociacao: *nodo*

```
| terminais associacao terminal multXMI |
terminais ← OrderedCollection new addAll: (nodo elementAt: 'UML:Association.connection') elements.
associacao ← XMIAssociacao new.
terminal ← terminais at: 1.
associacao
    papelOrigem: (terminal
        attributeAt: 'name'
        ifAbsent: [' ']).
associacao
    classeOrigem: (documentoXMI
        classeParaIdentificadorInterno: (((terminal elementAt: 'UML:AssociationEnd.participant')
            elementAt: 'UML:Classifier')
            attributeAt: 'xmi.idref')).
multXMI ← (((terminal elementAt: 'UML:AssociationEnd.multiplicity')
        elementAt: 'UML:Multiplicity')
        elementAt: 'UML:Multiplicity.range')
        elementAt: 'UML:MultiplicityRange'.
associacao
    multiplicidadeOrigem: (XMIMultiplicidade
        de: (multXMI attributeAt: 'lower')
        ate: (multXMI attributeAt: 'upper')).
terminal ← terminais at: 2.
associacao
    papelDestino: (terminal
        attributeAt: 'name'
        ifAbsent: [' ']).
associacao
    classeDestino: (documentoXMI
        classeParaIdentificadorInterno: (((terminal elementAt: 'UML:AssociationEnd.participant')
            elementAt: 'UML:Classifier')
            attributeAt: 'xmi.idref')).
multXMI ← (((terminal elementAt: 'UML:AssociationEnd.multiplicity')
        elementAt: 'UML:Multiplicity')
        elementAt: 'UML:Multiplicity.range')
        elementAt: 'UML:MultiplicityRange'.
associacao
    multiplicidadeDestino: (XMIMultiplicidade
        de: (multXMI attributeAt: 'lower')
        ate: (multXMI attributeAt: 'upper')).
self documentoXMI associacoes add: associacao.
↑ associacao!
```

parseAssociacoes

```
| nodoPrincipal |
```

```
  nodoPrincipal ← (((documentoXML elementAt: 'XMI')
        elementAt: 'XMI.content')
        elementAt: 'UML:Model')
        elementAt: 'UML:Namespace.ownedElement'.
  (nodoPrincipal elements
    select: [:element | element name = 'UML:Association'])
    do: [:nodo | self parseAssociacao: nodo].
!
```

parseAtributos: *nodoDeAtributos*

```
  | atributo atributos idInternoClasse |
  atributos ← OrderedCollection new.
  (nodoDeAtributos elements
    select: [:elemento | elemento name = 'UML:Attribute'])
    do: [:nodo | ((nodo attributeAt: 'name')
          beginsWith: 'lnk')
        ifFalse: [atributo ← XMIAtributo new.
          atributo
            nome: (nodo attributeAt: 'name').
          idInternoClasse ← ((((nodo elementAt: 'UML:StructuralFeature.type')
                elementAt: 'UML:Classifier')
                elementAt: 'UML:Namespace.ownedElement')
                elementAt: 'UML:DataType')
                attributeAt: 'xmi.idref'.
          atributo
            tipo: (documentoXMI classeParaIdentificadorInterno: idInternoClasse).
          atributos add: atributo.
          ] ].
  ↑ atributos!
```

parseCabecalho

```
  | nodo cabecalho |
  cabecalho ← XMICabecalho new.
  nodo ← documentoXML elementAt: 'XMI'.
  nodo isNil
    ifFalse: [nodo ← nodo elementAt: 'XMI.header'].
  nodo isNil
    ifFalse: [nodo ← nodo elementAt: 'XMI.metamodel'].
  nodo isNil
    ifFalse: [cabecalho
        metamodel: (nodo
            attributeAt: 'xmi.name'
            ifAbsent: [ ]).
      cabecalho
        versao: (nodo
            attributeAt: 'xmi.version'
            ifAbsent: [ ])].
  documentoXMI cabecalho: cabecalho.
!
```

parseClasse: *nodo*

```
  | classe |
  classe ← XMIClasse new.
  classe
    nome: (nodo attributeAt: 'name').
  classe abstrata: (nodo attributeAt: 'isAbstract')
      = 'yes'.
  classe
    identificadorInterno: (nodo attributeAt: 'xmi.id').
  documentoXMI classes add: classe.
!
```

parseClasses

```
  | nodoPrincipal |
  nodoPrincipal ← (((documentoXML elementAt: 'XMI')
        elementAt: 'XMI.content')
        elementAt: 'UML:Model')
```

```
            elementAt: 'UML:Namespace.ownedElement'.
    (nodoPrincipal elements
        select: [:nodo | nodo name = 'UML:Class'])
        do: [:nodo | self parseClasse: nodo].
    (nodoPrincipal elements
        select: [:nodo | nodo name = 'UML:DataType'])
        do: [:nodo | self parseTipoDeDados: nodo].
(nodoPrincipal elements
        select: [:nodo | nodo name = 'UML:Class'])
        do: [:nodo || atributos |
            atributos ← nodo elements
                    detect: [:elemento | elemento name = 'UML:Classifier.feature']
                    ifNone: [ ].
            atributos isNil
                ifFalse: [(documentoXMI
                    classeParaIdentificadorInterno: (nodo attributeAt: 'xmi.id')) atributos
                    addAll: (self parseAtributos: atributos)] ]!
```

parseHeranca: *nodo*

```
    | classeSuperior classeDescendente nodoGeneralizacao |
    (nodo elementAt: 'UML:Namespace.ownedElement') isNil
        ifFalse: [(nodoGeneralizacao ← (nodo elementAt: 'UML:Namespace.ownedElement')
                    elementAt: 'UML:Generalization') isNil
                ifFalse: [
                    classeSuperior ← documentoXMI
                            classeParaIdentificadorInterno: (((nodoGeneralizacao elementAt: 'UML:Generalization.parent')
                                elementAt: 'UML:GeneralizableElement')
                                attributeAt: 'xmi.idref').
                    classeDescendente ← documentoXMI
                            classeParaIdentificadorInterno: (((nodoGeneralizacao elementAt: 'UML:Generalization.child')
                                elementAt: 'UML:GeneralizableElement')
                                attributeAt: 'xmi.idref').
                    documentoXMI herancas
                        add: (XMIHeranca new: classeDescendente herdaDe: classeSuperior)].
]!
```

parseHerancas

```
    | nodoPrincipal |
    nodoPrincipal ← (((documentoXML elementAt: 'XMI')
            elementAt: 'XMI.content')
            elementAt: 'UML:Model')
            elementAt: 'UML:Namespace.ownedElement'.
    (nodoPrincipal elements
        select: [:nodo | nodo name = 'UML:Class'])
        do: [:nodo | self parseHeranca: nodo].
!
```

parseMetodos

```
    "A definicao de metodos do squeak nao eh compativel com o Together,
    portanto nao eh feito o parsing dos mesmos"
    ↑ nil!
```

parseTipoDeDados: *nodo*

```
    | tipoDeDados |
    tipoDeDados ← XMIClasse new.
    tipoDeDados
        nome: (nodo attributeAt: 'name').
    (documentoXMI classeParaNome: tipoDeDados nome) isNil
        ifTrue: [tipoDeDados
            identificadorInterno: (nodo attributeAt: 'xmi.id').
          documentoXMI tiposDeDados add: tipoDeDados]!

      parsing dos mesmos"
    ↑ nil!
```

parseTipoDeDados: *nodo*

```
| tipoDeDados |
tipoDeDados ← XMIClasse new.
tipoDeDados
   nome: (nodo attributeAt: 'name').
(documentoXMI classeParaNome: tipoDeDados nome) isNil
   ifTrue: [tipoDeDados
         identificadorInterno: (nodo attributeAt: 'xmi.id').
      documentoXMI tiposDeDados add: tipoDeDados]!
```

OrderedCollection subclass: LimitedOrderedCollection
Category: OORDBM
Instance variable names: capacity

Extende uma OrderedCollection, dando a possibilidade de definir um tamanho maximo(capacity:) para a mesma

Class protocol

Methods for category: as yet unclassified

new: *aValue*

```
| retorno |
retorno ← super new: aValue.
retorno initialize.
retorno capacity: aValue.
↑ retorno!
```

Methods for category: nil

new

```
| retorno |
retorno ← super new.
retorno initialize.
retorno capacity: 0.
↑ retorno!
```

Instance protocol

Methods for category: as yet unclassified

addFirst: *anElement*

```
self size = capacity
   ifTrue: [self error: 'Colecao no tamanho maximo!'] ifFalse: [ super addFirst: anElement. ]!
```

addLast: *anElement*

```
self size = capacity
   ifTrue: [self error: 'Colecao no tamanho maximo!']
   ifFalse: [super addLast: anElement]!
```

capacity

```
↑ capacity
```

capacity: *aQuantity*

```
capacity ← aQuantity.!
```

insert: *anElement* **before:** *otherElement*

```
self size = capacity
   ifTrue: [self error: 'Colecao no tamanho maximo!']
   ifFalse: [super insert: anElement before: otherElement]!
```

Methods for category: nil

initialize

```
capacity ← 0!
```

Object subclass: OORDBMTiposBasicos
Category: OORDBM

Esta classe gerencia as classes presentes nos diagramas de classes que devem ser consideradas tipos basicos − strings, numeros, datas, etc − dessa forma evita−se que o sistema gere classes e codigo de persistencia para as mesmas. Tambem contem blocos de codigo associados a cada tipo basico, para validacao interna pelos metodos getter/setter das classes.

Class protocol

Methods for category: initialization

initialize

```
tipos ← #('Inteiro' 'Texto' 'Racional' 'Real' 'Data' 'Hora' 'ValorMonetario' 'Imagem' 'SimNao' 'Qualquer' 'Chave' ).
tiposUsuario ← #('Inteiro' 'Texto' 'Racional' 'Real' 'Data' 'Hora' 'ValorMonetario' 'Imagem' 'SimNao' 'Qualquer' ).
blocosValidacao ← Dictionary new.
blocosValidacao at: 'Inteiro' put: '[:valorTMPSET | valorTMPSET asInteger ]'.
blocosValidacao at: 'Texto' put: '[:valorTMPSET | valorTMPSET asString ]'.
blocosValidacao at: 'Racional' put: '[:valorTMPSET | valorTMPSET asFraction ]'.
blocosValidacao at: 'Real' put: '[:valorTMPSET | valorTMPSET asNumber ]'.
blocosValidacao at: 'Decimal' put: '[:valorTMPSET | valorTMPSET asNumber ]'.
blocosValidacao at: 'Data' put: '[:valorTMPSET | valorTMPSET asDate ]'.
blocosValidacao at: 'Hora' put: '[:valorTMPSET | valorTMPSET asTime ] '.
blocosValidacao at: 'ValorMonetario' put: '[:valorTMPSET | valorTMPSET asScaledDecimal: 2 ]'.
blocosValidacao at: 'Imagem' put: '[:valorTMPSET | valorTMPSET isKindOf: Form ]'.
blocosValidacao at: 'Qualquer' put: '[:valorTMPSET | true ]'.
blocosValidacao at: 'SimNao' put: '[:valorTMPSET | valorTMPSET isKindOf: boolean ]'.
blocosSetter ← Dictionary new.
blocosSetter at: 'Inteiro' put: '[:valorTMPSET | valorTMPSET asInteger]'.
blocosSetter at: 'Texto' put: '[:valorTMPSET | valorTMPSET asString]'.
blocosSetter at: 'Racional' put: '[:valorTMPSET | valorTMPSET asFraction]'.
blocosSetter at: 'Real' put: '[:valorTMPSET | valorTMPSET asNumber]'.
blocosSetter at: 'Decimal' put: '[:valorTMPSET | valorTMPSET asNumber]'.
blocosSetter at: 'Data' put: '[:valorTMPSET | valorTMPSET asDate]'.
blocosSetter at: 'Hora' put: '[:valorTMPSET | valorTMPSET asTime]'.
blocosSetter at: 'ValorMonetario' put: '[:valorTMPSET | valorTMPSET asScaledDecimal: 2]'.
blocosSetter at: 'Imagem' put: '[:valorTMPSET || imgTMP | imgTMP ← ReadWriteStream on: #(). valorTMPSET writeOn:
imgTMP. ByteArray withAll: imgTMP contents]'.
blocosSetter at: 'Qualquer' put: '[:valorTMPSET || imgTMP | imgTMP ← ReadWriteStream on: #(). valorTMPSET writeOn:
imgTMP. ByteArray withAll: imgTMP contents]'.
blocosSetter at: 'SimNao' put: '[:valorTMPSET | valorTMPSET]'.
blocosGetter ← Dictionary new.
blocosGetter at: 'Imagem' put: '[:valorTMPSET | Form new readFrom: (ReadStream on: valorTMPSET) reset]'.
blocosGetter at: 'Racional' put: '[:valorTMPSET | valorTMPSET asFraction]'!
```

new

```
↑ super new initialize.
```

tiposUsuario

```
"Retorna os nomes de tipos que podem ser utilizados pelo usuario na
modelagem de um sistema (Inteiro, Texto, Real, Racional, etc.)"
↑ tiposUsuario!
```

Methods for category: resolucao

blocoGetterPara: *umNomeDeTipoBasico*

```
"Dado um nome de um tipo, retorna o codigo responsavel por recuperar o
valor do tipo basico no atributo da classe"
blocosGetter isNil
    ifTrue: [self initialize].
↑ blocosGetter
    at: (blocosGetter keys
        detect: [:key | (RxMatcher forString: key)
            matches: umNomeDeTipoBasico] ifNone: ['none'])
    ifAbsent: ' '!
```

blocoSetterPara: *umNomeDeTipoBasico*

```
"Dado um nome de um tipo, retorna o codigo responsavel por gravar o
valor do tipo basico no atributo da classe − exemplo: para um atributo
do tipo Texto, retorna '← asString'"
blocosSetter isNil
    ifTrue: [self initialize].
```

```
  ↑ blocosSetter
    at: (blocosSetter keys
          detect: [:key | (RxMatcher forString: key)
                matches: umNomeDeTipoBasico])!
```

blocoValidadorPara: *umNomeDeTipoBasico*

```
  "Dado um nome de tipo basico, retorna o codigo para validar seu valor a
  ser incluido no setter do atributo da classe que tem eh deste tipo
  informado – por exemplo, no caso de um tipo booleano, 'isBoolean' – , etc."
  blocosValidacao isNil
    ifTrue: [self initialize].
  ↑ blocosValidacao
    at: (blocosValidacao keys
          detect: [:key | (RxMatcher forString: key)
                matches: umNomeDeTipoBasico])!
```

ehTipoBasico: *umNomeDeClasse*

```
  "Dado um nome de um tipo, verifica se o mesmo eh um tipo basico"
  tipos isNil
    ifTrue: [self initialize].
  ↑ tipos
    anySatisfy: [:tipo | (RxMatcher forString: tipo)
          matches: umNomeDeClasse]!
```

LimitedOrderedCollection subclass: TypedLimitedOrderedCollection

```
    Category: OORDBM
    Instance variable names: tipo
```

Extende uma LimitedOrderedCollection permitindo definir uma classe que deve ser respeitada quando um objeto e inserido, alem de ter uma quantidade maxima de elementos que podem ficar armazenados na mesma.

Class protocol

Methods for category: as yet unclassified

new
```
| retorno |
retorno ← super new.
retorno tipo: Object.
↑ retorno.
```
new: *aValue*
```
| retorno |
retorno ← super new: aValue.
retorno tipo: Object.
↑ retorno!
```

Instance protocol

Methods for category: accessing

tipo
```
"Answer the value of tipo"
↑ tipo
```
tipo: *anObject*
```
"Set the value of tipo"
tipo ← anObject
```

Methods for category: as yet unclassified

addFirst: *anElement*
```
(anElement isKindOf: tipo) ifFalse: [ self error: 'Classe invalida!' ] ifTrue: [ super addFirst: anElement ].!
```

addLast: *anElement*
```
(anElement isKindOf: tipo)
    ifTrue: [super addLast: anElement]
    ifFalse: [self error: 'Classe invalida!']!
```

insert: *anElement* **before:** *anotherElement*
```
(anElement isKindOf: tipo)
    ifTrue: [super insert: anElement before: anotherElement]
    ifFalse: [self error: 'Classe invalida!']!
```

OrderedCollection subclass: TypedOrderedCollection
Category: OORDBM
Instance variable names: tipo

Extende uma LimitedOrderedCollection permitindo definir uma classe que deve ser respeitada quando um objeto e inserido

Class protocol

Methods for category: as yet unclassified

new

```
| retorno |
retorno ← super new.
retorno tipo: Object.
↑ retorno!
```

new: *aValue*

```
| retorno |
retorno ← super new: aValue.
retorno tipo: Object.
↑ retorno!
```

Instance protocol

Methods for category: accessing

tipo

```
"Answer the value of tipo"
↑ tipo
```

tipo: *anObject*

```
"Set the value of tipo"
tipo ← anObject
```

Methods for category: as yet unclassified

addFirst: *anElement*

```
(anElement isKindOf: tipo)
    ifTrue: [super addFirst: anElement]
    ifFalse: [self error: 'Classe invalida!']!
```

addLast: *anElement*

```
(anElement isKindOf: tipo)
    ifTrue: [super addLast: anElement]
    ifFalse: [self error: 'Classe invalida!']!
```

insert: *anElement* **before:** *anotherElement*

```
(anElement isKindOf: tipo)
    ifTrue: [super insert: anElement before: anotherElement]
    ifFalse: [self error: 'Classe invalida!']!
```

Object subclass: ValidadorDeClasses
Category: OORDBM

Esta classe valida as definicoes de classe informadas (nomes de atributos, assinaturas de metodos, nomes de classes, etc.)

Class protocol

Methods for category: as yet unclassified

validarAssinaturaDeMetodo: *umaAssinaturaDeMetodo*

```
| tokens token varsInt |
tokens ← umaAssinaturaDeMetodo findTokens: $ .
tokens isEmpty
   ifTrue: [↑ false].
tokens size = 1
   ifTrue: [tokens last isAllAlphaNumeric not
         ifTrue: [↑ false]
         ifFalse: [↑ tokens last first isUppercase not] ].
tokens size odd
   ifTrue: [↑ false].
varsInt ← Set new.
1
   to: tokens size
   do: [:pos |
      token ← tokens at: pos.
      pos even
         ifTrue: [(self validarAtributo: token)
               ifFalse: [↑ false]. (varsInt includes: token) ifTrue: [↑ false]. varsInt add: token.]
         ifFalse: [(token first: token size − 1) isAllAlphaNumeric
               ifTrue: [token first isUppercase
                     ifTrue: [↑ false] ]
               ifFalse: [↑ false].
            (tokens at: pos) last = $:
               ifFalse: [↑ false] ]].
↑ true!
```

validarAssinaturasDeMetodos: *umaListaDeAssinaturasDeMetodos*

```
↑ umaListaDeAssinaturasDeMetodos
   allSatisfy: [:metodo | self validarAssinaturaDeMetodo: metodo].
!
```

validarAtributo: *umAtributo*

```
umAtributo withBlanksTrimmed size = 0
   ifTrue: [↑ false].
↑ umAtributo isAllAlphaNumeric
   and: [umAtributo first isUppercase not]!
```

validarAtributos: *umCollectionDeAtributos*

```
↑ umCollectionDeAtributos allSatisfy: [:atributo | self validarAtributo: atributo. ].
```

validarCategoria: *umaCategoria*

```
umaCategoria withBlanksTrimmed size = 0
   ifTrue: [↑ false].
umaCategoria
   do: [:car | (car isAlphaNumeric
         or: [car = $−])
      ifFalse: [↑ false] ].
↑ umaCategoria first isUppercase.!
```

validarNome: *umNome*

```
umNome withBlanksTrimmed isEmpty ifTrue: [ ↑ false. ].
↑ (umNome isAllAlphaNumeric) and: [umNome first isUppercase].
```

validarNomes: *listaDeNomes*

```
↑ (listaDeNomes anySatisfy: [:nome | (self validarNome: nome) not]) not

  immed isEmpty ifTrue: [ ↑ false. ].
↑ (umNome isAllAlphaNumeric) and: [umNome first isUppercase].
```

validarNomes: *listaDeNomes*

```
↑ (listaDeNomes anySatisfy: [:nome | (self validarNome: nome) not]) not
```

Object subclass: OORDBMAtributo
Category: OORDBM-Classes
Instance variable names: nome tipo

Esta classe representa um atributo de uma classe (OORDBMClasse)

Instance protocol

Methods for category: accessing

nome
↑ nome.
nome: *umNome*
nome ← umNome.
tipo
↑ tipo.
tipo: *umTipo*
"Tipo deste atritbuto. Pode ser tanto um literal → caso de atributos simples
− quanto uma instancia de OORDBMClasse"
tipo ← umTipo !

Methods for category: printing

OORDBMAtributo subclass: OORDBMAtributoDeColecao
Category: OORDBM-Classes
Instance variable names: maximo

Representa um atributo que pode conter n−valores

Class protocol

Methods for category: as yet unclassified

new
↑ super new initalize.

Instance protocol

Methods for category: accessing

maximo
↑ maximo!

maximo: *umValor*
"Maximo de elementos que podem participar da relacao. Se umValor $= -1$,
assume−se que o maximo eh o infinito"
maximo ← umValor!

Methods for category: initialization

initialize
maximo ← -1.

Object subclass: OORDBMClasse
Category: OORDBM-Classes
Instance variable names: nome atributos abstrata metodos

Esta classe representa uma classe em uma hierarquia de classes (OORDBMHierarquiaDeClasses)

Class protocol

Methods for category: as yet unclassified

new
↑ super new initialize.

Instance protocol

Methods for category: accessing

abstrata
↑ abstrata.
abstrata: *umBoolean*
abstrata ← umBoolean
adicionaAtributo: *umAtributo*
"Recebe um OORDBMAtributo ou descendentes do mesmo"
atributos at: umAtributo nome put: umAtributo!

atributoNomeado: *umNome*
↑ atributos at: umNome.!

atributos
↑ atributos values.
metodos
↑ metodos
nome
↑ nome.
nome: *umNome*
nome ← umNome.
nomesDosAtributos
↑ atributos keys.

Methods for category: as yet unclassified

Methods for category: initialization

initialize
atributos ← Dictionary new.
metodos ← OrderedCollection new.!

Methods for category: printing

Object subclass: OORDBMHeranca
Category: OORDBM-Classes
Instance variable names: classeSuperior classeDescendente

Representa uma heranca entre classes

Class protocol

Methods for category: as yet unclassified

new: *umaClasse* **a:** *outraClasse*
↑ (super new classeSuperior: outraClasse)
classeDescendente: umaClasse!

Instance protocol

Methods for category: accessing

classeDescendente
↑ classeDescendente!

classeDescendente: *umOORDBMClasse*
classeDescendente ← umOORDBMClasse!

classeSuperior
↑ classeSuperior!

classeSuperior: *umOORDBMClasse*
classeSuperior ← umOORDBMClasse!

Methods for category: as yet unclassified

Object subclass: OORDBMHierarquiaDeClasses
Category: OORDBM-Classes
Instance variable names: grafo prefixo categoria

Esta classe representa uma hierarquia de classes em Smalltalk.
Categoria define a qual categoria as classes pertencem.

Class protocol

Methods for category: as yet unclassified

new
```
↑ super new initialize.
```

Instance protocol

Methods for category: accessing

adicionaClasse: *umaClasse*
```
grafo adicionaNodo: umaClasse.
```
categoria
```
"Categoria (segundo o conceito dentro do Smalltalk) a que pertencem as classes dessa hierarquia"
↑ categoria isNil
   ifTrue: [↑ ' ']
   ifFalse: [categoria]!
```

categoria: *umaCategoria*
```
categoria ← umaCategoria.
```
classeMaisSuperiorDe: *umaEntidade*
```
"Retorna a entidade do grafo que representa a classe mais superior a informa na hierarquia"
| entidade |
entidade ← umaEntidade.
[(self classePai: entidade) isNil] whileFalse: [
   entidade ← self classePai: entidade.
].
↑ entidade.
```
classeNomeada: *umNome*
```
grafo nodos do: [:nodo | nodo nome = umNome ifTrue: [ ↑ nodo ]. ].
↑ nil.
```
classePai: *umaEntidade*
```
| heranca |
heranca ← (grafo arestasChegandoA: umaEntidade).
heranca isNil ifTrue: [ ↑ nil. ].
heranca ← (heranca detect: [ :aresta | aresta rotulo isKindOf: OORDBMHeranca ] ifNone: [nil]).
heranca isNil ifTrue: [ ↑ nil. ] ifFalse: [ ↑ heranca nodoA. ].
```

classes
```
↑ grafo nodos.
```
classesDescendentes: *umaClasse*
```
"Retorna todas as classes que estao abaixo da informada na hierarquia"
| retorno entidades filhas |
entidades ← OrderedCollection with: umaClasse.
retorno ← OrderedCollection new.
[entidades isEmpty]
   whileFalse: [filhas ← self classesFilhas: entidades removeFirst.
      retorno addAll: filhas.
      entidades addAll: filhas].
↑ retorno!
```

classesFilhas: *umaEntidade*
```
↑ (grafo arestasPartindoDe: umaEntidade) select: [:aresta | aresta rotulo isKindOf: OORDBMHeranca ] thenCollect: [:aresta | aresta nodoB ].
```

classesSuperiores
```
"Retorna as classes do grafo que nao possuem uma classe superior presente no mesmo"
↑ (self classes) select: [ :entidade |
   ((grafo arestasChegandoA: entidade) select: [:aresta | aresta rotulo isKindOf: OORDBMHeranca]) isEmpty.
].
```
classesSuperioresDe: *umaEntidade*
```
"Retorna as entidades do grafo que sao superiores na heranca a informada"
| entidades retorno pai |
entidades ← OrderedCollection with: umaEntidade.
retorno ← OrderedCollection new.
```

```
    [entidades isEmpty] whileFalse: [
        pai ← self classePai: (entidades removeFirst).
        pai isNil ifFalse: [
            retorno add: pai.
            entidades add: pai.
        ].
    ].
    ↑ retorno.
```

estabelecaHerancaDe: *umaClasse* **para:** *outraClasse*

```
    (grafo arestasChegandoA: outraClasse)
        do: [:aresta | (aresta rotulo isKindOf: OORDBMHeranca)
                ifTrue: [grafo removeAresta: aresta] ].
    grafo
        conecta: umaClasse
        a: outraClasse
        rotulo: (OORDBMHeranca new: outraClasse a: umaClasse)!
```

expandaAtributosParaFilhos

```
    "Para cada classe com subclasses, coloca os atributos constantes na
    mesma em todas as suas subclasses"
    | classesSuperiores filhos novoGrafo novaHierarquia |
    novoGrafo ← grafo veryDeepCopy.
novaHierarquia ← self shallowCopy grafo: novoGrafo.
    classesSuperiores ← OrderedCollection withAll: novaHierarquia classesSuperiores.
    classesSuperiores
        do: [:classe |
            filhos ← novoGrafo arestasPartindoDe: classe.
            filhos isNil
                ifFalse: [
                    filhos ← filhos
                            select: [:aresta | aresta rotulo isKindOf: OORDBMHeranca].
                    filhos
                        do: [:filho | classe atributos
                                do: [:atributo |
                                    filho nodoB adicionaAtributo: atributo.
                                    classesSuperiores addLast: filho nodoB] ]] ].
    ↑ novaHierarquia!
```

grafo

```
    ↑grafo
```

grafo: *umGrafo*

```
    grafo ← umGrafo.
```

herancas

```
    "Retornas as herancas existentes no grafo (adjacencias)"
    ↑ grafo arestas
        select: [:aresta | aresta rotulo isKindOf: OORDBMHeranca]
        thenCollect: [:heranca | heranca rotulo]!
```

relacionamentos

```
    ↑ grafo arestas
        select: [:aresta | (aresta rotulo isKindOf: OORDBMHeranca) not]
        thenCollect: [:aresta | aresta rotulo]!
```

relacionamentosDe: *umaEntidade*

```
    | arestas relacionamentos |
    arestas ← grafo arestas
            select: [:aresta | (aresta nodoA = umaEntidade
                    or: [aresta nodoB = umaEntidade])
                and: [(aresta rotulo isKindOf: OORDBMHeranca) not] ].
    relacionamentos ← arestas
            collect: [:aresta | aresta rotulo].
    ↑ relacionamentos!
```

relacionamentosIncluindoDescendentesDe: *umaEntidade*

```
    | arestas relacionamentos |
    arestas ← grafo arestas
            select: [:aresta | (aresta nodoA = umaEntidade
```

```
                or: [aresta nodoB = umaEntidade])
            and: [(aresta rotulo isKindOf: OORDBMHeranca) not] ].
  (self classesDescendentes: umaEntidade)
    do: [:classe | arestas
        addAll: (grafo arestas
            select: [:aresta | (aresta nodoA = classe
                or: [aresta nodoB = classe])
            and: [(aresta rotulo isKindOf: OORDBMHeranca) not] ])].
  relacionamentos ← arestas
        collect: [:aresta | aresta rotulo].
  ↑ relacionamentos!
```

relacione: *umaEntidade* **a:** *outraEntidade* **segundo:** *umRelacionamento*

```
  grafo
    conecta: umaEntidade
    a: outraEntidade
    rotulo: umRelacionamento
```

Methods for category: initialization

initialize

```
  grafo ← Digrafo new.
  grafo admiteArestasMultiplas: true.
```

Methods for category: private

OORDBMHierarquiaDeClasses subclass: OORDBMHierarquiaDeClassesComParserXMI

Category: OORDBM-Classes

Inclui um parser de XMIDocumento

Instance protocol

Methods for category: nil

parseXMIAssociacoes: *umDocumentoXMI*

```
| multipOrigem multipDestino dest1 orig1 destN origN entOrig entDest entN ent1 multN role1 roleN |
umDocumentoXMI associacoes
  do: [:associacao |
    multipOrigem ← associacao multiplicidadeOrigem superior asNumber.
    multipDestino ← associacao multiplicidadeDestino superior asNumber.
    orig1 ← multipOrigem = 1.
    dest1 ← multipDestino = 1.
    origN ← multipOrigem ≠ 1.
    destN ← multipDestino ≠ 1.
    entOrig ← self classeNomeada: associacao classeOrigem nome.
    entDest ← self classeNomeada: associacao classeDestino nome.
    “1 a 1"
    (orig1
        and: [dest1])
      ifTrue: [self
          relacione: entOrig
          a: entDest
          segundo: ((OORDBMRelacionamento1a1 new: entOrig a: entDest) papelOrigem: associacao papelOrigem;
              papelDestino: associacao papelDestino)].
    “1 a N"
    ((dest1
          and: [origN])
        or: [destN
            and: [orig1] ])
      ifTrue: [destN
          ifTrue: [ent1 ← entOrig.
            entN ← entDest.
            multN ← multipDestino.
            role1 ← associacao papelOrigem.
            roleN ← associacao papelDestino]
          ifFalse: [entN ← entOrig.
            ent1 ← entDest.
            multN ← multipOrigem.
            role1 ← associacao papelDestino.
            roleN ← associacao papelOrigem].
        self
          relacione: entN
          a: ent1
          segundo: ((OORDBMRelacionamento1aN
              new: entN
              a: ent1
              grau: multN) papelOrigem: roleN;
              papelDestino: role1)].
    “N a N"
    (destN
        and: [origN])
      ifTrue: [self
          relacione: entOrig
          a: entDest
          segundo: ((OORDBMRelacionamentoNaN
              new: entOrig
              a: entDest
              grauOrigem: multipOrigem
              grauDestino: multipDestino) papelOrigem: associacao papelOrigem;
              papelDestino: associacao papelDestino)] ]!
```

parseXMIClasses: *umDocumentoXMI*

```
| entidade atributo tipo |
umDocumentoXMI classes
  do: [:classeXMI |
```

```
        entidade ← OORDBMClasse new.
        entidade nome: classeXMI nome.
        entidade abstrata: classeXMI abstrata.
        classeXMI atributos
          do: [:atributoXMI |
            atributo ← OORDBMAtributo new.
            atributo nome: atributoXMI nome.
            atributo tipo: atributoXMI tipo nome.
            entidade adicionaAtributo: atributo].
        classeXMI metodos
          do: [:metodoXMI | entidade metodos add: (OORDBMMetodo new assinatura: (metodoXMI assinatura))].
        self adicionaClasse: entidade].
    umDocumentoXMI tiposDeDados
      do: [:classeXMI | (self classeNomeada: classeXMI nome) isNil
          ifTrue: [entidade ← OORDBMClasse new.
            entidade nome: classeXMI nome.
            entidade abstrata: false.
            self adicionaClasse: entidade] ].
    grafo nodos
      do: [:nodoEntidade | nodoEntidade atributos
          do: [:nodoAtributo |
            tipo ← self classeNomeada: nodoAtributo tipo.
            tipo isNil
              ifFalse: [nodoAtributo tipo: tipo] ]]!
```

parseXMIHerancas: *umDocumentoXMI*

```
    umDocumentoXMI herancas
      do: [:heranca | self
          estabelecaHerancaDe: (self classeNomeada: heranca classeSuperior nome)
          para: (self classeNomeada: heranca classeDescendente nome)]!
```

xmi: *umDocumentoXMI*

```
    self initialize.
    self parseXMIClasses: umDocumentoXMI.
    self parseXMIAssociacoes: umDocumentoXMI.
    self parseXMIHerancas: umDocumentoXMI.
!
```

Object subclass: OORDBMMetodo

Category: OORDBM-Classes

Instance variable names: assinatura

Representa um metodo de uma classe (OORDBMHierarquiaDeClasses)

Instance protocol

Methods for category: accessing

assinatura

```
    ↑ assinatura!
```

assinatura: *umTextoDescrevendoAAssinaturaDoMetodo*

```
    assinatura ← umTextoDescrevendoAAssinaturaDoMetodo!
```

Object subclass: OORDBMRelacionamento

Category: OORDBM-Classes

Instance variable names: entidadeA entidadeB nomeAtributoLadoA nomeAtributoLadoB

Esta classe representa um relacionamento na hierarquia de classes (OORDBMHierarquiaDeClasses).

Pode−se definir os participantes tanto em um esquema direcionado (origem/destino) quanto adirecional(entidade do LadoA e entidade do LadoB)

Class protocol

Methods for category: creation

new: *umaClasse* **a:** *outraClass*
↑ super new initialize: umaClasse a: outraClass!

new: *umaClasse* **a:** *outraClasse* **papelOrigem:** *papelOrigem* **papelDestino:** *papelDestino*
↑ super new
initialize: umaClasse
a: outraClasse
papelOrigem: papelOrigem
papelDestino: papelDestino!

Instance protocol

Methods for category: accessing

classeDestino
↑ entidadeB

classeLadoA
↑ entidadeA!

classeLadoB
↑ entidadeB!

classeOrigem
↑ entidadeA

grauDestino
↑ self grauLadoB!

grauLadoA
↑ self subclassResponsibility!

grauLadoB
↑ self subclassResponsibility!

grauOrigem
↑ self grauLadoA!

nomeAtributoDestino
↑ self nomeAtributoLadoB

nomeAtributoDestino: *umNome*
↑ self nomeAtributoLadoB: umNome!

nomeAtributoLadoA
nomeAtributoLadoA isNil
ifTrue: [↑ ' ']
ifFalse: [↑ nomeAtributoLadoA]!

nomeAtributoLadoA: *umNome*
nomeAtributoLadoA ← umNome!

nomeAtributoLadoB
nomeAtributoLadoB isNil
ifTrue: [↑ ' ']
ifFalse: [↑ nomeAtributoLadoB]!

nomeAtributoLadoB: *umNome*
nomeAtributoLadoB ← umNome!

nomeAtributoOrigem
↑ self nomeAtributoLadoA

nomeAtributoOrigem: *umNome*

↑ self nomeAtributoLadoA: umNome!

papelDestino
↑ self nomeAtributoDestino!

papelDestino: *umPapel*
↑ self nomeAtributoDestino: umPapel!

papelOrigem
↑ self nomeAtributoOrigem

papelOrigem: *umPapel*
↑ self nomeAtributoOrigem: umPapel!

Methods for category: initialization

initialize: *umaEntidade* **a:** *outraEntidade*
entidadeA ← umaEntidade.
entidadeB ← outraEntidade.
nomeAtributoLadoA ← ' '.
nomeAtributoLadoB ← ' '.!

initialize: *umaEntidade* **a:** *outraEntidade* **papelOrigem:** *umPapel* **papelDestino:** *outroPapel*
entidadeA ← umaEntidade.
entidadeB ← outraEntidade.
nomeAtributoLadoA ← umPapel.
nomeAtributoLadoB ← outroPapel!

OORDBMRelacionamento subclass: OORDBMRelacionamento1a1
Category: OORDBM-Classes

Esta classe representa um relacionamento na hierarquia de classes (OORDBMHierarquiaDeClasses)

Instance protocol

Methods for category: accessing

grauLadoA
↑ 1!

grauLadoB
↑ 1!

OORDBMRelacionamento subclass: OORDBMRelacionamento1aN
Category: OORDBM-Classes
Instance variable names: n

Esta classe representa um relacionamento na hierarquia de classes (OORDBMHierarquiaDeClasses)

Class protocol

Methods for category: as yet unclassified

new: *umaClasse* **a:** *outraClasse* **grau:** *umNumero*

```
↑ super new
   initialize: umaClasse
   a: outraClasse
   grau: umNumero!
```

Methods for category: nil

Instance protocol

Methods for category: accessing

grauLadoA

```
↑ 1!
```

grauLadoB

```
↑ n!
```

Methods for category: initialization

initialize: *umaEntidade* **a:** *outraEntidade*

```
super initialize: umaEntidade a: outraEntidade.
n ← −1.!
```

initialize: *umaEntidade* **a:** *outraEntidade* **grau:** *umGrau*

```
super initialize: umaEntidade a: outraEntidade.
n ← umGrau!
```

OORDBMRelacionamento subclass: OORDBMRelacionamentoNaN

Category: OORDBM-Classes
Instance variable names: nOrigem nDestino

Esta classe representa um relacionamento na hierarquia de classes (OORDBMHierarquiaDeClasses)

Class protocol

Methods for category: as yet unclassified

new: *umaEntidade* **a:** *outraEntidade* **grauOrigem:** *umNumero* **grauDestino:** *outroNumero*

```
↑ super new
  initialize: umaEntidade
  a: outraEntidade
  grauOrigem: umNumero
  grauDestino: outroNumero!
```

Instance protocol

Methods for category: accessing

grauLadoA

```
↑ nOrigem!
```

grauLadoB

```
↑ nDestino!
```

Methods for category: as yet unclassified

initialize: *umaEntidade* **a:** *outraEntidade* **grauOrigem:** *umNumero* **grauDestino:** *outroNumero*

```
self initialize: umaEntidade a: outraEntidade.
nOrigem ← umNumero.
nDestino ← outroNumero!
```

HerancaComParserXMI
Category: OORDBM-Classes
Instance variable names: xmi hierarquia

Testa o sistema de importacao da classe OORDBMHierarquiaDeClassesComParserXMI

Instance protocol

Methods for category: Running

Methods for category: as yet unclassified

testValidarAtributosClasses

```
xmi classes
   do: [:classeXMI |
      | classe atributosXMI atributos |
      classe ← hierarquia classeNomeada: classeXMI nome.
      atributos ← classe nomesDosAtributos.
      atributosXMI ← classeXMI atributos
            collect: [:atributo | atributo nome].
      self assert: (atributos difference: atributosXMI) isEmpty.
      self assert: (atributosXMI difference: atributos) isEmpty]!
```

testValidarClassesCriadas

```
   | classesHierarquia classesXMI |
   classesXMI ← (OrderedCollection withAll: (xmi classes collect: [:classe | classe nome])) addAll: (xmi tiposDeDados collect: [:classe | classe nome]).
   classesHierarquia ← OrderedCollection withAll: (hierarquia classes collect: [:classe | classe nome]).
   self assert: ( (classesXMI difference: classesHierarquia) isEmpty ).
```

testValidarHerancas

```
   | herancas herancasXMI |
   herancas ← hierarquia herancas.
   herancasXMI ← xmi herancas.
   herancasXMI do: [:herancaXMI |
      self deny: (herancas detect: [:heranca | heranca classeSuperior nome = herancaXMI classeSuperior nome and: [heranca classe-Descendente nome = herancaXMI classeDescendente nome] ] ifNone: [nil]) isNil.
   ].
   herancas do: [:heranca |
      self deny: (herancasXMI detect: [:herancaXMI | heranca classeSuperior nome = herancaXMI classeSuperior nome and: [heranca classeDescendente nome = herancaXMI classeDescendente nome] ] ifNone: [nil]) isNil.
   ].
!
```

testValidarMetodosClasses

```
   xmi classes
      do: [:classeXMI |
         | classe metodosXMI metodos |
         classe ← hierarquia classeNomeada: classeXMI nome.
         metodos ← classe metodos collect: [:metodo | metodo assinatura].
         metodosXMI ← classeXMI metodos.
         self assert: (metodos difference: metodosXMI) isEmpty.
         self assert: (metodosXMI difference: metodos) isEmpty]!
```

testValidarRelacionamentos

```
   | associacoes associacoesXMI |
   associacoes ← hierarquia relacionamentos.
   associacoesXMI ← xmi associacoes.
   associacoesXMI
      do: [:associacaoXMI | self deny: (associacoes
            detect: [:associacao | associacaoXMI classeOrigem nome = associacao classeOrigem nome
                  and: [associacaoXMI classeDestino nome = associacao classeDestino nome
                        and: [associacaoXMI papelOrigem = associacao papelOrigem
                              and: [(associacaoXMI papelDestino = associacao papelDestino
                                    and: [associacaoXMI multiplicidadeOrigem superior asNumber = associacao grauOrigem
                                          and: [associacaoXMI multiplicidadeDestino superior asNumber = associacao grauDestino] ])
or: [(associacaoXMI papelDestino = associacao papelOrigem
                                    and: [associacaoXMI multiplicidadeOrigem superior asNumber = associacao grauDestino
```

```
                                        and: [associacaoXMI multiplicidadeDestino superior asNumber = associacao grauOrigem] ])]
] ]] ]
            ifNone: [ ]) isNil].
   associacoes
      do: [:associacao | self deny: (associacoesXMI
            detect: [:associacaoXMI | associacaoXMI classeOrigem nome = associacao classeOrigem nome
                  and: [associacaoXMI classeDestino nome = associacao classeDestino nome
                        and: [associacaoXMI papelOrigem = associacao papelOrigem
                              and: [(associacaoXMI papelDestino = associacao papelDestino
                                    and: [associacaoXMI multiplicidadeOrigem superior asNumber = associacao grauOrigem
                                          and: [associacaoXMI multiplicidadeDestino superior asNumber = associacao grauDestino] ])
or: [(associacaoXMI papelDestino = associacao papelOrigem
                                    and: [associacaoXMI multiplicidadeOrigem superior asNumber = associacao grauDestino
                                          and: [associacaoXMI multiplicidadeDestino superior asNumber = associacao grauOrigem] ])]
] ]] ]
            ifNone: [ ]) isNil].!
```

testValidarTiposAtributosClasses

```
   xmi classes
      do: [:classeXMI |
         | classe |
         classe ← hierarquia classeNomeada: classeXMI nome.
         classeXMI atributos do: [:atributo |
            self assert: (atributo tipo nome = ((classe atributoNomeado: (atributo nome)) tipo) nome).
         ].
].
         !
```

Methods for category: nil

setUp

```
   xmi ← XMIParserTogether new.
   xmi
      stream: (ReadStream on: '<?xml version = ' '1.0' ' encoding = ' 'ISO8859←1' ' ?>
<XMI xmi.version = ' '1.1' ' xmlns:UML = ' '//org.omg/UML/1.3' '>
<XMI.header>
<XMI.documentation>
<XMI.exporter>
   TogetherSoft
</XMI.exporter>
<XMI.exporterVersion>
   6.0
</XMI.exporterVersion>
</XMI.documentation>
<XMI.metamodel xmi.name = ' 'UML' ' xmi.version = ' '1.4' '/>
</XMI.header>
<XMI.content>
<UML:Model xmi.id = ' 'S.1' ' name = ' 'Project' ' visibility = ' 'public' '>
<UML:Namespace.ownedElement>
<UML:Class xmi.id = ' 'S.4' '
name = ' 'Pessoa' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isAbstract = ' 'false' ' isActive = ' 'false' '>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ' 'S.5' '
name = ' 'nome' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
```

<UML:DataType xmi.idref = ’ ’G.1’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.6’ ’
name = ’ ’dataNascimento’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.2’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.7’ ’
name = ’ ’lnkDocumentacao’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.3’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
</UML:Class>
<UML:Class xmi.id = ’ ’S.8’ ’
name = ’ ’Motorista’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ’ ’S.9’ ’
name = ’ ’lnkAutomovel’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.5’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.10’ ’
name = ’ ’carteira’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’

changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.7’ ’
name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.8’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.4’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ’ ’S.11’ ’
name = ’ ’Automovel’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ’ ’S.12’ ’
name = ’ ’lnkFabricante’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.9’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.13’ ’
name = ’ ’ano’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.14’ ’
name = ’ ’ano’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.15’ ’
name = ’ ’ano’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.8’ ’
name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.9’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.4’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ’ ’S.16’ ’
name = ’ ’Fabricante’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ’ ’S.17’ ’
name = ’ ’lnkAutomovel’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.9’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.18’ ’
name = ’ ’ano’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.9’ ’
name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.10’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.4’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ’ ’S.19’ ’
name = ’ ’Motor’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ’ ’S.20’ ’
name = ’ ’lnkAutomovel’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.9’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.21’ ’
name = ’ ’ano’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.10’ ’
name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.11’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.4’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ’ ’S.22’ ’
name = ’ ’Pneu’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ’ ’S.23’ ’
name = ’ ’lnkAutomovel’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.9’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.24’ ’
name = ’ ’ano’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.11’ ’
name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.12’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.4’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ’ ’S.25’ ’
name = ’ ’Volante’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ’ ’S.26’ ’
name = ’ ’lnkAutomovel’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.9’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.27’ ’
name = ’ ’ano’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.12’ ’
name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.13’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.4’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ’ ’S.28’ ’
name = ’ ’Direcao’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ’ ’S.29’ ’
name = ’ ’lnkAutomovel’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.9’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.30’ ’
name = ’ ’ano’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.13’ ’
name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.14’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.4’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ’ ’S.31’ ’
name = ’ ’Roda’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ’ ’S.32’ ’
name = ’ ’lnkAutomovel’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.9’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.33’ ’
name = ’ ’ano’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.14’ ’
name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.15’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.4’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
</UML:Namespace.ownedElement>
</UML:Model>
</XMI.content>
</XMI>

</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ' 'S.14' '
name = ' 'combustivel' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.1' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ' 'S.15' '
name = ' 'chassis' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.1' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ' 'S.16' '
name = ' 'valor' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.10' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
</UML:Class>
<UML:Class xmi.id = ' 'S.17' '
name = ' 'Carro' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isAbstract = ' 'false' ' isActive = ' 'false' '>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ' 'S.18' '
name = ' 'lnkModelosDeCarro' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>

<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.12' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ' 'S.19' '
name = ' 'numPortas' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.6' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ' 'G.13' '
name = ' '" ' visibility = ' 'public' ' isSpecification = ' 'false' '
discriminator = ' '" '>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ' 'S.17' '/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ' 'S.11' '/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ' 'S.20' '
name = ' 'Moto' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isAbstract = ' 'false' ' isActive = ' 'false' '>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ' 'S.21' '
name = ' 'lnkModelosDeMoto' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.15' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>

```
</UML:Attribute>
<UML:Attribute xmi.id = ' 'S.22' '
name = ' 'cilindradas' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.6' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ' 'G.16' '
name = ' '" ' visibility = ' 'public' ' isSpecification = ' 'false' '
discriminator = ' '" '>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ' 'S.20' '/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ' 'S.11' '/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ' 'S.23' '
name = ' 'Caminhao' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isAbstract = ' 'false' ' isActive = ' 'false' '>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ' 'S.24' '
name = ' 'lnkModelosDeCaminhao' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.18' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ' 'S.25' '
name = ' 'capacidadeEmToneladas' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
```

<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ’ ’S.26’ ’
name = ’ ’numeroDeRodas’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.6’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.19’ ’
name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.23’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.11’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ’ ’S.27’ ’
name = ’ ’Fabricante’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ’ ’S.28’ ’
name = ’ ’nome’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.1’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
</UML:Class>
<UML:Class xmi.id = ’ ’S.29’ ’
name = ’ ’ModelosDeCarro’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.20’ ’

name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.29’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.30’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ’ ’S.31’ ’
name = ’ ’ModelosDeCaminhao’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.21’ ’
name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.31’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.30’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ’ ’S.32’ ’
name = ’ ’ModelosDeMoto’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Namespace.ownedElement>
<UML:Generalization xmi.id = ’ ’G.22’ ’
name = ’ ” ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
discriminator = ’ ” ’>
<UML:Generalization.child>
<UML:GeneralizableElement xmi.idref = ’ ’S.32’ ’/>
</UML:Generalization.child>
<UML:Generalization.parent>
<UML:GeneralizableElement xmi.idref = ’ ’S.30’ ’/>
</UML:Generalization.parent>
</UML:Generalization>
</UML:Namespace.ownedElement>
</UML:Class>
<UML:Class xmi.id = ’ ’S.30’ ’
name = ’ ’ModelosDeAutomoveis’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’ isActive = ’ ’false’ ’>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ’ ’S.33’ ’
name = ’ ’nome’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
changeability = ’ ’changeable’ ’ ownerScope = ’ ’instance’ ’>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ’ ’G.1’ ’/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
</UML:Class>

<UML:Class xmi.id = ' 'S.34' '
name = ' 'Documentacao' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isAbstract = ' 'false' ' isActive = ' 'false' '>
<UML:Classifier.feature>
<UML:Attribute xmi.id = ' 'S.35' '
name = ' 'CPF' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.1' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
<UML:Attribute xmi.id = ' 'S.36' '
name = ' 'RG' ' visibility = ' 'private' ' isSpecification = ' 'false' '
changeability = ' 'changeable' ' ownerScope = ' 'instance' '>
<UML:StructuralFeature.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:StructuralFeature.multiplicity>
<UML:StructuralFeature.type>
<UML:Classifier>
<UML:Namespace.ownedElement>
<UML:DataType xmi.idref = ' 'G.1' '/>
</UML:Namespace.ownedElement>
</UML:Classifier>
</UML:StructuralFeature.type>
</UML:Attribute>
</UML:Classifier.feature>
</UML:Class>
<!−−From Class Pessoa to Class Documentacao−−→
<UML:Association xmi.id = ' 'G.0' '
name = ' '{Pessoa−Documentacao}' ' visibility = ' 'private' ' isSpecification = ' 'false' '
isAbstract = ' 'false' '>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = ' 'G.23' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isNavigable = ' 'true' ' ordering = ' 'unordered' ' aggregation = ' 'none' '
targetScope = ' 'instance' ' changeability = ' 'changeable' '>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ' 'S.4' '/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id = ' 'G.24' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isNavigable = ' 'true' ' ordering = ' 'unordered' ' aggregation = ' 'none' '
targetScope = ' 'instance' ' changeability = ' 'changeable' '>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>

<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ’ ’S.34’ ’/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<!−−From Class Motorista to Class Automovel−−→
<UML:Association xmi.id = ’ ’G.4’ ’
name = ’ ’{Motorista−Automovel}’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = ’ ’G.25’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isNavigable = ’ ’true’ ’ ordering = ’ ’unordered’ ’ aggregation = ’ ’none’ ’
targetScope = ’ ’instance’ ’ changeability = ’ ’changeable’ ’>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’0’ ’ upper = ’ ’−1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ’ ’S.8’ ’/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id = ’ ’G.26’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isNavigable = ’ ’true’ ’ ordering = ’ ’unordered’ ’ aggregation = ’ ’none’ ’
targetScope = ’ ’instance’ ’ changeability = ’ ’changeable’ ’>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’0’ ’ upper = ’ ’−1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ’ ’S.11’ ’/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<!−−From Class Automovel to Class Fabricante−−→
<UML:Association xmi.id = ’ ’G.8’ ’
name = ’ ’{Automovel−Fabricante}’ ’ visibility = ’ ’private’ ’ isSpecification = ’ ’false’ ’
isAbstract = ’ ’false’ ’>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = ’ ’G.27’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isNavigable = ’ ’true’ ’ ordering = ’ ’unordered’ ’ aggregation = ’ ’none’ ’
targetScope = ’ ’instance’ ’ changeability = ’ ’changeable’ ’>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ’ ’1’ ’ upper = ’ ’−1’ ’/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ’ ’S.11’ ’/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id = ’ ’G.28’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’
isNavigable = ’ ’true’ ’ ordering = ’ ’unordered’ ’ aggregation = ’ ’none’ ’
targetScope = ’ ’instance’ ’ changeability = ’ ’changeable’ ’>

<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ' 'S.27' '/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<!−−From Class Carro to Class ModelosDeCarro−−→
<UML:Association xmi.id = ' 'G.11' '
name = ' '{Carro−ModelosDeCarro}' ' visibility = ' 'private' ' isSpecification = ' 'false' '
isAbstract = ' 'false' '>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = ' 'G.29' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isNavigable = ' 'true' ' ordering = ' 'unordered' ' aggregation = ' 'none' '
targetScope = ' 'instance' ' changeability = ' 'changeable' '>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '−1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ' 'S.17' '/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id = ' 'G.30' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isNavigable = ' 'true' ' ordering = ' 'unordered' ' aggregation = ' 'none' '
targetScope = ' 'instance' ' changeability = ' 'changeable' '>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ' 'S.29' '/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<!−−From Class Moto to Class ModelosDeMoto−−→
<UML:Association xmi.id = ' 'G.14' '
name = ' '{Moto−ModelosDeMoto}' ' visibility = ' 'private' ' isSpecification = ' 'false' '
isAbstract = ' 'false' '>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = ' 'G.31' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isNavigable = ' 'true' ' ordering = ' 'unordered' ' aggregation = ' 'none' '
targetScope = ' 'instance' ' changeability = ' 'changeable' '>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '−1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ' 'S.20' '/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>

<UML:AssociationEnd xmi.id = ' 'G.32' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isNavigable = ' 'true' ' ordering = ' 'unordered' ' aggregation = ' 'none' '
targetScope = ' 'instance' ' changeability = ' 'changeable' '>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ' 'S.32' '/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<!−−From Class Caminhao to Class ModelosDeCaminhao−→
<UML:Association xmi.id = ' 'G.17' '
name = ' '{Caminhao−ModelosDeCaminhao}' ' visibility = ' 'private' ' isSpecification = ' 'false' '
isAbstract = ' 'false' '>
<UML:Association.connection>
<UML:AssociationEnd xmi.id = ' 'G.33' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isNavigable = ' 'true' ' ordering = ' 'unordered' ' aggregation = ' 'none' '
targetScope = ' 'instance' ' changeability = ' 'changeable' '>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '−1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ' 'S.23' '/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
<UML:AssociationEnd xmi.id = ' 'G.34' ' visibility = ' 'public' ' isSpecification = ' 'false' '
isNavigable = ' 'true' ' ordering = ' 'unordered' ' aggregation = ' 'none' '
targetScope = ' 'instance' ' changeability = ' 'changeable' '>
<UML:AssociationEnd.multiplicity>
<UML:Multiplicity>
<UML:Multiplicity.range>
<UML:MultiplicityRange lower = ' '1' ' upper = ' '1' '/>
</UML:Multiplicity.range>
</UML:Multiplicity>
</UML:AssociationEnd.multiplicity>
<UML:AssociationEnd.participant>
<UML:Classifier xmi.idref = ' 'S.31' '/>
</UML:AssociationEnd.participant>
</UML:AssociationEnd>
</UML:Association.connection>
</UML:Association>
<UML:DataType xmi.id = ' 'G.1' '
name = ' 'Texto' ' visibility = ' 'public' ' isSpecification = ' 'false' '/>
<UML:DataType xmi.id = ' 'G.2' '
name = ' 'Data' ' visibility = ' 'public' ' isSpecification = ' 'false' '/>
<UML:DataType xmi.id = ' 'G.3' '
name = ' 'Documentacao' ' visibility = ' 'public' ' isSpecification = ' 'false' '/>
<UML:DataType xmi.id = ' 'G.5' '
name = ' 'Automovel' ' visibility = ' 'public' ' isSpecification = ' 'false' '/>
<UML:DataType xmi.id = ' 'G.6' '
name = ' 'Inteiro' ' visibility = ' 'public' ' isSpecification = ' 'false' '/>
<UML:DataType xmi.id = ' 'G.9' '
name = ' 'Fabricante' ' visibility = ' 'public' ' isSpecification = ' 'false' '/>
<UML:DataType xmi.id = ' 'G.10' '
name = ' 'Valor' ' visibility = ' 'public' ' isSpecification = ' 'false' '/>
<UML:DataType xmi.id = ' 'G.12' '
name = ' 'ModelosDeCarro' ' visibility = ' 'public' ' isSpecification = ' 'false' '/>

<UML:DataType xmi.id = ’ ’G.15’ ’
name = ’ ’ModelosDeMoto’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’/>
<UML:DataType xmi.id = ’ ’G.18’ ’
name = ’ ’ModelosDeCaminhao’ ’ visibility = ’ ’public’ ’ isSpecification = ’ ’false’ ’/>
</UML:Namespace.ownedElement>
</UML:Model>
<UML:TaggedValue xmi.id = ’ ’XX.1’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’S.7’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.2’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’S.7’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.4’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’S.9’ ’>
<UML:TaggedValue.dataValue>
0..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.5’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’S.9’ ’>
<UML:TaggedValue.dataValue>
0..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.7’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’S.12’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.8’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’S.12’ ’>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.10’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’S.18’ ’>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.11’ ’
name = ’ ’clientCardinality’ ’
modelElement = ’ ’S.18’ ’>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ’ ’XX.13’ ’
name = ’ ’supplierCardinality’ ’
modelElement = ’ ’S.21’ ’>
<UML:TaggedValue.dataValue>
1

</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.14' '
name = ' 'clientCardinality' '
modelElement = ' 'S.21' '>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.16' '
name = ' 'supplierCardinality' '
modelElement = ' 'S.24' '>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.17' '
name = ' 'clientCardinality' '
modelElement = ' 'S.24' '>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.27' '
name = ' 'clientCardinality' '
modelElement = ' 'G.0' '>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.28' '
name = ' 'supplierCardinality' '
modelElement = ' 'G.0' '>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.32' '
name = ' 'clientCardinality' '
modelElement = ' 'G.4' '>
<UML:TaggedValue.dataValue>
0..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.33' '
name = ' 'supplierCardinality' '
modelElement = ' 'G.4' '>
<UML:TaggedValue.dataValue>
0..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.37' '
name = ' 'supplierCardinality' '
modelElement = ' 'G.8' '>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.38' '
name = ' 'clientCardinality' '
modelElement = ' 'G.8' '>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.42' '
name = ' 'supplierCardinality' '

modelElement = ' 'G.11' '>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.43' '
name = ' 'clientCardinality' '
modelElement = ' 'G.11' '>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.47' '
name = ' 'supplierCardinality' '
modelElement = ' 'G.14' '>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.48' '
name = ' 'clientCardinality' '
modelElement = ' 'G.14' '>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.52' '
name = ' 'supplierCardinality' '
modelElement = ' 'G.17' '>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.53' '
name = ' 'clientCardinality' '
modelElement = ' 'G.17' '>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
</XMI.content>
</XMI>').
xmi ← xmi documentoXMI.
hierarquia ← OORDBMHierarquiaDeClassesComParserXMI new xmi: xmi!

ggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.48' '
name = ' 'clientCardinality' '
modelElement = ' 'G.14' '>
<UML:TaggedValue.dataValue>
1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.52' '
name = ' 'supplierCardinality' '
modelElement = ' 'G.17' '>
<UML:TaggedValue.dataValue>
1
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
<UML:TaggedValue xmi.id = ' 'XX.53' '
name = ' 'clientCardinality' '
modelElement = ' 'G.17' '>
<UML:TaggedValue.dataValue>

1..×
</UML:TaggedValue.dataValue>
</UML:TaggedValue>
</XMI.content>
</XMI>').
xmi ← xmi documentoXMI.
hierarquia ← OORDBMHierarquiaDeClassesComParserXMI new xmi: xmi!

Object subclass: OORDBMSQL
Category: OORDBM-DB

Representa uma interface de traducao entre uma representacao em memoria de uma estrutura relacional e seu respectivo codigo SQL para um dado banco de dados.

Class protocol

Methods for category: as yet unclassified

retorneSQLsDeFormaIncrementalParaTabela: *umaTabela*
"Dada umaTabela(OORDBMTabela) deve retornar os SQLs necessarios para sua criacao. Estes devem criar a tabela campo a campo (incremental)"
↑ self subclassResponsibility!

retorneSQLsRelacionamentos: *umaTabela*
"Dada umaTabela(OORDBMTabela) deve retornar os SQLs para criacao das foreign−keys da tabela"
↑ self subclassResponsibility!

retorneSQLsTabela: *umaTabela*
"Deve retornar os SQLs necessarios para criar a tabela informada"
↑ self subclassResponsibility!

retorneSQLsVisao: *umaVisao*
"Deve retornar os SQLs necessarios para criar a visao informada"
↑ self subclassResponsibility!

OORDBMSQL subclass: OORDBMSQLPostgres
Category: OORDBM-DB

Esta classe gera o codigo DDL que representa as tabelas, colunas e relacionamentos do sistema de persistencia(Utiliza as classes OORDBMTabela e OORDBMColuna)

Class protocol

Methods for category: class initialization

initialize

```
dados ← Dictionary new.
dados at: 'Inteiro' put: 'INTEGER'.
dados at: 'Texto' put: 'TEXT'.
dados at: 'Texto\[[0−9]+\]' put: 'TEXT'.
dados at: 'Racional' put: 'DOUBLE PRECISION'.
dados at: 'Real' put: 'DOUBLE PRECISION'.
dados at: 'Decimal' put: 'NUMERIC(30,10)'.
dados at: 'Decimal\[[0−9]+\]' put: 'NUMERIC(30,10)'.
dados at: 'Data' put: 'DATE'.
dados at: 'Hora' put: 'TIME'.
dados at: 'ValorMonetario' put: 'NUMERIC(15,2)'.
dados at: 'Imagem' put: 'BYTEA'.
dados at: 'Qualquer' put: 'BYTEA'.
dados at: 'SimNao' put: 'BOOLEAN'.
dados at: 'Chave' put: 'SERIAL'!
```

new

```
↑ super new initialize.
```

Methods for category: sql

retorneSQLsDeFormaIncrementalParaTabela: *umaTabela*

```
"Retorna os SQLs necessarios para criar a tabela informada, fazendo com
que cada coluna seja criada por um SQL diferente. Dessa forma, se a
estrutura de uma tabela for alterada, e capaz de criar os campos novos
necessarios sem precisar derrubar e criar a tabela novamente.
(ToDo: Nao faz alteracao de tipos de dados nas colunas existentes por
enquanto) "
| retorno linha |
retorno ← OrderedCollection new.
linha ← 'CREATE TABLE ' , umaTabela nome , '('.
umaTabela chaves
   do: [:chave | linha ← linha , chave nome , ' '
            , (OORDBMSQLPostgres tipoDadosPara: chave tipo) , ','].
linha ← linha , ' PRIMARY KEY('.
umaTabela chaves
   do: [:chave | linha ← linha , chave nome , ','].
linha ← (linha first: linha size − 1)
        , '))'.
retorno add: linha.
(umaTabela colunas
   reject: [:coluna | umaTabela chaves includes: coluna])
   do: [:coluna | retorno add: 'ALTER TABLE ' , umaTabela nome , ' ADD ' , coluna nome , ' '
            , (OORDBMSQLPostgres tipoDadosPara: coluna tipo)].
↑ retorno!
```

retorneSQLsRelacionamentos: *umaTabela*

```
"Desativado"
↑ OrderedCollection new"
| retorno |
retorno ← OrderedCollection new.
umaTabela colunas values
do: [:coluna | coluna chaveEstrangeira isNil
ifFalse: [retorno add: 'ALTER TABLE ' , umaTabela nome , ' ADD
CONSTRAINT CE←' , coluna nome , '←' , coluna chaveEstrangeira tabela
nome , ' FOREIGN KEY(' , coluna nome , ') REFERENCES ' , coluna
chaveEstrangeira tabela nome] ].
↑ retorno"!
```

retorneSQLsTabela: *umaTabela*

```
  “Retorna os SQLs necessarios para criar a tabela informada"
  | retorno linha |
  retorno ← OrderedCollection new.
  linha ← ’CREATE TABLE ’ , umaTabela nome , ’(’.
  umaTabela colunas values do: [ :coluna |
    linha ← linha , coluna nome , ’ ’ , (OORDBMSQLPostgres tipoDadosPara: (coluna tipo)) , ’,’.
  ].
  linha ← linha , ’ PRIMARY KEY(’.
  umaTabela chaves do: [ :chave |
    linha ← linha , chave nome , ’,’.
  ].
  linha ← (linha first: (linha size − 1)) , ’))’.
  retorno add: linha.
  ↑ retorno.
```

retorneSQLsVisao: *umaVisao*

```
  “Retorna os SQLs necessarios para criar a tabela informada"
  | retorno linha |
  retorno ← OrderedCollection new.
  retorno add: ’DROP VIEW ’ , umaVisao nome.
  linha ← ’CREATE VIEW ’ , umaVisao nome , ’ AS (SELECT ’.
  umaVisao colunas keys
    do: [:alias |
      | coluna |
      coluna ← umaVisao colunas at: alias.
      linha ← linha , coluna tabela nome , ’.’ , coluna nome , ’ AS ’ , alias , ’, ’].
  linha ← linha , ’ tpCtrlInt as TIPOOBJETO FROM ’ , umaVisao colunas values first tabela nome , ’ WHERE ’ , umaVisao clausulaSQLParaConsulta , ’)’.
  retorno add: linha.
  ↑ retorno!
```

tipoDadosPara: *umTipo*

```
  | tipo |
  dados isNil
    ifTrue: [self initialize].
  tipo ← dados at: umTipo.
  ↑ tipo.!

    il
     ifTrue: [self initialize].
  tipo ← dados at: umTipo.
  ↑ tipo.!
```

Object subclass: OORDBMColuna

Category: OORDBM-RDBM

Instance variable names: nome chaveEstrangeira tipo tabela

Esta classe representa uma coluna de uma tabela (OORDBMTabela)

Instance protocol

Methods for category: accessing

chaveEstrangeira
↑ chaveEstrangeira.
chaveEstrangeira: *umOORDBMCampo*
"Recebe o campo que eh chave primaria de outra tabela"
chaveEstrangeira ← umOORDBMCampo!

nome
↑ nome.
nome: *umNome*
nome ← umNome.
tabela
↑ tabela.
tabela: *umaTabela*
tabela ← umaTabela.
tipo
↑ tipo.
tipo: *umNomeDeTipo*
"Um dos tipos especificados em OORDBMTiposBasicos"
tipo ← umNomeDeTipo!

Object subclass: OORDBMTabela
Category: OORDBM-RDBM
Instance variable names: nome colunas chave chaves

Esta classe representa uma tabela do sistema de persistncia − inclui um nome, um conjunto de colunas e as chaves primarias

Class protocol

Methods for category: as yet unclassified

new
↑ super new initialize.

Instance protocol

Methods for category: accessing

adicionaColuna: *umaColuna*
"Adiciona uma coluna a lista de colunas da tabela"
colunas at: umaColuna nome put: umaColuna.
umaColuna tabela: self.
chave: *umaChave*
chaves add: umaChave.
chaves
↑ chaves.
colunas
↑ colunas
nome
↑ nome.
nome: *umNome*
nome ← umNome.

Methods for category: initialization

initialize
colunas ← Dictionary new.
chaves ← OrderedCollection new.

Object subclass: OORDBMVisao
Category: OORDBM-RDBM
Instance variable names: colunas clausulaConsulta nome

Representa uma visao em um banco de dados relacional. E composta por um conjunto de colunas de outras OORDBMTabela

Class protocol

Methods for category: as yet unclassified

new
↑ super new initialize.

Instance protocol

Methods for category: accessing

nome
↑ nome!

nome: *umNome*
nome ← umNome!

Methods for category: as yet unclassified

adicionarColuna: *umaColuna* **sobOAlias:** *umAlias*
colunas at: umAlias put: umaColuna.

clausulaSQLParaConsulta
↑ clausulaConsulta
clausulaSQLParaConsulta: *umaClausulaSQL*
"O correto seria implementar uma maneira via objetos de definir a condicao, mas para o escopo necessario no momento esta alternativa eh viavel"
clausulaConsulta ← umaClausulaSQL
colunas
↑ colunas
initialize
colunas ← Dictionary new.
esta alternativa eh viavel"
clausulaConsulta ← umaClausulaSQL

colunas
↑ colunas

initialize
colunas ← Dictionary new.

Object subclass: OORDBMGeradorClasses
Category: OORDBM-Geradores
Instance variable names: grafo
Class variable names: Debug

Esta classe abstrata define os aspectos basicos de um gerador de classes a partir de uma hieraquia de classes

Class protocol

Methods for category: as yet unclassified

debug

```
Debug isNil ifTrue: [ ↑ false ] ifFalse: [ ↑ Debug ].
```

debug: *umBoolean*

```
Debug ← umBoolean.
```

gerarMetodo: *umaAssinatura* para: *umaClasse*

```
| assTokens assinatura |
assTokens ← umaAssinatura assinatura findTokens: $ .
assinatura ← ' '.
1
   to: assTokens size
   do: [:pos | pos odd
         ifTrue: [assinatura ← assinatura
                  , (assTokens at: pos)] ].
(umaClasse includesSelector: assinatura asSymbol)
   ifFalse: [Transcript show: umaAssinatura assinatura;
         cr.
      umaClasse compile: umaAssinatura assinatura classified: 'metodos']!
```

Methods for category: geracao

adicionarAtributo: *umAtributo* a: *umaClasse*

```
   [umaClasse addInstVarName: umAtributo.
]
      ifError: [↑ false].
   ↑ true!
```

adicionarAtributos: *listaAtributos* a: *umaClasse*

```
   | retorno |
   retorno ← true.
   listaAtributos
      do: [:atributo | retorno ← retorno
               and: [self adicionarAtributo: atributo a: umaClasse] ].
   ↑ retorno!
```

gerarClasse: *nomeClasse*

```
   ↑ self gerarClasse: nomeClasse categoria: ' '.
```

gerarClasse: *nomeClasse* categoria: *categoriaDaClasse*

```
   ↑ self gerarClasse: nomeClasse superClasse: 'Object' categoria: categoriaDaClasse.
```

gerarClasse: *nomeClasse* categoria: *categoriaDaClasse* variaveis: *variaveis.*

```
   ↑ self gerarClasse: nomeClasse superClasse: 'Object' categoria: categoriaDaClasse variaveis: variaveis.
```

gerarClasse: *nomeClasse* superClasse: *nomeSuperclasse*

```
   ↑ self gerarClasse: nomeClasse superClasse: nomeSuperclasse categoria: ' '
```

gerarClasse: *nomeClasse* superClasse: *nomeSuperclasse* categoria: *categoriaDaClasse*

```
   ↑ self gerarClasse: nomeClasse superClasse: nomeSuperclasse categoria: categoriaDaClasse variaveis: ' '.
```

gerarClasse: *nomeClasse* superClasse: *nomeSuperclasse* categoria: *categoriaDaClasse* variaveis: *variaveis*

```
   "Gera a classe conforme as informacoes passadas − se ja existir, apenas
   recompila a mesma, incluindo variaveis novas que nao existiam antes e
   corrigindo a categoria da mesma, caso tenha sido trocada"
   | codigoClasse referencias variaveisNovas variaveisAntigas variaveisAtualizadas |
   referencias ← OrderedCollection new.
   (Smalltalk classNamed: nomeClasse) isNil
      ifFalse: [referencias
            addAll: (self
                  referenciasA: (Smalltalk classNamed: nomeClasse))].
   self debug
      ifTrue: [Transcript show: 'Referencias a ' , nomeClasse , ':' , referencias size asString;
            cr].
   (Smalltalk classNamed: nomeClasse) isNil
```

```
    ifTrue: [variaveisAtualizadas ← variaveis]
    ifFalse: [variaveisAtualizadas ← ' '.
      variaveisAntigas ← (Smalltalk classNamed: nomeClasse) instVarNames.
      variaveisNovas ← (variaveis findTokens: $ )
            difference: variaveisAntigas.
      ((OrderedCollection withAll: variaveisAntigas)
        addAll: variaveisNovas; yourself)
        do: [:variavel | variaveisAtualizadas ← variaveisAtualizadas , ' ' , variavel].
      ].
  codigoClasse ← nomeSuperclasse , ' subclass: #' , nomeClasse , ' instanceVariableNames: ' '' , variaveisAtualizadas , ' '' classVa-
riableNames: ' '' ' poolDictionaries: ' '' ' category: ' '' , categoriaDaClasse , ' '' '.
  self debug
    ifTrue: [Transcript show: codigoClasse].
  Cursor wait
    showWhile: [Compiler evaluate: codigoClasse logged: true].
  referencias
    do: [:referencia | referencia classIsMeta
        ifTrue: [[(Smalltalk classNamed: referencia actualClass asString) theMetaClass recompile: referencia methodSymbol]
            ifError: [:err | Transcript cr; show: '[META]Erro na compilacao!'; show: err] ]
        ifFalse: [[(Smalltalk classNamed: referencia actualClass asString)
            recompile: referencia methodSymbol]
            ifError: [:err | Transcript cr; show: 'Erro na compilacao!'; show: err] ]]!
```

gerarGetterColecaoDe: *umAtributo* **em:** *umaClasse*

```
  | codigoMetodo |
  codigoMetodo ← TextStream with: ' '.
  codigoMetodo nextPutAll: umAtributo nome;
    cr;
    crtab.
  codigoMetodo nextPutAll: umAtributo nome , ' isNil ifTrue: [ ' , umAtributo nome , ' ← '.
  umAtributo maximo ≠ −1
    ifTrue: [codigoMetodo nextPutAll: '(TypedLimitedOrderedCollection new: ' , umAtributo maximo asString , ') tipo: ' , umA-
tributo tipo nome , '; yourself.';
        cr;
        crtab]
    ifFalse: [codigoMetodo nextPutAll: 'TypedOrderedCollection new tipo: ' , umAtributo tipo nome;
        crtab].
  codigoMetodo nextPutAll: '].';
    cr;
    crtab;
    nextPutAll: '↑ ' , umAtributo nome , '.'.
  umAtributo nome , '.'.
  self debug
    ifTrue: [Transcript show: codigoMetodo contents;
        cr].
  umaClasse compile: codigoMetodo contents classified: 'atributos'!
```

gerarGetterDe: *umAtributo* **em:** *umaClasse*

```
  | codigoMetodo |
  codigoMetodo ← TextStream with: ' '.
  codigoMetodo ← umAtributo nome , '
    ↑ '.
  (OORDBMTiposBasicos blocoGetterPara: umAtributo tipo nome)
      = ' '
    ifTrue: [codigoMetodo ← codigoMetodo , umAtributo nome , '.']
    ifFalse: [codigoMetodo ← codigoMetodo
          , (OORDBMTiposBasicos blocoGetterPara: umAtributo tipo nome) , ' value: ' , umAtributo nome].
  Transcript show: codigoMetodo contents;
    cr.
  umaClasse compile: codigoMetodo contents classified: 'atributos'!
```

gerarGetterSetterColecaoDe: *umAtributo* **em:** *umaClasse*

```
  self
    gerarGetterColecaoDe: umAtributo
    em: umaClasse. self
    gerarSetterColecaoDe: umAtributo
    em: umaClasse!
```

gerarGetterSetterDe: *umAtributo* **em:** *umaClasse*

```
  self gerarGetterDe: umAtributo em: umaClasse.
  self gerarSetterDe: umAtributo em: umaClasse.
!
```

gerarSetterColecaoDe: *umAtributo* **em:** *umaClasse*

```
  | codigoMetodo nomeParametro |
  codigoMetodo ← TextStream with: ' '.
  nomeParametro ← 'os' , umAtributo nome capitalized.
  codigoMetodo nextPutAll: (umAtributo nome , ': ' , nomeParametro); cr; crtab.
  codigoMetodo nextPutAll: umAtributo nome , ' ← '.
  umAtributo maximo ≠ −1 ifTrue: [
    codigoMetodo nextPutAll: '(TypedLimitedOrderedCollection new: ' , (umAtributo maximo asString) , ') tipo: ' , umAtributo
tipo nome , '; yourself.'; cr; crtab.
  ] ifFalse: [
    codigoMetodo nextPutAll: 'TypedOrderedCollection new tipo: ' , umAtributo tipo nome , '.'; crtab.
  ].
  codigoMetodo nextPutAll: umAtributo nome , ' addAll: ' , nomeParametro , '.'.
Transcript show: codigoMetodo contents;
        cr.
      umaClasse compile: codigoMetodo contents classified: 'atributos'
!
```

gerarSetterDe: *umAtributo* **em:** *umaClasse*

```
  | codigoMetodo nomeParametro |
  codigoMetodo ← TextStream with: ' '.
  nomeParametro ← 'um' , umAtributo nome capitalized.
  codigoMetodo nextPutAll: umAtributo nome , ': ' , nomeParametro;
    cr;
    crtab.
  (OORDBMTiposBasicos ehTipoBasico: umAtributo tipo nome)
    ifTrue: [codigoMetodo nextPutAll: '[ '
          , (OORDBMTiposBasicos blocoValidadorPara: umAtributo tipo nome) , ' value: ' , nomeParametro , '] ifError: [self
error: ' 'Valor passado incompatível!' '].';
          crtab;
          tab.
      codigoMetodo nextPutAll: umAtributo nome , ' ← '
          , (OORDBMTiposBasicos blocoSetterPara: umAtributo tipo nome) , ' value: ' , nomeParametro]
    ifFalse: [codigoMetodo nextPutAll: '(' , nomeParametro , ' isKindOf: ' , umAtributo tipo nome , ') ifFalse: [ ↑ self error: ' 'Valor
nao e do tipo esperado' ' ] ifTrue: [';
          crtab;
          tab.
      codigoMetodo nextPutAll: umAtributo nome , ' ← ' , nomeParametro].
  (OORDBMTiposBasicos ehTipoBasico: umAtributo tipo nome)
    ifFalse: [codigoMetodo crtab; nextPutAll: ']'].
  Transcript show: codigoMetodo contents;
    cr.
  umaClasse compile: codigoMetodo contents classified: 'atributos'.
!
```

referenciasA: *umaClasse*

```
  ↑ (SystemNavigation default allCallsOn: (Smalltalk associationAt: umaClasse theNonMetaClass name)).
```

Instance protocol

Methods for category: accessing

hierarquia

```
  ↑ grafo
```

hierarquia: *anObject*

```
  "Recebe um OORDBMHierarquiaDeClasses do qual ira gerar as classes do sistema"
  grafo ← anObject!
```

Methods for category: converting

gerar

```
  self gerarClasses.
  self adicionarAtributosClasses.
```

```
self adicionarRelacionamentosClasses.
self gerarMetodos.!
```

Methods for category: private

adicionarAtributosClasse: *umaClasse*

```
"Adiciona os atributos (e seus correspondentes getters/setters) das classes"
| classe atributosSimples |
classe ← Smalltalk at: (umaClasse nome) asSymbol.
atributosSimples ← OrderedCollection new.
umaClasse atributos
   do: [:atributo |
      self species adicionarAtributo: atributo nome a: classe.
      atributosSimples add: atributo].
self species adicionarAtributo: 'intid' a: classe.
(grafo classesSuperiores includes: umaClasse)
   ifTrue: [atributosSimples add: (OORDBMAtributo new nome: 'intid';
            tipo: (OORDBMClasse new nome: 'Inteiro');
            yourself)].
atributosSimples
   do: [:nAtributo | self species gerarGetterSetterDe: nAtributo em: classe]!
```

adicionarAtributosClasses

```
"Adiciona os atributos as classes presentes na hierarquia. Gera tambem getters e setters para os mesmos"
| entidades entidade |
entidades ← OrderedCollection
         withAll: (grafo classesSuperiores
               select: [:ent | (OORDBMTiposBasicos ehTipoBasico: ent nome) not]).
[entidades isEmpty]
   whileFalse: [entidade ← entidades removeFirst.
      self adicionarAtributosClasse: entidade.
      entidades
         addAll: (grafo classesFilhas: entidade)]!
```

adicionarMetodosClasse: *umaClasse*

```
"Para a classe informada, investiga as assinaturas de metodos associadas
a mesma e gera os metodos nas classes correspondentes no sistema"
| classe |
classe ← Smalltalk at: (umaClasse nome) asSymbol.
umaClasse metodos
   do: [:metodo | self species gerarMetodo: metodo para: classe]!
```

adicionarMetodosClasses

```
"Adiciona os metodos das classes constantes na hierarquia"
| entidades entidade |
entidades ← OrderedCollection
         withAll: (grafo classesSuperiores
               select: [:ent | (OORDBMTiposBasicos ehTipoBasico: ent nome) not]).
[entidades isEmpty]
   whileFalse: [entidade ← entidades removeFirst.
      self adicionarMetodosClasse: entidade.
      entidades
         addAll: (grafo classesFilhas: entidade)]!
```

adicionarRelacionamentosClasse: *umaClasse*

```
"Percorre os relacionamentos da classe informada e gera os atributos (tanto simples quanto colecoes)"
| classe atributosSimples atributosColecao |
classe ← Smalltalk at: umaClasse nome asSymbol.
atributosSimples ← OrderedCollection new.
atributosColecao ← OrderedCollection new.
(grafo relacionamentosDe: umaClasse)
   do: [:relacionamento | self
         gerarAtributoCorrespondenteA: relacionamento
         considerando: umaClasse
         em: atributosSimples
         ou: atributosColecao].
self species
```

```
    adicionarAtributos: (atributosSimples
        collect: [:atributo | atributo nome])
    a: classe.
  self species
    adicionarAtributos: (atributosColecao
        collect: [:atributo | atributo nome])
    a: classe.
  atributosSimples
    do: [:nAtributo | self species gerarGetterSetterDe: nAtributo em: classe].
  atributosColecao
    do: [:nAtributo | self species gerarGetterSetterColecaoDe: nAtributo em: classe]!
```

adicionarRelacionamentosClasses

```
  "Percorre o grafo de classes, gerando o suporte no sistema aos relacionamentos entre classes (a excecao da heranca) contidos no
mesmo"
  (grafo classes
    select: [:entidade | (OORDBMTiposBasicos ehTipoBasico: entidade nome) not])
    do: [:entidade | self adicionarRelacionamentosClasse: entidade]!
```

gerarAtributoCorrespondenteA: *umRelacionamento* considerando: *umaClasse* em: *colecaoAtributosSimples* ou: *colecaoAtributosDeColecao*

```
  "Gera duas colecoes contendo OORDBMAtributo (e derivados) que serao utilizados posteriormente para gerar os getters/setters/variaveis
de instancia dos atributos nas classes"
  | classeOrigem classeDestino grauOrigem grauDestino nomeAtributoOrigem nomeAtributoDestino |
  classeOrigem ← umRelacionamento classeOrigem.
  classeDestino ← umRelacionamento classeDestino.
  grauOrigem ← umRelacionamento grauOrigem.
  grauDestino ← umRelacionamento grauDestino.
  nomeAtributoOrigem ← umRelacionamento nomeAtributoOrigem.
  nomeAtributoDestino ← umRelacionamento nomeAtributoDestino.
  classeOrigem = umaClasse
    ifTrue: [(grauOrigem = 1
          and: [nomeAtributoOrigem isEmpty not])
        ifTrue: [colecaoAtributosSimples add: (OORDBMAtributo new nome: nomeAtributoOrigem;
              tipo: (OORDBMClasse new nome: classeDestino nome);
              yourself)]
        ifFalse: [(grauOrigem ≠ 1
              and: [nomeAtributoOrigem isEmpty not])
            ifTrue: [colecaoAtributosDeColecao add: (OORDBMAtributoDeColecao new nome: nomeAtributoOrigem;
                  tipo: (OORDBMClasse new nome: classeDestino nome);
                  maximo: grauOrigem;
                  yourself)] ]].
  classeDestino = umaClasse
    ifTrue: [(grauDestino = 1
          and: [nomeAtributoDestino isEmpty not])
        ifTrue: [colecaoAtributosSimples add: (OORDBMAtributo new nome: nomeAtributoDestino;
              tipo: (OORDBMClasse new nome: classeOrigem nome);
              yourself)]
        ifFalse: [(grauDestino ≠ 1
              and: [nomeAtributoDestino isEmpty not])
            ifTrue: [colecaoAtributosDeColecao add: (OORDBMAtributoDeColecao new nome: nomeAtributoDestino;
                  tipo: (OORDBMClasse new nome: classeOrigem nome);
                  maximo: grauDestino;
                  yourself)] ]]!
```

gerarClasses

```
  | nomeClasse entidades categoria pai entidade |
  "Utilizando um algoritimo similiar a uma busca em largura, percorre a hierarquia de classes gerando cada classe presente na mesma"
  categoria ← grafo categoria.
  entidades ← OrderedCollection
        withAll: (grafo classesSuperiores
            select: [:ent | (OORDBMTiposBasicos ehTipoBasico: ent nome) not]).
  [entidades isEmpty]
    whileFalse: [entidade ← entidades removeFirst.
      pai ← grafo classePai: entidade.
      nomeClasse ← entidade nome.
      pai isNil
```

```
            ifTrue: [self species gerarClasse: nomeClasse categoria: categoria]
            ifFalse: [self species
                  gerarClasse: nomeClasse
                  superClasse: pai nome
                  categoria: categoria].
        entidades
            addAll: (grafo classesFilhas: entidade)]!
```

gerarMetodos

```
    ↑ self adicionarMetodosClasses.!
```

Object subclass: OORDBMGeradorDeInterfaceDeEdicao

Category: OORDBM-Geradores

Instance variable names: classe nomeGerenciadorDePersistencia nomeClasseGerar nomeCategoria

Esta classe deve gerar uma interface (visual) para manipulacao de objetos da classe passada, incluindo aspectos de persistencia de dados.

Instance protocol

Methods for category: accessing

classe
```
↑ classe!
```

classe: *anObject*
```
classe ← anObject!
```

codigoParaAbrirNavegador
```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'abrir';
    cr;
    crtab.
codigo nextPutAll: 'self openInSystemWindow.';
    tab.
↑ codigo contents!
```

codigoParaIniciarSessao
```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'iniciarSessao';
    cr;
    crtab.
codigo nextPutAll: 'sessao ← ' , nomeGerenciadorDePersistencia , ' new. sessao iniciarSessao. elementos ← OrderedCollecti-
onWithCursor withAll: (sessao recuperarEntidadesDoTipo: ' , classe nome , ' ). self atualizarDadosNaTela.';
    crtab;
    tab.
↑ codigo contents!
```

codigoParaInitializeDaClasse
```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'initialize';
    cr;
    crtab.
codigo nextPutAll: 'super initialize. self buildOn: self. corpo ← self. inserindo ← false.';
    crtab.
↑ codigo contents!
```

codigoParaInserirRegistro
```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'inserirRegistro';
    cr;
    crtab.
codigo nextPutAll: 'elementos addLast: (' , classe nome , ' new). elementos cursorToLast. self atualizarDadosNaTela. inserindo
← true.';
    crtab.
↑ codigo contents!
```

codigoParaMetodoIrParaPrimeiroRegistro
```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'irParaPrimeiroRegistro';
    cr;
    crtab.
codigo nextPutAll: 'elementos cursorToFirst.';
    crtab.
codigo nextPutAll: 'self atualizarDadosNaTela.'.
↑ codigo contents!
```

codigoParaMetodoIrParaRegistroAnterior
```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'irParaRegistroAnterior';
   cr;
   crtab.
codigo nextPutAll: 'elementos retrocedeCursor.';
   crtab.
codigo nextPutAll: 'self atualizarDadosNaTela.'.
↑ codigo contents!
```

codigoParaTerminarSessao
```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'terminarSessao';
   cr;
   crtab.
codigo nextPutAll: 'sessao terminarSessao. elementos ← OrderedCollectionWithCursor new.';
   tab.
↑ codigo contents!
```

gerar
```
| aClasse |
self gerarClasse.
aClasse ← Smalltalk at: nomeClasseGerar asSymbol.
aClasse compile: self codigoParaAtualizarDadosNaTela classified: 'private'.
aClasse compile: self codigoParaCancelarAlteracoes classified: 'private'.
aClasse compile: self codigoParaExcluirRegistro classified: 'private'.
aClasse compile: self codigoParaGravarRegistro classified: 'private'.
aClasse compile: self codigoParaIniciarSessao classified: 'private'.
aClasse compile: self codigoParaInserirRegistro classified: 'private'.
aClasse compile: self codigoParaTerminarSessao classified: 'private'.
aClasse compile: self codigoParaMetodoIrParaPrimeiroRegistro classified: 'private'.
aClasse compile: self codigoParaMetodoIrParaUltimoRegistro classified: 'private'.
aClasse compile: self codigoParaMetodoIrParaProximoRegistro classified: 'private'.
aClasse compile: self codigoParaMetodoIrParaRegistroAnterior classified: 'private'.
aClasse compile: self codigoParaMetodoIrParaUltimoRegistro classified: 'private'.
aClasse compile: self codigoParaConstrucaoInterface classified: 'private'.
aClasse compile: self codigoParaInitializeDaClasse classified: 'private'.
aClasse compile: self codigoParaAtualizarDadosDoRegistro classified: 'private'.
aClasse compile: self codigoParaAbrirNavegador classified: 'system'.!
```

gerarClasse
```
↑ OORDBMGeradorClasses
   gerarClasse: nomeClasseGerar
   superClasse: (self classeSuperior)
   categoria: nomeCategoria
   variaveis: 'sessao elementos inserindo corpo'!
```

larguraDaTela
```
↑ 579!
```

nomeCategoria
```
↑ nomeCategoria!
```

nomeCategoria: *anObject*
```
"Set the value of nomeCategoria"
nomeCategoria ← anObject
```

nomeClasseGerar
```
"Answer the value of nomeClasseGerar"
↑ nomeClasseGerar
```

nomeClasseGerar: *anObject*
```
"Set the value of nomeClasseGerar"
nomeClasseGerar ← anObject
```

nomeGerenciadorDePersistencia
```
"Answer the value of nomeGerenciadorDePersistencia"
↑ nomeGerenciadorDePersistencia
```

nomeGerenciadorDePersistencia: *anObject*
"Set the value of nomeGerenciadorDePersistencia"
nomeGerenciadorDePersistencia ← anObject

Methods for category: as yet unclassified

alturaDoNavegador
self subClassResponsibility

alturaMaximaDaTela
↑ 454!

alturaParaUmTipoDeAtributo: *umTipoDeAtributo*
self subClassResponsibility!

classeSuperior
self subClassResponsibility

codigoBasicoNavegador
↑ self subClassResponsibility

codigoBasicoTela
self subClassResponsibility

codigoParaAtributoImagemChamado: *umNome* **aPartirDaCoordenada:** *umaCoordenada*
self subClassResponsibility

codigoParaAtributoTextoChamado: *umNome* **aPartirDaCoordenada:** *umaCoordenada*
self subClassResponsibility!

codigoParaAtributos

```
| posicao codigo |
posicao ← Point x: 10 y: self alturaDoNavegador + 10.
codigo ← TextStream with: ' '.
classe atributos
   do: [:atributo |
      codigo
         nextPutAll: (self codigoParaAtributoTextoChamado: atributo nome aPartirDaCoordenada: posicao).
      posicao ← Point x: 10 y: posicao y
                  + (self alturaParaUmTipoDeAtributo: atributo tipo nome)].
↑ codigo contents!
```

codigoParaAtualizarDadosDoRegistro

```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'atualizarDadosDoRegistro';
   cr;
   crtab.
codigo nextPutAll: 'elementos elementAtCursor isNil ifFalse: [';
   crtab.
classe atributos
   do: [:atributo | codigo nextPutAll: 'elementos elementAtCursor ' , atributo nome , ': '
         , (self codigoParaRecuperarValorNaTelaDoAtributo: atributo nome);
      crtab].
codigo nextPutAll: ']';
   crtab.
↑ codigo contents!
```

codigoParaAtualizarDadosNaTela

```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'atualizarDadosNaTela';
   cr;
   crtab.
codigo nextPutAll: 'elementos elementAtCursor isNil ifFalse: [';
   crtab.
classe atributos
   do: [:atributo | codigo nextPutAll: '(elementos elementAtCursor ' , atributo nome , ') isNil ifTrue: [ '
         , (self codigoParaDefinirValorNaTelaDoAtributo: atributo nome) , ' '" ' ] ifFalse: [ (corpo clientWithTag: ' '" , atributo
nome , ' '") text: (elementos elementAtCursor ' , atributo nome , ' asString).].';
      crtab].
codigo nextPutAll: '] ifTrue: [';
   crtab.
```

```
classe atributos
    do: [:atributo | codigo nextPutAll: (self codigoParaDefinirValorNaTelaDoAtributo: atributo nome)
                ,' ' ,'' '.';
            crtab].
codigo nextPutAll: ']';
    crtab.
↑ codigo contents!
```

codigoParaCancelarAlteracoes

```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'cancelarAlteracoes';
    cr;
    crtab.
codigo nextPutAll: 'inserindo ifTrue: [elementos removeLast. elementos adjustCursor. self atualizarDadosNaTela] ifFalse: [';
    crtab;
    tab.
codigo nextPutAll: 'self atualizarDadosNaTela.]';
    crtab.
↑ codigo contents!
```

codigoParaConstrucaoInterface

```
↑ self codigoBasicoTela , self codigoBasicoNavegador , self codigoParaAtributos!
```

codigoParaExcluirRegistro

```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'excluirRegistro';
    cr;
    crtab.
codigo nextPutAll: 'inserindo ifTrue: [elementos removeLast. elementos cursorToLast. self atualizarDadosNaTela. inserindo ← false.] ifFalse: [';
    crtab;
    tab.
codigo nextPutAll: 'elementos elementAtCursor isNil ifFalse: [sessao excluir: (elementos elementAtCursor). sessao gravarEContinuarSessao. elementos remove: (elementos elementAtCursor). elementos adjustCursor. self atualizarDadosNaTela.] ]';
    crtab.
↑ codigo contents!
```

codigoParaGravarRegistro

```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'gravarRegistro';
    cr;
    crtab.
codigo nextPutAll: 'inserindo ifTrue: [sessao registrarNovo: (elementos elementAtCursor). self atualizarDadosDoRegistro. sessao gravarEContinuarSessao. inserindo ← false.] ifFalse: [';
    crtab;
    tab.
codigo nextPutAll: 'self atualizarDadosDoRegistro. sessao gravarEContinuarSessao.]';
    crtab.
↑ codigo contents!
```

codigoParaMetodoIrParaProximoRegistro

```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: 'irParaProximoRegistro';
    cr;
    crtab.
codigo nextPutAll: 'elementos advanceCursor.';
    crtab.
codigo nextPutAll: 'self atualizarDadosNaTela.'.
↑ codigo contents!
```

codigoParaMetodoIrParaUltimoRegistro

```
| codigo |
codigo ← TextStream with: ' '.
```

```
codigo nextPutAll: 'irParaUltimoRegistro';
   cr;
   crtab.
codigo nextPutAll: 'elementos cursorToLast.';
   crtab.
codigo nextPutAll: 'self atualizarDadosNaTela.'.
↑ codigo contents!
```

OORDBMGeradorDeInterfaceDeEdicao subclass: OORDBMGeradorDeInterfaceDeEdicaoPrefab
Category: OORDBM-Geradores

Gera interfaces de edicao usando o pacote Prefab do Squeak

Instance protocol

Methods for category: as yet unclassified

alturaDoNavegador

```
↑ 50
```

alturaParaUmTipoDeAtributo: *umTipoDeAtributo*

```
umTipoDeAtributo = 'Imagem'
    ifTrue: [↑ 155]
    ifFalse: [↑ 30]!
```

classeSuperior

```
↑ 'PrefabManager'!
```

codigoBasicoNavegador

```
| codigo |
codigo ← TextStream with: ' '.
codigo nextPutAll: '    aManager addClient: (PrefabFrame new';
    crtab.
codigo nextPutAll: '        extent: 536@35;';
    crtab.
codigo nextPutAll: '        color: (Color r: 0.851 g: 0.851 b: 0.851);';
    crtab.
codigo nextPutAll: '        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
    crtab.
codigo nextPutAll: '        style: #inset;';
    crtab.
codigo nextPutAll: '        yourself)';
    crtab.
codigo nextPutAll: '        atRelPosition: 13@10.';
    crtab.
codigo nextPutAll: 'aManager addClient: (PrefabButton new';
    crtab.
codigo nextPutAll: '        extent: 64@28;';
    crtab.
codigo nextPutAll: '        color: (Color r: 0.851 g: 0.851 b: 0.851);';
    crtab.
codigo nextPutAll: '        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
    crtab.
codigo nextPutAll: '        action: [self irParaPrimeiroRegistro];';
    crtab.
codigo nextPutAll: '        tag: #botaoPrimeiro;';
    crtab.
codigo nextPutAll: '        text: ' ' Primeiro ' ';';
    crtab.
codigo nextPutAll: '        contentOrientation: #centered;';
    crtab.
codigo nextPutAll: '        yourself)';
    crtab.
codigo nextPutAll: '        atRelPosition: 19@13.';
    crtab.
codigo nextPutAll: '    aManager addClient: (PrefabButton new';
    crtab.
codigo nextPutAll: '        extent: 79@27;';
    crtab.
codigo nextPutAll: '        color: (Color r: 0.851 g: 0.851 b: 0.851);';
    crtab.
codigo nextPutAll: '        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
    crtab.
codigo nextPutAll: '        action: [self cancelarAlteracoes];';
    crtab.
codigo nextPutAll: '        tag: #botaoCancelar;';
    crtab.
codigo nextPutAll: '        text: ' 'Cancelar alt.' ';';
    crtab.
```

```
codigo nextPutAll: '        contentOrientation: #centered;';
    crtab.
codigo nextPutAll: '        yourself)';
    crtab.
codigo nextPutAll: '        atRelPosition: 462@14.';
    crtab.
codigo nextPutAll: '    aManager addClient: (PrefabButton new';
    crtab.
codigo nextPutAll: '        extent: 56@28;';
    crtab.
codigo nextPutAll: '        color: (Color r: 0.851 g: 0.851 b: 0.851);';
    crtab.
codigo nextPutAll: '        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
    crtab.
codigo nextPutAll: '        action: [self irParaProximoRegistro];';
    crtab.
codigo nextPutAll: '        tag: #botaoProximo;';
    crtab.
codigo nextPutAll: '        text: ' 'Proximo' ';';
    crtab.
codigo nextPutAll: '        contentOrientation: #centered;';
    crtab.
codigo nextPutAll: '        yourself)';
    crtab.
codigo nextPutAll: '        atRelPosition: 147@13.';
    crtab.
codigo nextPutAll: '    aManager addClient: (PrefabButton new';
    crtab.
codigo nextPutAll: '        extent: 43@27;';
    crtab.
codigo nextPutAll: '        color: (Color r: 0.851 g: 0.851 b: 0.851);';
    crtab.
codigo nextPutAll: '        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
    crtab.
codigo nextPutAll: '        action: [self irParaUltimoRegistro];';
    crtab.
codigo nextPutAll: '        tag: #botaoUltimo;';
    crtab.
codigo nextPutAll: '        text: ' 'Ultimo' ';';
    crtab.
codigo nextPutAll: '        contentOrientation: #centered;';
    crtab.
codigo nextPutAll: '        yourself)';
    crtab.
codigo nextPutAll: '        atRelPosition: 204@13.';
    crtab.
codigo nextPutAll: '    aManager addClient: (PrefabButton new';
    crtab.
codigo nextPutAll: '        extent: 68@27;';
    crtab.
codigo nextPutAll: '        color: (Color r: 0.851 g: 0.851 b: 0.851);';
    crtab.
codigo nextPutAll: '        foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
    crtab.
codigo nextPutAll: '        action: [self gravarRegistro];';
    crtab.
codigo nextPutAll: '        tag: #botaoSalvar;';
    crtab.
codigo nextPutAll: '        text: ' 'Salvar alt.' ';';
    crtab.
codigo nextPutAll: '        contentOrientation: #centered;';
    crtab.
codigo nextPutAll: '        yourself)';
    crtab.
codigo nextPutAll: '        atRelPosition: 394@14.';
    crtab.
codigo nextPutAll: '    aManager addClient: (PrefabButton new';
```

```
    crtab.
  codigo nextPutAll: '      extent: 62@28;';
    crtab.
  codigo nextPutAll: '      color: (Color r: 0.851 g: 0.851 b: 0.851);';
    crtab.
  codigo nextPutAll: '      foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
    crtab.
  codigo nextPutAll: '      action: [self irParaRegistroAnterior];';
    crtab.
  codigo nextPutAll: '      tag: #botaoAnterior;';
    crtab.
  codigo nextPutAll: '      text: ' 'Anterior' ';';
    crtab.
  codigo nextPutAll: '      contentOrientation: #centered;';
    crtab.
  codigo nextPutAll: '      yourself)';
    crtab.
  codigo nextPutAll: '      atRelPosition: 84@13.';
    crtab.
  codigo nextPutAll: '   aManager addClient: (PrefabButton new';
    crtab.
  codigo nextPutAll: '      extent: 46@27;';
    crtab.
  codigo nextPutAll: '      color: (Color r: 0.851 g: 0.851 b: 0.851);';
    crtab.
  codigo nextPutAll: '      foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
    crtab.
  codigo nextPutAll: '      action: [self excluirRegistro];';
    crtab.
  codigo nextPutAll: '      tag: #botaoExcluir;';
    crtab.
  codigo nextPutAll: '      text: ' 'Excluir' ';';
    crtab.
  codigo nextPutAll: '      contentOrientation: #centered;';
    crtab.
  codigo nextPutAll: '      yourself)';
    crtab.
  codigo nextPutAll: '      atRelPosition: 327@13.';
    crtab.
  codigo nextPutAll: '   aManager addClient: (PrefabButton new';
    crtab.
  codigo nextPutAll: '      extent: 54@26;';
    crtab.
  codigo nextPutAll: '      color: (Color r: 0.851 g: 0.851 b: 0.851);';
    crtab.
  codigo nextPutAll: '      foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
    crtab.
  codigo nextPutAll: '      action: [self inserirRegistro];';
    crtab.
  codigo nextPutAll: '      tag: #botaoInserir;';
    crtab.
  codigo nextPutAll: '      text: ' 'Inserir' ';';
    crtab.
  codigo nextPutAll: '      contentOrientation: #centered;';
    crtab.
  codigo nextPutAll: '      yourself)';
    crtab.
  codigo nextPutAll: '      atRelPosition: 273@13.';
    crtab.
  ↑ codigo contents!
```

codigoBasicoTela

```
  | codigo alturaTotal dimensaoTela |
  alturaTotal ← self alturaDoNavegador + 10.
  dimensaoTela ← Point x: self larguraDaTela y: self alturaMaximaDaTela.
  classe atributos
    do: [:atributo | alturaTotal ← alturaTotal
```

```
            + (self alturaParaUmTipoDeAtributo: atributo tipo nome)].
    alturaTotal < self alturaMaximaDaTela
        ifTrue: [dimensaoTela ← Point x: self larguraDaTela y: alturaTotal].
    codigo ← TextStream with: 'buildOn: aManager'.
    codigo crtab.
    codigo nextPutAll: 'aManager';
        crtab.
    codigo nextPutAll: '      extent: ' , dimensaoTela asString , ';';
        crtab.
    codigo nextPutAll: '      color: (Color r: 0.851 g: 0.851 b: 0.851);';
        crtab.
    codigo nextPutAll: '      foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
        crtab.
    codigo nextPutAll: '      minSize: 10@10;';
        crtab.
    codigo nextPutAll: '      maxSize: 1073741823@1073741823.';
        crtab.
    alturaTotal > self alturaMaximaDaTela
        ifTrue: [codigo nextPutAll: '   aManager addClient: (PrefabScrollBar new';
            crtab.
          codigo nextPutAll: '      extent: 18@435;';
            crtab.
          codigo nextPutAll: '      color: (Color r: 0.6 g: 0.6 b: 0.8);';
            crtab.
          codigo nextPutAll: '      foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
            crtab.
          codigo nextPutAll: '      orientation: #vertical;';
            crtab.
          codigo nextPutAll: '      yourself)';
            crtab.
          codigo nextPutAll: '      atRelPosition: 552@10.';
            crtab].
    ↑ codigo contents!
```

codigoParaAtributoTextoChamado: *umNome* **aPartirDaCoordenada:** *umaCoordenada*

```
    | codigo |
    codigo ← TextStream with: ''.
    codigo nextPutAll: ' aManager addClient: (PrefabLabel new';
        crtab.
    codigo nextPutAll: '      extent: 49@28;';
        crtab.
    codigo nextPutAll: '      color: (Color r: 0.851 g: 0.851 b: 0.851);';
        crtab.
    codigo nextPutAll: '      foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);';
        crtab.
    codigo nextPutAll: '      text: '' , umNome , '';';
        crtab.
    codigo nextPutAll: '      contentOrientation: #centered;';
        crtab.
    codigo nextPutAll: '      yourself)';
        crtab.
    codigo nextPutAll: '      atRelPosition: ' , umaCoordenada asString , '.';
        crtab.
    codigo nextPutAll: '   aManager addClient: (PrefabEdit new';
        crtab.
    codigo nextPutAll: '      extent: 473@20;';
        crtab.
    codigo nextPutAll: '      color: (Color r: 0.0 g: 0.0 b: 0.0);';
        crtab.
    codigo nextPutAll: '      tag: '' , umNome , '';';
        crtab.
    codigo nextPutAll: '      foregroundColor: (Color r: 0.323 g: 1.0 b: 0.161);';
        crtab.
    codigo nextPutAll: '      yourself)';
        crtab.
    codigo nextPutAll: '      atRelPosition: '
        , (Point x: umaCoordenada x + 54 y: umaCoordenada y) asString , '.';
```

```
    crtab.
  ↑ codigo contents!
```

codigoParaDefinirValorNaTelaDoAtributo: *umNomeDeAtributo*

```
  ↑ '(corpo clientWithTag: ' '' , umNomeDeAtributo , ' '') text: '!
```

codigoParaOpenInSystemWindow

```
  | codigo |
  codigo ← TextStream with: ' '.
  codigo nextPutAll: 'openInSystemWindow';
    cr;
    crtab.
  codigo nextPutAll: '    | nm |';
    crtab.
  codigo nextPutAll: '    self world';
    crtab.
  codigo nextPutAll: '        ifNotNil: [self delete].';
    crtab.
  codigo nextPutAll: '    (nm ← Prefab managerFrameClass new) client: self;';
    crtab.
  codigo nextPutAll: '        setLabel: title;';
    crtab.
  codigo nextPutAll: '        openInWorld.';
    crtab.
  codigo nextPutAll: '    nm';
    crtab.
  codigo nextPutAll: '        color: (Color';
    crtab.
  codigo nextPutAll: '                r: 0.851';
    crtab.
  codigo nextPutAll: '                g: 0.851';
    crtab.
  codigo nextPutAll: '                b: 0.851).';
    crtab.
  codigo nextPutAll: '    nm setLabel: ' 'Navegador de entidades ' , classe nome , ' ''.';
    crtab.
  codigo nextPutAll: 'self iniciarSessao.    ↑ self'.
  ↑ codigo contents!
```

codigoParaRecuperarValorNaTelaDoAtributo: *umNomeDeAtributo*

```
  ↑ '(corpo clientWithTag: ' '' , umNomeDeAtributo , ' '') text'!
```

gerar

```
  | aClasse |
  super gerar.
  aClasse ← Smalltalk at: nomeClasseGerar asSymbol.
  aClasse compile: self codigoParaOpenInSystemWindow classified: 'private'!
```

Object subclass: OORDBMGeradorEsquemaRelacional

```
    Category: OORDBM-Geradores
    Instance variable names: grafo
```

Esta classe gerar a representacao em memoria (OORDBMTabela e OORDBMColuna) correspondente ao banco de dados que servira como suporte a persistncia das classes da hierarquia

Class protocol

Methods for category: as yet unclassified

retorneSQLParaEntidade: *umaEntidade*
```
self subclassResponsibility.
```
tipoDadosPara: *umTipo*
```
self subclassResponsibility.
```

Instance protocol

Methods for category: as yet unclassified

visoes
```
↑ self subClassResponsibility.
```

Methods for category: gerar

tabelas
```
"Retorna um Dictionary contendo as tabelas"
↑ self subClassResponsibility.
```

Methods for category: private

completaRelacionamentoNAN: *relacionamento* **considerando:** *tabelas*
```
| entOrigem entDestino coluna tabela |
entOrigem ← relacionamento nomeAtributoOrigem.
entDestino ← relacionamento nomeAtributoDestino.
tabela ← OORDBMTabela new.
tabela nome: entOrigem , '←' , entDestino.
coluna ← OORDBMColuna new.
coluna nome: relacionamento classeOrigem nome.
coluna tipo: 'Inteiro'.
coluna
   chaveEstrangeira: ((tabelas at: relacionamento classeOrigem nome) colunas at: 'intid').
tabela adicionaColuna: coluna.
tabela chave: coluna.
coluna ← OORDBMColuna new.
coluna nome: relacionamento classeDestino nome.
coluna tipo: 'Inteiro'.
coluna
   chaveEstrangeira: ((tabelas at: relacionamento classeDestino nome) colunas at: 'intid').
tabela adicionaColuna: coluna.
tabela chave: coluna.
tabelas at: tabela nome put: tabela.
↑ tabelas!
```

completaTabelaComRelacionamento1A1: *tabela* **segundoRelacionamento:** *relacionamento* **considerando:** *tabelas*
```
| entOrigem entDest coluna |
entOrigem ← relacionamento classeOrigem.
entDest ← relacionamento classeDestino.
(relacionamento nomeAtributoOrigem ≠ ' '
      and: [tabela nome = relacionamento classeOrigem nome])
   ifTrue: [coluna ← OORDBMColuna new nome: relacionamento nomeAtributoOrigem;
            tipo: 'Inteiro'.
      coluna
         chaveEstrangeira: ((tabelas at: entDest nome) colunas at: 'intid').
      tabela adicionaColuna: coluna].
((relacionamento nomeAtributoDestino ≠ ' ') and: [tabela nome = relacionamento classeDestino nome])
      ifTrue: [coluna ← OORDBMColuna new nome: relacionamento nomeAtributoDestino;
               tipo: 'Inteiro'.
         coluna
            chaveEstrangeira: ((tabelas at: entDest nome) colunas at: 'intid').
         tabela adicionaColuna: coluna]!
```

completaTabelaComRelacionamento1AN: *tabela* **segundoRelacionamento:** *relacionamento* **considerando:** *tabelas*

```
  | entOrigem entDest coluna |
  entOrigem ← relacionamento classeOrigem.
  entDest ← relacionamento classeDestino.
  (relacionamento grauOrigem ≠ 1
      and: [entDest nome = tabela nome])
    ifTrue: [coluna ← OORDBMColuna new nome: relacionamento nomeAtributoDestino;
              tipo: 'Inteiro'.
      coluna
        chaveEstrangeira: ((tabelas at: entOrigem nome) colunas at: 'intid').
      tabela adicionaColuna: coluna].
  (relacionamento grauDestino ≠ 1
      and: [entOrigem nome = tabela nome])
    ifTrue: [coluna ← OORDBMColuna new nome: relacionamento nomeAtributoOrigem;
              tipo: 'Inteiro'.
      coluna
        chaveEstrangeira: ((tabelas at: entDest nome) colunas at: 'intid').
      tabela adicionaColuna: coluna]!
```

completaTabelasComRelacionamentos1A1: *tabelas*

```
  | entidade |
  tabelas values
    do: [:tabela |
      entidade ← grafo classeNomeada: tabela nome.
      (grafo relacionamentosDe: entidade)
        do: [:relacionamento | (relacionamento grauOrigem = 1
            and: [relacionamento grauDestino = 1])
          ifTrue: [self completaTabelaComRelacionamento1A1: tabela segundoRelacionamento: relacionamento considerando:
tabelas] ]].
  ↑ tabelas!
```

completaTabelasComRelacionamentos1AN: *tabelas*

```
  | entidade |
  tabelas values
    do: [:tabela |
      entidade ← grafo classeNomeada: tabela nome.
      (grafo relacionamentosDe: entidade)
        do: [:relacionamento | ((relacionamento grauOrigem = 1
              and: [relacionamento grauDestino ≠ 1])
            or: [relacionamento grauOrigem = 1
              and: [relacionamento grauDestino ≠ 1] ])
          ifTrue: [self completaTabelaComRelacionamento1AN: tabela segundoRelacionamento: relacionamento conside-
rando: tabelas]
        ] ].
  ↑ tabelas!
```

completaTabelasComRelacionamentos: *tabelas*

```
  | entidade |
  tabelas values
    do: [:tabela |
      entidade ← grafo classeNomeada: tabela nome.
      (grafo relacionamentosDe: entidade)
        do: [:relacionamento | (relacionamento grauOrigem = 1
            and: [relacionamento grauDestino = 1])
          ifTrue: [self completaTabelaComRelacionamento1A1: tabela segundoRelacionamento: relacionamento considerando:
tabelas]
          ifFalse: [((relacionamento grauOrigem = 1
              and: [relacionamento grauDestino ≠ 1])
            or: [relacionamento grauOrigem = 1
              and: [relacionamento grauDestino ≠ 1] ])
            ifTrue: [self completaTabelaComRelacionamento1AN: tabela segundoRelacionamento: relacionamento conside-
rando: tabelas]
            ifFalse: [self completaRelacionamentoNAN: relacionamento considerando: tabelas] ]] ].
  ↑ tabelas!
```

geraTabelaBasicaSemRelacionamentos: *entidade*

```
  ↑ self geraTabelaBasicaSemRelacionamentos: entidade gerandoChaveSequencialmente: true.!
```

geraTabelaBasicaSemRelacionamentos: *entidade* **gerandoChaveSequencialmente:** *umBooleano*

```
"Gera a tabela basica para a entidade fornecida. Assume como tipos das
colunas os mesmos informados na entidade — nao converte os tipos nao
basicos em inteiro para os relacionamentos (para tal, veja
geraRelacionamentos)"
| tabela coluna |
tabela ← OORDBMTabela new.
tabela nome: entidade nome.
entidade atributos
   do: [:atributo | tabela adicionaColuna: (OORDBMColuna new nome: atributo nome;
         tipo: atributo tipo nome)].
umBooleano
   ifTrue: [coluna ← OORDBMColuna new nome: 'intid';
            tipo: 'Chave']
   ifFalse: [coluna ← OORDBMColuna new nome: 'intid';
            tipo: 'Inteiro'].
tabela adicionaColuna: coluna.
tabela chave: coluna.
↑ tabela!
```

hierarquia

```
↑ grafo.
```

hierarquia: *umGrafo*

```
grafo ← umGrafo.
```

OORDBMGeradorEsquemaRelacional subclass: OORDBMGeradorEsquemaRelacionalClassesConcretas

Category: OORDBM-Geradores

Esta classe gerar a representacao em memoria (OORDBMTabela e OORDBMColuna) correspondente ao banco de dados que servira como suporte a persistncia das classes da hierarquia

Instance protocol

Methods for category: as yet unclassified

visoes

```
↑ Dictionary new.!
```

Methods for category: gerar

tabelas

```
↑ self completaTabelasComRelacionamentos: self geraTabelasSemRelacoes!
```

Methods for category: private

geraTabelasSemRelacoes

```
| entidades tabelas tabela |
entidades ← (grafo classes) select: [:entidade | (OORDBMTiposBasicos ehTipoBasico: (entidade nome)) not. ].
tabelas ← Dictionary new.
entidades do: [:entidade |
   tabela ← self geraTabelaBasicaSemRelacionamentos: entidade.
   tabelas at: (tabela nome) put: tabela.
].
↑ tabelas.
```

hierarquia: *umGrafo*

```
grafo ← (umGrafo expandaAtributosParaFilhos).
```

OORDBMGeradorEsquemaRelacional subclass: OORDBMGeradorEsquemaRelacionalHierarquiaCompleta

Category: OORDBM-Geradores

Esta classe gerar a representacao em memoria (OORDBMTabela e OORDBMColuna) correspondente ao banco de dados que servira como suporte a persistncia das classes da hierarquia

Instance protocol

Methods for category: as yet unclassified

visoes

```
↑ Dictionary new.!
```

Methods for category: gerar

tabelas

```
| tabelas |
tabelas ← Dictionary new.
tabelas ← self
        completaTabelasComRelacionamentos: (self
            geraTabelasDescendentes: (self geraTabelasSuperiores: tabelas)).
tabelas ← self
        completaTabelasComRelacionamentos: tabelas.
tabelas explore.
↑ tabelas!
```

Methods for category: private

geraTabelasDescendentes: *tabelas*

```
| classes tabela classesFilhas |
classes ← (grafo classesSuperiores
        select: [:entidade | (OORDBMTiposBasicos ehTipoBasico: entidade nome) not]) asOrderedCollection.
[classes isEmpty]
   whileFalse: [classesFilhas ← grafo classesFilhas: classes removeFirst.
      classes addAll: classesFilhas.
      classesFilhas
         do: [:classe |
            tabela ← self geraTabelaBasicaSemRelacionamentos: classe gerandoChaveSequencialmente: false.
            tabelas at: tabela nome put: tabela.
            (tabela colunas at: 'intid')
               chaveEstrangeira: ((tabelas at: (grafo classePai: classe) nome) colunas at: 'intid')] ].
↑ tabelas!
```

geraTabelasSuperiores: *tabelas*

```
| classes tabela |
classes ← grafo classesSuperiores
        select: [:entidade | (OORDBMTiposBasicos ehTipoBasico: entidade nome) not].
classes
   do: [:classe |
      tabela ← self geraTabelaBasicaSemRelacionamentos: classe gerandoChaveSequencialmente: true.
      tabela adicionaColuna: (OORDBMColuna new nome: 'tpCtrlInt';
            tipo: 'Texto').
      tabelas at: tabela nome put: tabela].
↑ tabelas!
```

OORDBMGeradorEsquemaRelacional subclass: OORDBMGeradorEsquemaRelacionalTabelaUnica

Category: OORDBM-Geradores

Esta classe gerar a representacao em memoria (OORDBMTabela e OORDBMColuna) correspondente ao banco de dados que servira como suporte a persistncia das classes da hierarquia

Instance protocol

Methods for category: as yet unclassified

montaTabelasParaVisoes

```
| tabelas tabelasSuperiores grafoCompleto novoGerador|
novoGerador ← (OORDBMGeradorEsquemaRelacionalClassesConcretas new hierarquia: grafo).
tabelas ← novoGerador geraTabelasSemRelacoes.
tabelas ← novoGerador completaTabelasComRelacionamentos1A1: tabelas.
tabelas ← novoGerador completaTabelasComRelacionamentos1AN: tabelas.
tabelasSuperiores ← self geraTabelasSuperiores: Dictionary new.
grafoCompleto ← grafo expandaAtributosParaFilhos.
tabelas
   do: [:tabela |
      | tabelaMaisSuperior |
      tabelaMaisSuperior ← tabelasSuperiores
            detect: [:tabelaPesq | tabelaPesq nome = ((grafoCompleto classeMaisSuperiorDe: (grafoCompleto classeNomeada:
tabela nome)) nome)].
      tabela colunas values
         do: [:coluna | coluna tabela: tabelaMaisSuperior].
   ].
↑ tabelas!
```

visoes

```
| tabelas visoes |
tabelas ← self montaTabelasParaVisoes.
visoes ← Dictionary new.
tabelas
   do: [:tabela |
      | visao |
      visao ← OORDBMVisao new.
      visao nome: 'Objetos', tabela nome.
      tabela colunas
         do: [:coluna | visao adicionarColuna: coluna sobOAlias: coluna nome].
      visao clausulaSQLParaConsulta: 'tpCtrlInt = ' '' , tabela nome , ' '' '.
      visoes at: visao nome put: visao].
↑ visoes!
```

Methods for category: gerar

tabelas

```
| tabelas |
tabelas ← Dictionary new.
tabelas ← self
      completaTabelasComRelacionamentos: (self
            geraTabelasDescendentes: (self geraTabelasSuperiores: tabelas)).
↑ tabelas!
```

Methods for category: private

completaRelacionamentoNAN: *relacionamento* considerando: *tabelas*

```
| entOrigem entDestino coluna tabela |
entOrigem ← relacionamento nomeAtributoOrigem.
entDestino ← relacionamento nomeAtributoDestino.
tabela ← OORDBMTabela new.
tabela nome: entOrigem , '←' , entDestino.
coluna ← OORDBMColuna new.
coluna nome: relacionamento classeOrigem nome.
coluna tipo: 'Inteiro'.
coluna
   chaveEstrangeira: ((tabelas at: (grafo classeMaisSuperiorDe: (relacionamento classeOrigem)) nome) colunas at: 'intid').
tabela adicionaColuna: coluna.
tabela chave: coluna.
coluna ← OORDBMColuna new.
```

```
  coluna nome: relacionamento classeDestino nome.
  coluna tipo: 'Inteiro'.
  coluna
    chaveEstrangeira: ((tabelas at: (grafo classeMaisSuperiorDe: (relacionamento classeDestino)) nome) colunas at: 'intid').
  tabela adicionaColuna: coluna.
  tabela chave: coluna.
  tabelas at: tabela nome put: tabela.
  ↑ tabelas!
```

completaTabelaComRelacionamento1A1: *tabela* **segundoRelacionamento:** *relacionamento* **considerando:** *tabelas*

```
  | entOrigem entDest coluna |
  entOrigem ← relacionamento classeOrigem.
  entDest ← relacionamento classeDestino.
  (relacionamento nomeAtributoOrigem ≠ ' '
      and: [tabela nome = (grafo classeMaisSuperiorDe: relacionamento classeOrigem) nome])
    ifTrue: [coluna ← OORDBMColuna new nome: relacionamento nomeAtributoOrigem;
              tipo: 'Inteiro'.
      coluna
        chaveEstrangeira: ((tabelas at: (grafo classeMaisSuperiorDe: entDest) nome) colunas at: 'intid').
      tabela adicionaColuna: coluna].
  (relacionamento nomeAtributoDestino ≠ ' '
      and: [tabela nome = (grafo classeMaisSuperiorDe: relacionamento classeDestino) nome])
    ifTrue: [coluna ← OORDBMColuna new nome: relacionamento nomeAtributoDestino;
              tipo: 'Inteiro'.
      coluna
        chaveEstrangeira: ((tabelas at: (grafo classeMaisSuperiorDe: entDest) nome) colunas at: 'intid').
      tabela adicionaColuna: coluna]!
```

completaTabelaComRelacionamento1AN: *tabela* **segundoRelacionamento:** *relacionamento* **considerando:** *tabelas*

```
  | entOrigem entDest coluna |
  entOrigem ← relacionamento classeOrigem.
  entDest ← relacionamento classeDestino.
  (grafo classeMaisSuperiorDe: entOrigem) nome = tabela nome
    ifTrue: [coluna ← OORDBMColuna new nome: relacionamento nomeAtributoOrigem;
              tipo: 'Inteiro'.
      coluna
        chaveEstrangeira: ((tabelas at: (grafo classeMaisSuperiorDe: entDest) nome) colunas at: 'intid').
      tabela adicionaColuna: coluna]!
```

completaTabelasComRelacionamentos: *tabelas*

```
  | entidade |
  tabelas values
    do: [:tabela |
      entidade ← grafo classeNomeada: tabela nome.
      (grafo relacionamentosIncluindoDescendentesDe: entidade)
        do: [:relacionamento | (relacionamento grauOrigem = 1
              and: [relacionamento grauDestino = 1])
            ifTrue: [self completaTabelaComRelacionamento1A1: tabela segundoRelacionamento: relacionamento considerando:
tabelas]
            ifFalse: [((relacionamento grauOrigem = 1
                    and: [relacionamento grauDestino ≠ 1])
                  or: [relacionamento grauOrigem = 1
                      and: [relacionamento grauDestino ≠ 1] ])
                ifTrue: [self completaTabelaComRelacionamento1AN: tabela segundoRelacionamento: relacionamento conside-
rando: tabelas]
                ifFalse: [self completaRelacionamentoNAN: relacionamento considerando: tabelas] ]] ].
  ↑ tabelas!
```

geraTabelasDescendentes: *tabelas*

```
  | classes tabela classesFilhas classe tabelaFilha |
  classes ← (grafo classesSuperiores
        select: [:entidade | (OORDBMTiposBasicos ehTipoBasico: entidade nome) not]) asOrderedCollection.
  [classes isEmpty]
    whileFalse: [classe ← classes removeFirst.
      classesFilhas ← grafo classesDescendentes: classe.
```

```
        tabela ← tabelas at: classe nome.
        classesFilhas
          do: [:classeFilha |
            tabelaFilha ← self geraTabelaBasicaSemRelacionamentos: classeFilha gerandoChaveSequencialmente: false.
            tabelaFilha colunas
              do: [:coluna |
                (tabela colunas keys includes: coluna nome)
                  ifFalse: [tabela colunas at: coluna nome put: coluna].
                "tabelas at: tabelaFilha nome put: tabelaFilha"] ].
].
  ↑ tabelas!
```

geraTabelasSuperiores: *tabelas*

```
  | classes tabela |
  classes ← grafo classesSuperiores
          select: [:entidade | (OORDBMTiposBasicos ehTipoBasico: entidade nome) not].
  classes
    do: [:classe |
      tabela ← self geraTabelaBasicaSemRelacionamentos: classe gerandoChaveSequencialmente: true.
      tabela adicionaColuna: (OORDBMColuna new nome: 'tpCtrlInt';
            tipo: 'Texto').
      tabelas at: tabela nome put: tabela].
  ↑ tabelas!
```

Object subclass: OORDBMGeradorPersistencia

Category: OORDBM-Geradores

Instance variable names: grafo tabelas prefixo

Esta classe e responsavel por criar o suporte a persistencia das classes

Instance protocol

Methods for category: accessing

prefixo

```
  "Prefixo a ser adicionado ao nome do gerenciador de persistencia e tambem do descritor de persistencia"
  ↑ prefixo
```

prefixo: *umPrefixo*

```
  prefixo ← umPrefixo
```

tabelas

```
  "Tabelas onde os objetos serao persistidos (colecao de OORDBMTabela)"
  ↑ tabelas!
```

tabelas: *anObject*

```
  "Set the value of tabelas"
  tabelas ← anObject
```

Methods for category: converting

gerar

```
  "Deve gerar o suporte a persistencia da hierarquia de classes informada"
  ↑ self subClassResponsibility!
```

Methods for category: private

hierarquia

```
  "Hierarquia das classes do sistema a serem persistidas"
  ↑ grafo!
```

hierarquia: *umGrafo*

```
  grafo ← umGrafo.
```

OORDBMGeradorPersistencia subclass: OORDBMGeradorPersistenciaGLORP

Category: OORDBM-Geradores

Esta classe e responsavel por criar o suporte a persistencia das classes usando o GLORP como framework de persistencia.

Class protocol

Methods for category: as yet unclassified

initialize

```
dados ← Dictionary new.
dados at: 'Chave' put: 'serial'.
dados at: 'Inteiro' put: 'integer'.
dados at: 'Texto' put: 'varchar'.
dados at: 'Racional' put: 'float'.
dados at: 'Real' put: 'float'.
dados at: 'Data' put: 'date'.
dados at: 'Hora' put: 'time'.
dados at: 'ValorMonetario' put: 'numeric'.
dados at: 'Imagem' put: 'blob'.
dados at: 'Qualquer' put: 'blob'.
dados at: 'SimNao' put: 'boolean'!
```

tipoPara: *umTipo*

```
dados isNil ifTrue: [ self initialize. ].
↑ dados at: umTipo.
```

Instance protocol

Methods for category: gerar

gerar

```
"Gera o suporte a persistencia do projeto. Inclui gerar as classes do
GLORP encarregadas de cuidar da persistencia e gerar os metodos de
classes responsaveis por manter os IDs das classes na persistncia da
hierarquia"
self gerarDescritor!
```

Methods for category: private

gerarBaseDescritor

```
"Gera o corpo basico da classe responsavel pela definicao da persistncia"
| codigo classes |
OORDBMGeradorClasses
   gerarClasse: self nomeDescritor
   superClasse: 'DescriptorSystem'
   categoria: grafo categoria.
classes ← grafo classes
        select: [:entidade | (OORDBMTiposBasicos ehTipoBasico: entidade nome) not].
codigo ← TextStream with: 'constructAllClasses '.
codigo cr; crtab; nextPutAll: '↑ ((super constructAllClasses) '.
classes
   do: [:classe | codigo nextPutAll: 'add: ' , classe nome , '; '].
codigo nextPutAll: ' yourself).'.
(Smalltalk at: self nomeDescritor asSymbol)
   compile: codigo contents
   classified: 'descritores'.
codigo ← TextStream with: 'allTableNames '.
codigo cr; crtab; nextPutAll: ' ↑ #( '.
tabelas asOrderedCollection
   do: [:tabela | codigo nextPutAll: ' '' ' , tabela nome , ' '' '].
codigo nextPutAll: ').'.
(Smalltalk at: self nomeDescritor asSymbol)
   compile: codigo contents
   classified: 'descritores'!
```

gerarClassesDescritor

```
↑ self subClassResponsibility
```

gerarDescritor

```
"Gera o descritor de persistncia"
self gerarBaseDescritor.
self gerarTabelasDescritor.
```

```
self gerarClassesDescritor!
```

gerarGerenciadorPersistencia

```
"Cria a classe responsavel por gerenciar a persistncia de objetos"
OORDBMGeradorClasses
    gerarClasse: self nomeGerenciador
    categoria: grafo categoria
    variaveis: 'accessor sessao descritor usuarioBD senhaBD enderecoBD nomeBD'.
self gerarGerenciadorPersistenciaMetodosBD.
self gerarGerenciadorPersistenciaMetodosSessao.
self gerarGerenciadorPersistenciaMetodosRegistrar.
self gerarGerenciadorPersistenciaMetodosRecuperar.
self gerarGerenciadorPersistenciaMetodosExcluir!
```

gerarGerenciadorPersistenciaMetodosBD

```
"Metodo Metodo do gerenciador de persistncia para iniciar a conexao com o banco de dados"
| classe codigo |
classe ← Smalltalk at: self nomeGerenciador asSymbol.
codigo ← TextStream with: 'conectarBancoDeDados '.
codigo cr; crtab; nextPutAll: 'accessor ← DatabaseAccessor forLogin: (Login new database: PostgreSQLPlatform new; username:
usuarioBD; password: senhaBD; connectString: enderecoBD , ' ':' ' , ' '5432' ' , ' '←' ' , nomeBD).'; crtab.
codigo nextPutAll: 'accessor login.';
    crtab.
classe compile: codigo contents classified: 'sessao'!
```

gerarGerenciadorPersistenciaMetodosExcluir

```
"Metodo do gerenciador de persistncia para excluir um objeto do banco de dados"
| classe codigo |
classe ← Smalltalk at: self nomeGerenciador asSymbol.
codigo ← TextStream with: 'excluir: umObjeto '.
codigo cr; crtab; nextPutAll: 'sessao isNil ifTrue: ['; crtab; tab.
codigo nextPutAll: 'self error: ' 'Voce nao esta em uma sessao com o banco de dados' '.';
    crtab;
    nextPutAll: '].';
    cr;
    cr.
codigo nextPutAll: 'sessao delete: umObjeto.'.
classe compile: codigo contents classified: 'registro'!
```

gerarGerenciadorPersistenciaMetodosRecuperar

```
"Metodos do gerenciador de persistencia para recuperar elementos do banco de dados"
| classe codigo |
classe ← Smalltalk at: self nomeGerenciador asSymbol.
"Recuperar varios objetos segundo condicao"
codigo ← TextStream with: 'recuperarEntidadesDoTipo: umaClasse segundo: umaCondicao '.
codigo cr; crtab; nextPutAll: 'sessao isNil ifTrue: ['; crtab; tab.
codigo nextPutAll: 'self error: ' 'Voce nao esta em uma sessao com o banco de dados' '.';
    crtab;
    nextPutAll: '].';
    cr;
    cr;
    tab.
codigo nextPutAll: '↑ sessao readManyOf: umaClasse where: umaCondicao.'.
classe compile: codigo contents classified: 'registro'.
"Recuperar varios objetos segundo condicao"
codigo ← TextStream with: 'recuperarEntidadesDoTipo: umaClasse '.
codigo cr; crtab; nextPutAll: 'sessao isNil ifTrue: ['; crtab; tab.
codigo nextPutAll: 'self error: ' 'Voce nao esta em uma sessao com o banco de dados' '.';
    crtab;
    nextPutAll: '].';
    cr;
    cr;
    tab.
codigo nextPutAll: '↑ sessao readManyOf: umaClasse.'.
classe compile: codigo contents classified: 'registro'.
"Recuperar objeto segundo condicao"
codigo ← TextStream with: 'recuperarEntidadeDoTipo: umaClasse segundo: umaCondicao '.
```

```
codigo cr; crtab; nextPutAll: 'sessao isNil ifTrue: ['; crtab; tab.
codigo nextPutAll: 'self error: ' 'Voce nao esta em uma sessao com o banco de dados' '.';
    crtab;
    nextPutAll: '].';
    cr;
    cr;
    tab.
codigo nextPutAll: '↑ sessao readOneOf: umaClasse where: umaCondicao.'.
classe compile: codigo contents classified: 'registro'!
```

gerarGerenciadorPersistenciaMetodosRegistrar

```
"Metodos do gerenciador de persistencia para registrar objetos novos ou jah existentes na sessao"
| classe codigo |
classe ← Smalltalk at: self nomeGerenciador asSymbol.
"Registrar um novo objeto"
codigo ← TextStream with: 'registrar: umObjeto '.
codigo cr; crtab; nextPutAll: 'sessao isNil ifTrue: ['; crtab; tab.
codigo nextPutAll: 'self error: ' 'Voce nao esta em uma sessao com o banco de dados' '.';
    crtab;
    nextPutAll: '].';
    cr;
    cr.
codigo nextPutAll: 'sessao register: umObjeto.';
    crtab;
    nextPutAll: 'sessao refresh: umObjeto'.
classe compile: codigo contents classified: 'registro'.
"Registar um objeto"
codigo ← TextStream with: 'registrarNovo: umObjeto '.
codigo cr; crtab; nextPutAll: 'sessao isNil ifTrue: ['; crtab; tab.
codigo nextPutAll: 'self error: ' 'Voce nao esta em uma sessao com o banco de dados' '.';
    crtab;
    nextPutAll: '].';
    cr;
    cr.
codigo nextPutAll: 'sessao registerAsNew: umObjeto.';
    crtab;
    nextPutAll: 'sessao refresh: umObjeto'.
classe compile: codigo contents classified: 'registro'!
```

gerarGerenciadorPersistenciaMetodosSessao

```
"Gera os metodos do gerencia de persistencia responsaveis por iniciar e terminar uma sessao de persistencia"
| classe codigo |
classe ← Smalltalk at: self nomeGerenciador asSymbol.
"Metodo de inicio de sessao"
codigo ← TextStream with: 'iniciarSessao '.
codigo cr; crtab; nextPutAll: 'self conectarBancoDeDados.'; crtab; nextPutAll: 'descritor ← ' , self nomeDescritor , ' new.'; crtab.
codigo nextPutAll: 'sessao ← GlorpSession new.';
    crtab;
    nextPutAll: 'sessao system: descritor.';
    crtab;
    nextPutAll: 'sessao accessor: accessor.';
    crtab.
codigo nextPutAll: 'sessao beginUnitOfWork.'.
classe compile: codigo contents classified: 'sessao'.
"Metodo de fim de sessao"
codigo ← TextStream with: 'terminarSessao '.
codigo cr; crtab; nextPutAll: 'sessao isNil ifTrue: ['; crtab; tab.
codigo nextPutAll: 'self error: ' 'Voce nao esta em uma sessao com o banco de dados' '.';
    crtab.
codigo nextPutAll: '].';
    cr;
    crtab.
codigo nextPutAll: 'sessao commitUnitOfWork.';
    crtab;
    nextPutAll: 'accessor logout.';
    cr;
    crtab.
```

```
    codigo nextPutAll: 'sessao ← nil.';
        crtab;
        nextPutAll: 'accessor ← nil.'.
    classe compile: codigo contents classified: 'sessao'.
    "Metodo de gravar sessao e continuar"
    codigo ← TextStream with: 'gravarEContinuarSessao '.
    codigo cr; crtab; nextPutAll: 'sessao isNil ifTrue: ['; crtab; tab.
    codigo nextPutAll: 'self error: ' 'Voce nao esta em uma sessao com o banco de dados' '.';
        crtab.
    codigo nextPutAll: ']';
        cr;
        crtab.
    codigo nextPutAll: 'sessao commitUnitOfWorkAndContinue.'.
    classe compile: codigo contents classified: 'sessao'!
```

gerarTabelaDescritor: *tabela*

```
    "Gera um descritor do GLORP para a tabela informada (OORDBMTabela)"
    | codigo |
    codigo ← 'tableFor' , tabela nome asUppercase , ': umDescritor | '.
    tabela colunas
        do: [:coluna | codigo ← codigo , coluna nome asLowercase, ' '].
    codigo ← codigo , '| ' , Character cr asString , Character cr asString.
    tabela colunas
        do: [:coluna |
            codigo ← codigo , coluna nome asLowercase , ' ← (umDescritor createFieldNamed: ' '' , coluna nome , ' '' type: (platform '
                    , (self species tipoPara: coluna tipo) , ')) '.
            (tabela chaves includes: coluna)
                ifTrue: [codigo ← codigo , ' bePrimaryKey'].
            codigo ← codigo , '. ' , Character cr asString].
    tabela colunas
        do: [:coluna | coluna chaveEstrangeira isNil
                ifFalse: [codigo ← codigo , ' umDescritor addForeignKeyFrom: ' , coluna nome asLowercase , ' to: ((self tableNamed: ' ''
, coluna chaveEstrangeira tabela nome asUppercase , ' '') fieldNamed: ' 'intid' ' ).' , Character cr asString] ].
    (Smalltalk at: self nomeDescritor asSymbol)
        compile: codigo
        classified: 'tabelas'!
```

gerarTabelasDescritor

```
    "Gera no descritor de persistencia as descricoes (metodos) para as tabelas envolvidas (colecao de OORDBMTabela)"
    tabelas values
        do: [:tabela | self gerarTabelaDescritor: tabela]!
```

nomeDescritor

```
    "Retorna o nome da classe que sera o descritor de persistencia para o sistema gerado"
    ↑ prefixo , 'DescritorPersistencia'!
```

nomeGerenciador

```
    "Retorna o nome da classe que sera o gerenciador de persistencia para o sistema gerado"
    ↑ prefixo , 'GerenciadorPesistencia'!
```

OORDBMGeradorPersistenciaGLORP subclass: OORDBMGeradorPersistenciaGLORPClassesConcretas

Category: OORDBM-Geradores

Esta classe e responsavel por criar o suporte a persistencia das classes usando o GLORP como framework de persistencia.

Instance protocol

Methods for category: private

gerarBaseClasseDescritor: *umaTabela*

```
| codigo |
codigo ← 'descriptorFor' , grafo prefixo , umaTabela nome , ': umDescritor | tabela | ' , (Character cr asString) , 'tabela ← self tableNamed: ' '' , umaTabela nome , ' ''.' , (Character cr asString) , ' umDescritor table: tabela. '.
umaTabela colunas do: [ :coluna |
   codigo ← codigo , 'umDescritor addMapping: (DirectMapping from: #' , coluna nome , ' to: (tabela fieldNamed: ' '' , coluna nome , ' '')).' , (Character cr asString).
].
↑ codigo.
```

gerarClasseDescritor: *umaTabela*

```
| codigo |
codigo ← self gerarBaseClasseDescritor: umaTabela.
codigo ← (self gerarRelacionamentos1a1De: umaTabela continuando: codigo).
codigo ← (self gerarRelacionamentos1aNDe: umaTabela continuando: codigo).
codigo ← (self gerarRelacionamentosNaNDe: umaTabela continuando: codigo).
(Smalltalk at: (self nomeDescritor asSymbol)) compile: codigo classified: 'descritores'.
```

gerarClassesDescritor

```
| classesADescrever |
classesADescrever ← tabelas asOrderedCollection select: [:tabela |
   (grafo classeNomeada: (tabela nome)) isNil not
].
classesADescrever do: [:classe |
   self gerarClasseDescritor: classe.
].
```

gerarRelacionamentos1a1De: *umaTabela* **continuando:** *umCodigo*

```
| codigo relacionamentos nomeAtributo entObjetivo |
codigo ← umCodigo.
relacionamentos ← (grafo
        relacionamentosDe: (grafo classeNomeada: umaTabela nome))
        select: [:rel | rel grauOrigem = 1
             and: [rel grauDestino = 1] ].
relacionamentos
   do: [:rel | rel classeOrigem nome = umaTabela nome
        ifTrue: [nomeAtributo ← rel nomeAtributoOrigem.
           entObjetivo ← (umaTabela colunas at: nomeAtributo) chaveEstrangeira tabela nome.
           codigo ← codigo , ' umDescritor addMapping: ((OneToOneMapping new) attributeName: #' , nomeAtributo , '; referenceClass: ' , entObjetivo , '; mappingCriteria: (Join from: (tabela fieldNamed: ' '' , nomeAtributo , ' '') to: ((self tableNamed: ' '' , entObjetivo , ' '') fieldNamed: ' 'intid' ''))).'] ].
↑ codigo!
```

gerarRelacionamentos1aNDe: *umaTabela* **continuando:** *umCodigo*

```
| codigo relacionamentos nomeAtributo |
codigo ← umCodigo.
relacionamentos ← (grafo
        relacionamentosDe: (grafo classeNomeada: umaTabela nome))
        select: [:rel | ((rel grauOrigem = 1) not
              and: [rel grauDestino = 1])
            or: [rel grauOrigem = 1
                 and: [(rel grauDestino = 1) not] ]].
relacionamentos
   do: [:rel |
     rel classeDestino nome = umaTabela nome
        ifTrue: [nomeAtributo ← rel nomeAtributoDestino.
           codigo ← codigo , ' umDescritor addMapping: ((OneToOneMapping new) attributeName: #' , nomeAtributo , '; referenceClass: ' , rel classeOrigem nome , '; mappingCriteria: (Join from: (tabela fieldNamed: ' '' , nomeAtributo , ' '') to: ((self tableNamed: ' '' , rel classeOrigem nome , ' '') fieldNamed: ' 'intid' ''))).'].
        rel classeOrigem nome = umaTabela nome
        ifTrue: [nomeAtributo ← rel nomeAtributoOrigem.
```

codigo ← codigo , ' umDescritor addMapping: ((OneToManyMapping new) attributeName: #' , nomeAtributo , 's; referenceClass: ' , rel classeDestino nome, '; mappingCriteria: (Join from: (tabela fieldNamed: ' 'intid' ') to: ((self tableNamed: ' '' , rel classeDestino nome , ' '') fieldNamed: ' '' , rel nomeAtributoDestino , ' ''))).']].
↑ codigo!

gerarRelacionamentosNaNDe: *umaTabela* **continuando:** *umCodigo*
| codigo relacionamentos nomeAtributo |
codigo ← umCodigo.
relacionamentos ← (grafo
relacionamentosDe: (grafo classeNomeada: umaTabela nome))
select: [:rel | (rel grauOrigem = 1) not
and: [(rel grauDestino = 1) not]].
relacionamentos
do: [:rel | rel classeDestino nome = umaTabela nome
ifTrue: [nomeAtributo ← rel nomeAtributoDestino.
codigo ← codigo , ' umDescritor addMapping: ((ManyToManyMapping new) attributeName: #' , nomeAtributo , 's; referenceClass: ' , rel classeOrigem nome , '; mappingCriteria: (Join from: (tabela fieldNamed: ' 'intid' ') to: ((self tableNamed: ' '' , rel classeOrigem nome , '←' , rel classeDestino nome , ' '') fieldNamed: ' '' , rel classeDestino nome , ' ''))).']
ifFalse: [nomeAtributo ← rel nomeAtributoOrigem.
codigo ← codigo , ' umDescritor addMapping: ((ManyToManyMapping new) attributeName: #' , nomeAtributo , 's; referenceClass: ' , rel classeDestino nome , '; mappingCriteria: (Join from: (tabela fieldNamed: ' 'intid' ') to: ((self tableNamed: ' '' , rel classeOrigem nome , '←' , rel classeDestino nome , ' '') fieldNamed: ' '' , rel classeOrigem nome , ' ''))).']].
↑ codigo!

OORDBMGeradorPersistenciaGLORP subclass: OORDBMGeradorPersistenciaGLORPHierarquiaCompleta
Category: OORDBM-Geradores

Esta classe e responsavel por criar o suporte a persistencia das classes usando o GLORP como framework de persistencia.

Instance protocol

Methods for category: private

gerarBaseClasseDescritor: *umaClasse*

```
| codigo entidadesPai entidadeMaisSuperior idClasse |
codigo ← TextStream with: 'descriptorFor' , umaClasse nome , ': umDescritor | tabela | '.
entidadesPai ← grafo classesSuperioresDe: umaClasse.
entidadesPai isEmpty
   ifFalse: [entidadeMaisSuperior ← (entidadesPai intersection: grafo classesSuperiores) first.
      entidadesPai remove: entidadeMaisSuperior].
entidadeMaisSuperior isNil
   ifFalse: [codigo crtab; nextPutAll: 'umDescritor table: (self tableNamed: ' '' , entidadeMaisSuperior nome asUppercase , ' '').'].
entidadesPai
   do: [:entidade |
      codigo crtab; nextPutAll: 'umDescritor table: (self tableNamed: ' '' , entidade nome asUppercase , ' '').'.
      codigo crtab; nextPutAll: 'umDescritor addMultipleTableJoin: (Join from: ((self tableNamed: ' '' , entidadeMaisSuperior nome
asUppercase , ' '') fieldNamed: ' 'intid' ') to: ((self tableNamed: ' '' , entidade nome asUppercase , ' '') fieldNamed: ' 'intid' ')).'].
codigo crtab; nextPutAll: 'umDescritor table: (self tableNamed: ' '' , umaClasse nome asUppercase , ' '').'.
entidadeMaisSuperior isNil
   ifFalse: [entidadesPai add: entidadeMaisSuperior.
      codigo crtab; nextPutAll: 'umDescritor addMultipleTableJoin: (Join from: ((self tableNamed: ' '' , entidadeMaisSuperior nome
asUppercase , ' '') fieldNamed: ' 'intid' ') to: ((self tableNamed: ' '' , umaClasse nome asUppercase , ' '') fieldNamed: ' 'intid' ')).'].
entidadesPai add: umaClasse.
entidadesPai
   do: [:entidade | entidade atributos
         do: [:atributo |
            codigo crtab; nextPutAll: 'umDescritor addMapping: (DirectMapping from: #' , atributo nome.
            codigo nextPutAll: ' to: ((self tableNamed: ' '' , entidade nome , ' '') fieldNamed: ' '' , atributo nome , ' '')).'] ].
entidadeMaisSuperior isNil
   ifTrue: [entidadeMaisSuperior ← umaClasse].
codigo crtab; nextPutAll: 'umDescritor addMapping: (DirectMapping from: #intid '.
codigo nextPutAll: ' to: ((self tableNamed: ' '' , entidadeMaisSuperior nome asUppercase , ' '') fieldNamed: ' 'intid' ')).'.
idClasse ← (Smalltalk classNamed: self nomeDescritor) idsClasses indexOf: umaClasse nome asSymbol.
codigo cr; crtab; nextPutAll: '(self typeResolverFor: ' , umaClasse nome , ') register: umDescritor keyedBy: ' , idClasse asString ,
' field: ((self tableNamed: ' '' , entidadeMaisSuperior nome asUppercase , ' '') fieldNamed: ' 'tpCtrlInt' ').'.
↑ codigo contents!
```

gerarClasseDescritor: *umaClasse*

```
| codigo |
codigo ← self gerarBaseClasseDescritor: umaClasse.
codigo ← (self gerarRelacionamentos1a1De: umaClasse continuando: codigo).
codigo ← (self gerarRelacionamentos1aNDe: umaClasse continuando: codigo).
codigo ← (self gerarRelacionamentosNaNDe: umaClasse continuando: codigo).
(Smalltalk at: (self nomeDescritor asSymbol)) compile: codigo classified: 'descritores'.
codigo ← self gerarTypeResolverPara: umaClasse.
(Smalltalk at: (self nomeDescritor asSymbol)) compile: codigo classified: 'typeResolvers'.
```

gerarClassesDescritor

```
(grafo classes select: [:entidade | (OORDBMTiposBasicos ehTipoBasico: (entidade nome)) not ]) do: [:classe |
   self gerarClasseDescritor: classe.
].
```

gerarDescritor

```
super gerarDescritor.!
```

gerarRelacionamentos1a1De: *umaClasse* continuando: *umCodigo*

```
| codigo relacionamentos nomeAtributo entObjetivo entidadesValidas |
codigo ← TextStream with: umCodigo.
relacionamentos ← (grafo relacionamentosDe: umaClasse)
      select: [:rel | rel grauOrigem = 1
            and: [rel grauDestino = 1] ].
(grafo classesSuperioresDe: umaClasse)
   do: [:entidade | relacionamentos
         addAll: ((grafo relacionamentosDe: entidade)
               select: [:rel | rel grauOrigem = 1
                     and: [rel grauDestino = 1] ])].
```

```
    entidadesValidas ← OrderedCollection
            withAll: (grafo classesSuperioresDe: umaClasse).
    entidadesValidas add: umaClasse.
    relacionamentos
        do: [:rel | (rel nomeAtributoOrigem ≠ ' '
                and: [entidadesValidas includes: rel classeOrigem])
            ifTrue: [nomeAtributo ← rel nomeAtributoOrigem.
                entObjetivo ← rel classeDestino.
                codigo crtab; nextPutAll: ' umDescritor addMapping: ((OneToOneMapping new) attributeName: #' , nomeAtributo.
                codigo nextPutAll: '; referenceClass: ' , entObjetivo nome , '; mappingCriteria: (Join from:'.
                codigo nextPutAll: ' ((self tableNamed: ' '' , rel classeOrigem nome asUppercase , ' '') fieldNamed: ' '' , nomeAtributo ,
' '') to: ((self tableNamed: ' '' , entObjetivo nome , ' '') fieldNamed: ' 'intid' ''))).'] ].
    ↑ codigo contents!
```

gerarRelacionamentos1aNDe: *umaClasse* **continuando:** *umCodigo*

```
    | codigo relacionamentos nomeAtributo entidadesValidas |
    codigo ← TextStream with: umCodigo.
    relacionamentos ← (grafo relacionamentosDe: umaClasse)
            select: [:rel | ((rel grauOrigem = 1) not
                    and: [rel grauDestino = 1])
                or: [rel grauOrigem = 1
                        and: [(rel grauDestino = 1) not] ]].
    (grafo classesSuperioresDe: umaClasse)
        do: [:entidade | relacionamentos
            addAll: ((grafo relacionamentosDe: entidade)
                select: [:rel | ((rel grauOrigem = 1) not
                        and: [rel grauDestino = 1])
                    or: [rel grauOrigem = 1
                            and: [(rel grauDestino = 1) not] ]])].
    entidadesValidas ← OrderedCollection
            withAll: (grafo classesSuperioresDe: umaClasse).
    entidadesValidas add: umaClasse.
    relacionamentos
        do: [:rel |
            (((entidadesValidas includes: rel classeDestino)
                    and: [rel grauDestino = 1])
                and: [rel nomeAtributoDestino ≠ ' '])
                ifTrue: [nomeAtributo ← rel nomeAtributoDestino.
                    codigo crtab; nextPutAll: ' umDescritor addMapping: ((OneToOneMapping new) attributeName: #' , nomeAtributo.
                    codigo nextPutAll: '; referenceClass: ' , rel classeOrigem nome , '; mappingCriteria: (Join '.
                    codigo nextPutAll: ' from: ((self tableNamed: ' '' , umaClasse nome asUppercase , ' '') fieldNamed: ' '' , nomeAtributo ,
' '') to: ((self tableNamed: ' '' , rel classeOrigem nome asUppercase , ' '') fieldNamed: ' 'intid' ''))).'].
            (((entidadesValidas includes: rel classeOrigem)
                    and: [rel grauOrigem = 1])
                and: [rel nomeAtributoOrigem ≠ ' '])
                ifTrue: [nomeAtributo ← rel nomeAtributoOrigem.
                    codigo crtab; nextPutAll: ' umDescritor addMapping: ((OneToOneMapping new) attributeName: #' , nomeAtributo.
                    codigo nextPutAll: '; referenceClass: ' , rel classeDestino nome , '; mappingCriteria: (Join '.
                    codigo nextPutAll: ' from: ((self tableNamed: ' '' , umaClasse nome asUppercase , ' '') fieldNamed: ' '' , nomeAtributo ,
' '') to: ((self tableNamed: ' '' , rel classeDestino nome asUppercase , ' '') fieldNamed: ' 'intid' ''))).'].
            (((entidadesValidas includes: rel classeDestino)
                    and: [rel grauDestino ≠ 1])
                and: [rel nomeAtributoDestino ≠ ' '])
                ifTrue: [nomeAtributo ← rel nomeAtributoDestino.
                    codigo crtab; nextPutAll: ' umDescritor addMapping: ((OneToManyMapping new) attributeName: #' , nomeAtributo ,
'; referenceClass: '.
                    codigo nextPutAll: rel classeOrigem nome , '; mappingCriteria: (Join from: ((self tableNamed: ' '' , umaClasse nome
asUppercase , ' '') fieldNamed: ' '' , nomeAtributo , ' '') to: '.
                    codigo nextPutAll: '((self tableNamed: ' '' , rel classeOrigem nome asUppercase , ' '') fieldNamed: ' 'intid' ''))).'].
            (((entidadesValidas includes: rel classeOrigem)
                    and: [rel grauOrigem ≠ 1])
                and: [rel nomeAtributoOrigem ≠ ' '])
                ifTrue: [nomeAtributo ← rel nomeAtributoOrigem.
                    codigo crtab; nextPutAll: ' umDescritor addMapping: ((OneToManyMapping new) attributeName: #' , nomeAtributo ,
'; referenceClass: '.
                    codigo nextPutAll: rel classeDestino nome , '; mappingCriteria: (Join from: ((self tableNamed: ' '' , umaClasse nome
asUppercase , ' '') fieldNamed: ' '' , nomeAtributo , ' '') to: '.
```

```
            codigo nextPutAll: '((self tableNamed: ' '' , rel classeDestino nome asUppercase , ' '') fieldNamed: ' 'intid' '))).'] ].
    ↑ codigo contents!
```

gerarRelacionamentosNaNDe: *umaClasse* **continuando:** *umCodigo*

```
    | codigo relacionamentos nomeAtributo entidadesValidas |
    codigo ← TextStream with: umCodigo.
    relacionamentos ← (grafo relacionamentosDe: umaClasse)
            select: [:rel | (rel grauOrigem = 1) not
                and: [(rel grauDestino = 1) not] ].
    (grafo classesSuperioresDe: umaClasse)
        do: [:entidade | relacionamentos
            addAll: ((grafo relacionamentosDe: entidade)
                select: [:rel | (rel grauOrigem = 1) not
                    and: [(rel grauDestino = 1) not] ])].
    entidadesValidas ← OrderedCollection
            withAll: (grafo classesSuperioresDe: umaClasse).
    entidadesValidas add: umaClasse.
    relacionamentos
        do: [:rel | (entidadesValidas includes: rel classeDestino)
            ifTrue: [nomeAtributo ← rel nomeAtributoDestino.
                codigo crtab; nextPutAll: ' umDescritor addMapping: ((ManyToManyMapping new) attributeName: #' , nomeAtributo ,
'; referenceClass: '.
                codigo nextPutAll: rel classeOrigem nome , '; mappingCriteria: (Join from: ((self tableNamed: ' '' , umaClasse nome
asUppercase , ' '') fieldNamed: ' 'intid' ') '.
                codigo nextPutAll: ' to: ((self tableNamed: ' '' , rel nomeAtributoOrigem asUppercase , '←' , rel nomeAtributoDestino
asUppercase , ' '') fieldNamed: ' '' , rel classeDestino nome , ' ''))).']
            ifFalse: [nomeAtributo ← rel nomeAtributoOrigem.
                codigo crtab; nextPutAll: ' umDescritor addMapping: ((ManyToManyMapping new) attributeName: #' , nomeAtributo ,
'; referenceClass: '.
                codigo nextPutAll: rel classeDestino nome , '; mappingCriteria: (Join from: ((self tableNamed: ' '' , umaClasse nome
asUppercase , ' '') fieldNamed: ' 'intid' ') '.
                codigo nextPutAll: ' to: ((self tableNamed: ' '' , rel nomeAtributoOrigem asUppercase , '←' , rel nomeAtributoDestino
asUppercase , ' '') fieldNamed: ' '' , rel classeOrigem nome , ' ''))).'] ].
    ↑ codigo contents!
```

gerarTypeResolverPara: *umaClasse*

```
    | codigo |
    codigo ← TextStream with: 'typeResolverFor' , umaClasse nome.
    codigo cr; crtab; nextPutAll: '↑ FilteredTypeResolver forRootClass: ' , (grafo classeMaisSuperiorDe: umaClasse) nome , '.'.
    ↑ codigo contents!
```

OORDBMGeradorPersistenciaGLORP subclass: OORDBMGeradorPersistenciaGLORPTabelaUnica

Category: OORDBM-Geradores

Esta classe e responsavel por criar o suporte a persistencia das classes usando o GLORP como framework de persistencia e adotando a abordagem de uma tabela por hierarquia de classes.

Instance protocol

Methods for category: as yet unclassified

gerarBaseClasseDescritor: *umaClasse*

```
"Gera a parte basica (atributos) do descritor de persistencia da classe
informada "
| codigo entidadesPai entidadeMaisSuperior idClasse |
codigo ← TextStream with: 'descriptorFor' , umaClasse nome , ': umDescritor | tabela | '.
entidadesPai ← grafo classesSuperioresDe: umaClasse.
entidadeMaisSuperior ← grafo classeMaisSuperiorDe: umaClasse.
codigo crtab; nextPutAll: 'umDescritor table: (self tableNamed: ' '' , entidadeMaisSuperior nome asUppercase , ' '').'.
entidadesPai add: umaClasse.
entidadesPai
    do: [:entidade | entidade atributos
        do: [:atributo |
            codigo crtab; nextPutAll: 'umDescritor addMapping: (DirectMapping from: #' , atributo nome.
            codigo nextPutAll: ' to: ((self tableNamed: ' '' , entidadeMaisSuperior nome , ' '') fieldNamed: ' '' , atributo nome , ' '')).'] ].
codigo cr; crtab; nextPutAll: 'umDescritor addMapping: (DirectMapping from: #intid to: ((self tableNamed: ' '' , entidadeMaisSuperior nome , ' '') fieldNamed: ' 'intid' ')).'.
idClasse ← umaClasse nome.
codigo cr; crtab; nextPutAll: '(self typeResolverFor: ' , entidadeMaisSuperior nome , ') register: umDescritor keyedBy: ' '' , idClasse asString , ' '' field: ((self tableNamed: ' '' , entidadeMaisSuperior nome asUppercase , ' '') fieldNamed: ' 'tpCtrlInt' ').'.
↑ codigo contents!
```

gerarClasseDescritor: *umaClasse*

```
"Gera o descritor de persistencia para a classe informada"
| codigo |
codigo ← self gerarBaseClasseDescritor: umaClasse.
codigo ← self gerarRelacionamentos1a1De: umaClasse continuando: codigo.
codigo ← self gerarRelacionamentos1aNDe: umaClasse continuando: codigo.
codigo ← self gerarRelacionamentosNaNDe: umaClasse continuando: codigo.
(Smalltalk at: self nomeDescritor asSymbol)
    compile: codigo
    classified: 'descritores'.
codigo ← self gerarTypeResolverPara: umaClasse.
(Smalltalk at: self nomeDescritor asSymbol)
    compile: codigo
    classified: 'typeResolvers'!
```

gerarClassesDescritor

```
"Gera os metodos de descricao de como sera a persistncia das classes do sistema"
(grafo classes
    select: [:entidade | (OORDBMTiposBasicos ehTipoBasico: entidade nome) not])
    do: [:classe | self gerarClasseDescritor: classe]!
```

gerarRelacionamentos1a1De: *umaClasse* continuando: *umCodigo*

```
"Gera a definicao da persistencia do GLORP relacionada aos
relacionamentos 1−1 da classe"
| codigo relacionamentos nomeAtributo entObjetivo entidadesValidas orMaisSuperior dsMaisSuperior |
codigo ← TextStream with: umCodigo.
relacionamentos ← (grafo relacionamentosDe: umaClasse)
        select: [:rel | rel grauOrigem = 1
            and: [rel grauDestino = 1] ].
(grafo classesSuperioresDe: umaClasse)
    do: [:entidade | relacionamentos
        addAll: ((grafo relacionamentosDe: entidade)
            select: [:rel | rel grauOrigem = 1
                and: [rel grauDestino = 1] ])].
entidadesValidas ← OrderedCollection
        withAll: (grafo classesSuperioresDe: umaClasse).
entidadesValidas add: umaClasse.
relacionamentos
```

```
    do: [:rel |
      orMaisSuperior ← grafo classeMaisSuperiorDe: rel classeOrigem.
      dsMaisSuperior ← grafo classeMaisSuperiorDe: rel classeDestino.
      (rel nomeAtributoOrigem ≠ ' '
          and: [entidadesValidas includes: rel classeOrigem])
        ifTrue: [nomeAtributo ← rel nomeAtributoOrigem.
          entObjetivo ← rel classeDestino.
          codigo crtab; nextPutAll: ' umDescritor addMapping: ((OneToOneMapping new) attributeName: #' , nomeAtributo.
          codigo nextPutAll: '; referenceClass: ' , entObjetivo nome , '; mappingCriteria: (Join from:'.
          codigo nextPutAll: ' ((self tableNamed: ' '" , orMaisSuperior nome asUppercase , ' '") fieldNamed: ' '" , nomeAtributo ,
' '") to: ((self tableNamed: ' '" , dsMaisSuperior nome , ' '") fieldNamed: ' 'intid' '))).'].
      (rel nomeAtributoDestino ≠ ' '
          and: [entidadesValidas includes: rel classeDestino])
        ifTrue: [nomeAtributo ← rel nomeAtributoDestino.
          entObjetivo ← rel classeOrigem.
          codigo crtab; nextPutAll: ' umDescritor addMapping: ((OneToOneMapping new) attributeName: #' , nomeAtributo.
          codigo nextPutAll: '; referenceClass: ' , entObjetivo nome , '; mappingCriteria: (Join from:'.
          codigo nextPutAll: ' ((self tableNamed: ' '" , dsMaisSuperior nome asUppercase , ' '") fieldNamed: ' '" , nomeAtributo ,
' '") to: ((self tableNamed: ' '" , orMaisSuperior nome , ' '") fieldNamed: ' 'intid' '))).'] ].
  ↑ codigo contents!
```

gerarRelacionamentos1aNDe: *umaClasse* continuando: *umCodigo*

```
  "Gera a definicao da persistencia do GLORP relacionada aos
  relacionamentos 1−N da classe"
  | codigo relacionamentos nomeAtributo entidadesValidas orMaisSuperior dsMaisSuperior |
  codigo ← TextStream with: umCodigo.
  relacionamentos ← (grafo relacionamentosDe: umaClasse)
        select: [:rel | rel grauDestino ≠ 1
            and: [rel grauOrigem = 1] ].
  (grafo classesSuperioresDe: umaClasse)
    do: [:entidade | relacionamentos
        addAll: ((grafo relacionamentosDe: entidade)
            select: [:rel | rel grauDestino ≠ 1
                and: [rel grauOrigem = 1] ])].
  entidadesValidas ← OrderedCollection
        withAll: (grafo classesSuperioresDe: umaClasse).
  entidadesValidas add: umaClasse.
  relacionamentos
    do: [:rel |
      orMaisSuperior ← grafo classeMaisSuperiorDe: rel classeOrigem.
      dsMaisSuperior ← grafo classeMaisSuperiorDe: rel classeDestino.
      (entidadesValidas includes: rel classeOrigem)
        ifTrue: [nomeAtributo ← rel nomeAtributoOrigem.
          codigo crtab; nextPutAll: ' umDescritor addMapping: ((OneToOneMapping new) attributeName: #' , nomeAtributo.
          codigo nextPutAll: '; referenceClass: ' , rel classeDestino nome , '; mappingCriteria: (Join '.
          codigo nextPutAll: ' from: ((self tableNamed: ' '" , orMaisSuperior nome asUppercase , ' '") fieldNamed: ' '" , nomeAtri-
buto , ' '") to: ((self tableNamed: ' '" , dsMaisSuperior nome asUppercase , ' '") fieldNamed: ' 'intid' '))).'].
      ((entidadesValidas includes: rel classeDestino)
          and: [rel nomeAtributoDestino ≠ ' '])
        ifTrue: [nomeAtributo ← rel nomeAtributoDestino.
          codigo crtab; nextPutAll: ' umDescritor addMapping: ((OneToManyMapping new) attributeName: #' , nomeAtributo ,
'; referenceClass: '.
          codigo nextPutAll: rel classeOrigem nome , '; mappingCriteria: (Join from: ((self tableNamed: ' '" , dsMaisSuperior
nome asUppercase , ' '") fieldNamed: ' 'intid' ') to: '.
          codigo nextPutAll: '((self tableNamed: ' '" , orMaisSuperior nome asUppercase , ' '") fieldNamed: ' '" , rel nomeAtributo-
Origem , ' '"))).'] ].
  ↑ codigo contents!
```

gerarRelacionamentosNaNDe: *umaClasse* continuando: *umCodigo*

```
  "Gera a definicao da persistencia do GLORP relacionada aos
  relacionamentos N−Nda classe"
  | codigo relacionamentos nomeAtributo entidadesValidas orMaisSuperior dsMaisSuperior |
  codigo ← TextStream with: umCodigo.
  relacionamentos ← (grafo relacionamentosDe: umaClasse)
        select: [:rel | (rel grauOrigem = 1) not
            and: [(rel grauDestino = 1) not] ].
  (grafo classesSuperioresDe: umaClasse)
```

```
    do: [:entidade | relacionamentos
        addAll: ((grafo relacionamentosDe: entidade)
            select: [:rel | (rel grauOrigem = 1) not
                and: [(rel grauDestino = 1) not] ])].
  entidadesValidas ← OrderedCollection
        withAll: (grafo classesSuperioresDe: umaClasse).
  entidadesValidas add: umaClasse.
  relacionamentos
    do: [:rel |
      orMaisSuperior ← grafo classeMaisSuperiorDe: rel classeOrigem.
      dsMaisSuperior ← grafo classeMaisSuperiorDe: rel classeDestino.
      (entidadesValidas includes: rel classeDestino)
        ifTrue: [nomeAtributo ← rel nomeAtributoDestino.
          codigo crtab; nextPutAll: ' umDescritor addMapping: ((ManyToManyMapping new) attributeName: #' , nomeAtributo ,
'; referenceClass: '.
          codigo nextPutAll: rel classeOrigem nome , '; mappingCriteria: (Join from: ((self tableNamed: ' '' , dsMaisSuperior
nome asUppercase , ' '') fieldNamed: ' 'intid' ') '.
          codigo nextPutAll: ' to: ((self tableNamed: ' '' , rel nomeAtributoOrigem asUppercase , '←' , rel nomeAtributoDestino
asUppercase , ' '') fieldNamed: ' '' , rel classeDestino nome , ' ''))).']
        ifFalse: [nomeAtributo ← rel nomeAtributoOrigem.
          codigo crtab; nextPutAll: ' umDescritor addMapping: ((ManyToManyMapping new) attributeName: #' , nomeAtributo ,
'; referenceClass: '.
          codigo nextPutAll: rel classeDestino nome , '; mappingCriteria: (Join from: ((self tableNamed: ' '' , orMaisSuperior
nome asUppercase , ' '') fieldNamed: ' 'intid' ') '.
          codigo nextPutAll: ' to: ((self tableNamed: ' '' , rel nomeAtributoOrigem asUppercase , '←' , rel nomeAtributoDestino
asUppercase , ' '') fieldNamed: ' '' , rel classeOrigem nome , ' ''))).'] ].
  ↑ codigo contents!
```

gerarTypeResolverPara: *umaClasse*

```
  "Gera os typeResolver, que sao metodos do descritor de persistencia do glorp que definem como filtrar os elementos de uma dada
classe na tabela onde estao sendo persistidos"
  | codigo |
  codigo ← TextStream with: 'typeResolverFor' , umaClasse nome.
  codigo cr; crtab; nextPutAll: '↑ FilteredTypeResolver forRootClass: ' , (grafo classeMaisSuperiorDe: umaClasse) nome , '.'.
  ↑ codigo contents!

    attributeName: #' , nomeAtributo , '; referenceClass: '.
          codigo nextPutAll: rel classeDestino nome , '; mappingCriteria: (Join from: ((self tableNamed: ' '' , orMaisSuperior
nome asUppercase , ' '') fieldNamed: ' 'intid' ') '.
          codigo nextPutAll: ' to: ((self tableNamed: ' '' , rel nomeAtributoOrigem asUppercase , '←' , rel nomeAtributoDestino
asUppercase , ' '') fieldNamed: ' '' , rel classeOrigem nome , ' ''))).'] ].
  ↑ codigo contents!
```

gerarTypeResolverPara: *umaClasse*

```
  "Gera os typeResolver, que sao metodos do descritor de persistencia do glorp que definem como filtrar os elementos de uma dada
classe na tabela onde estao sendo persistidos"
  | codigo |
  codigo ← TextStream with: 'typeResolverFor' , umaClasse nome.
  codigo cr; crtab; nextPutAll: '↑ FilteredTypeResolver forRootClass: ' , (grafo classeMaisSuperiorDe: umaClasse) nome , '.'.
  ↑ codigo contents!

    rMaisSuperior nome asUppercase , ' '') fieldNamed: ' 'intid' ') '.
          codigo nextPutAll: ' to: ((self tableNamed: ' '' , rel nomeAtributoOrigem asUppercase , '←' , rel nomeAtributoDestino
asUppercase , ' '') fieldNamed: ' '' , rel classeOrigem nome , ' ''))).'] ].
  ↑ codigo contents!
```

gerarTypeResolverPara: *umaClasse*

```
  "Gera os typeResolver, que sao metodos do descritor de persistencia do glorp que definem como filtrar os elementos de uma dada
classe na tabela onde estao sendo persistidos"
  | codigo |
  codigo ← TextStream with: 'typeResolverFor' , umaClasse nome.
  codigo cr; crtab; nextPutAll: '↑ FilteredTypeResolver forRootClass: ' , (grafo classeMaisSuperiorDe: umaClasse) nome , '.'.
  ↑ codigo contents!

    ue definem como filtrar os elementos de uma dada classe na tabela onde estao sendo persistidos"
```

```
| codigo |
codigo ← TextStream with: 'typeResolverFor' , umaClasse nome.
codigo cr; crtab; nextPutAll: '↑ FilteredTypeResolver forRootClass: ' , (grafo classeMaisSuperiorDe: umaClasse) nome , '.'.
↑ codigo contents!
```

Object subclass: OORDBMProjeto

Category: OORDBM-Projeto

Instance variable names: prefixo categoria hostDB nomeDB usuario senha conexao grafo gerarTabelasBD gerarAsClasses gerarSuportePersistencia gerarNavegadores

Esta entidade e responsavel por, dada uma modelagem de um sistema armazenada em um XMIDocumento, gerar o mesmo conforme os parametros definidos.

Class protocol

Methods for category: as yet unclassified

new
↑ super new initialize.

Instance protocol

Methods for category: accessing

categoria
"Answer the value of categoria"
↑ categoria

categoria: *anObject*
"Set the value of categoria"
categoria ← anObject

conexao
"Answer the value of conexao"
conexao isNil ifTrue: [self estabelecerConexao.].
↑ conexao

conexao: *anObject*
"Set the value of conexao"
conexao ← anObject

gerarAsClasses
"Answer the value of gerarAsClasses"
↑ gerarAsClasses

gerarAsClasses: *anObject*
"Set the value of gerarAsClasses"
gerarAsClasses ← anObject

gerarNavegadoresPara: *umGrafo*
| grafoCompleto geradorInterface classePersistencia |
grafoCompleto ← umGrafo expandaAtributosParaFilhos.
geradorInterface ← OORDBMGeradorDeInterfaceDeEdicaoPrefab new.
classePersistencia ← self classePersistencia new hierarquia: grafo.
classePersistencia prefixo: (self prefixo).
geradorInterface nomeGerenciadorDePersistencia: (classePersistencia nomeGerenciador).
geradorInterface nomeCategoria: (self categoria).
grafoCompleto classes
do: [:classe | ((OORDBMTiposBasicos ehTipoBasico: (classe nome)) not) ifTrue: [geradorInterface nomeClasseGerar: ((self prefixo) , 'NavegadorDe' , classe nome). geradorInterface classe: classe. geradorInterface gerar.]].!

gerarSuportePersistencia
"Answer the value of gerarSuportePersistencia"
↑ gerarSuportePersistencia

gerarSuportePersistencia: *anObject*
"Set the value of gerarSuportePersistencia"
gerarSuportePersistencia ← anObject

gerarTabelasBD
"Answer the value of gerarTabelasBD"
↑ gerarTabelasBD

gerarTabelasBD: *anObject*
"Set the value of gerarTabelasBD"
gerarTabelasBD ← anObject

hierarquia: *umaHierarquia*
grafo ← umaHierarquia !

hostDB
"Answer the value of hostDB"
↑ hostDB

hostDB: *anObject*
"Set the value of hostDB"
hostDB ← anObject

initialize
self gerarTabelasBD: true.
self gerarAsClasses: true.

```
  self gerarSuportePersistencia: true.
nomeDB
  “Answer the value of nomeDB"
  ↑ nomeDB
nomeDB: anObject
  “Set the value of nomeDB"
  nomeDB ← anObject
prefixo
  “Answer the value of prefixo"
  ↑ prefixo
prefixo: anObject
  “Set the value of prefixo"
  prefixo ← anObject
senha
  “Answer the value of senha"
  ↑ senha
senha: anObject
  “Set the value of senha"
  senha ← anObject
usuario
  “Answer the value of usuario"
  ↑ usuario
usuario: anObject
  “Set the value of usuario"
  usuario ← anObject
xmi: anObject
  grafo ← (OORDBMHierarquiaDeClassesComParserXMI new xmi: anObject).
```

Methods for category: gerar

```
gerarProjeto
  | persistencia tabelas progresso visoes |
  grafo isNil
    ifTrue: [self error: ’Uma hieraraquia de classes nao foi definida!’].
  self incluirPrefixoNasClasses.
  progresso ← ProgressMorphComFundo
        label: ’Gerando projeto’
        corFundo: Color yellow
        corBorda: Color black.
  progresso subLabel: ’Estabelecendo parametros’.
  progresso openInWorld.
  grafo categoria: categoria.
  progresso incrDone: 0.1.
  progresso subLabel: ’Gerando hierarquia de classes’.
  persistencia ← self classeGeradoraTabelas new hierarquia: grafo.
  progresso incrDone: 0.1.
  progresso subLabel: ’Gerando estrutura de tabelas’.
  tabelas ← persistencia tabelas.
  visoes ← persistencia visoes.
  progresso incrDone: 0.2.
  self gerarTabelasBD
    ifTrue: [progresso subLabel: ’Gerando entidades do banco de dados’.
      self gerarBanco: tabelas.
      self gerarVisoes: visoes].
  progresso incrDone: 0.2.
  self gerarAsClasses
    ifTrue: [progresso subLabel: ’Gerando classes’.
      self gerarClasses: grafo].
  progresso incrDone: 0.2.
  self gerarSuportePersistencia
    ifTrue: [progresso subLabel: ’Gerando suporte a persistencia’.
      self gerarPersistenciaEm: tabelas].
  progresso incrDone: 0.1.
  self gerarNavegadores
    ifTrue: [progresso subLabel: ’Gerando navegadores’.
      self gerarNavegadoresPara: grafo].
  progresso incrDone: 0.2.
```

```
  progresso subLabel: 'Concluido!'.
  self inform: 'Concluido!'.
  progresso delete.
  self removerPrefixoDasClasses!
```

Methods for category: private

classeGeradoraClasses
```
  ↑OORDBMGeradorClasses
```
classeGeradoraTabelas
```
  ↑ self subClassResponsibility
```
classePersistencia
```
  ↑ self subClassResponsibility
```
estabelecerConexao
```
  | argumentos |
  conexao ← PGConnection new.
  argumentos ← PGConnectionArgs hostname: hostDB portno: 5432 databaseName: nomeDB userName: usuario password: senha.
  conexao connectionArgs: argumentos.
  conexao startup.
```
gerarBanco: *tabelas*
```
  | sqls |
  sqls ← OrderedCollection new.
  tabelas values
    do: [:tabela | sqls
        addAll: (OORDBMSQLPostgres retorneSQLsDeFormaIncrementalParaTabela: tabela)].
  tabelas values
    do: [:tabela | sqls
        addAll: (OORDBMSQLPostgres retorneSQLsRelacionamentos: tabela)].
  sqls do: [ :sql | (self conexao) execute: sql. ].
"   sqls explore"!
```
gerarClasses: *grafo*
```
  | gerador |
  gerador ← (self classeGeradoraClasses) new.
  gerador hierarquia: grafo.
  gerador gerar.
```
gerarInicializacaoPersistencia: *nomeGerenciador*
```
  | classe codigo |
  classe ← Smalltalk at: (nomeGerenciador asSymbol).
  "Metodo de inicio de sessao"
  codigo ← TextStream with: 'initialize '.
  codigo nextPutAll: 'usuarioBD ← ' " , usuario , ' ".'; crtab.
  codigo nextPutAll: 'senhaBD ← ' " , senha , ' ".'; crtab.
  codigo nextPutAll: 'enderecoBD ← ' " , hostDB , ' ".'; crtab.
  codigo nextPutAll: 'nomeBD ← ' " , nomeDB , ' ".'; crtab.
  classe compile: (codigo contents) classified: 'initialization'.
  codigo ← (TextStream with: 'new ') cr; crtab; nextPutAll: ' ↑ super new initialize.'; yourself.
  classe theMetaClass compile: (codigo contents) classified: 'initialization'.
```
gerarNavegadores
```
  ↑ gerarNavegadores
```
gerarNavegadores: *umBooleano*
```
  gerarNavegadores ← umBooleano!
```
gerarPersistenciaEm: *tabelas*
```
  |desc|
  desc ← (self classePersistencia) new.
  desc tabelas: tabelas.
  desc hierarquia: grafo.
  desc prefixo: prefixo.
  desc gerar.
  desc gerarGerenciadorPersistencia.
  self gerarInicializacaoPersistencia: (desc nomeGerenciador).
```
gerarVisoes: *visoes*
```
  | sqls |
```

```
sqls ← OrderedCollection new.
visoes values do: [:visao | sqls
        addAll: (OORDBMSQLPostgres retorneSQLsVisao: visao)].
sqls do: [ :sql | (self conexao) execute: sql. ].
“sqls explore"!
```

incluirPrefixoNasClasses

```
grafo classes do: [:classe |
   (OORDBMTiposBasicos ehTipoBasico: (classe nome)) ifFalse: [
      classe nome: (prefixo, classe nome).
] ].
```

removerPrefixoDasClasses

```
grafo classes
   do: [:classe | (OORDBMTiposBasicos ehTipoBasico: classe nome)
        ifFalse: [classe nome: (classe nome copyFrom: (prefixo size + 1) to: (classe nome size))] ]!
```

OORDBMProjeto subclass: OORDBMProjetoClassesConcretas
Category: OORDBM-Projeto

Esta entidade e responsavel por, dada uma modelagem de um sistema armazenada em um XMIDocumento, gerar o mesmo conforme os parametros definidos.

Instance protocol

Methods for category: private

classeGeradoraTabelas

↑ OORDBMGeradorEsquemaRelacionalClassesConcretas

classePersistencia

↑ OORDBMGeradorPersistenciaGLORPClassesConcretas

OORDBMProjeto subclass: OORDBMProjetoHierarquiaCompleta
Category: OORDBM-Projeto

Esta entidade e responsavel por, dada uma modelagem de um sistema armazenada em um XMIDocumento, gerar o mesmo conforme os parametros definidos.

Instance protocol

Methods for category: private

classeGeradoraTabelas

↑ OORDBMGeradorEsquemaRelacionalHierarquiaCompleta.

classePersistencia

↑ OORDBMGeradorPersistenciaGLORPHierarquiaCompleta

OORDBMProjeto subclass: OORDBMProjetoTabelaUnica
Category: OORDBM-Projeto

Esta entidade e responsavel por, dada uma modelagem de um sistema armazenada em um XMIDocumento, gerar o mesmo conforme os parametros definidos.

Instance protocol

Methods for category: as yet unclassified

classeGeradoraTabelas
↑ OORDBMGeradorEsquemaRelacionalTabelaUnica!

classePersistencia
↑ OORDBMGeradorPersistenciaGLORPTabelaUnica
ompleta.

classePersistencia
↑ OORDBMGeradorPersistenciaGLORPHierarquiaCompleta

OORDBMProjeto subclass: OORDBMProjetoTabelaUnica
Category: OORDBM-Projeto

Esta entidade e responsavel por, dada uma modelagem de um sistema armazenada em um XMIDocumento, gerar o mesmo conforme os parametros definidos.

Instance protocol

Methods for category: as yet unclassified

classeGeradoraTabelas
↑ OORDBMGeradorEsquemaRelacionalTabelaUnica!

classePersistencia
↑ OORDBMGeradorPersistenciaGLORPTabelaUnica

NCDrawingProject subclass: OORDBMDiagramaProjeto
Category: OORDBM-Widgets
Instance variable names: telaGeracaoProjeto

Componente principal do sistema de modelagem de classes. Instancia um novo projeto no Squeak, ja configurado para modelar e posteriormente exportar/gerar classes, tabelas e sistemas de persistncia para a modelagem feita.

Class protocol

Methods for category: as yet unclassified

defaultFileSuffix

```
↑ 'dg'.
```

defaultName

```
↑'Novo diagrama'.
```

fileSuffixes

```
"Answer the file suffixes used to store my instances"
↑#( 'dg' )
```

itemNoMenuWorld

```
TheWorldMenu registerOpenCommand: { 'Modelador de sistemas' . { OORDBMDiagramaProjeto . #newProject }. 'Um modelador de sistemas' }
```

newMorphicOn: *aPasteUpOrNil* named: *aName*

```
| newProject |
newProject ← self basicNew initMorphic.
aName isNil ifFalse: [ newProject renameTo: aName.].
self addingProject: newProject.
aPasteUpOrNil ifNotNil: [newProject installPasteUpAsWorld: aPasteUpOrNil].
newProject createViewIfAppropriate.
↑newProject
```

newProjectNamed: *aName*

```
| ret shifter flap|
ret ← self newMorphicOn: nil named: aName.
ret changeSet name: aName.
ret world addMorph: ret buildBasicToolbar.
ret    flapsSuppressed: true.
ret world addMorph: (flap ← (Flaps newOORDBMDiagramaFlap)).
shifter ← OORDBMScroller new.
shifter classesToIgnore add: (ret class toolbarClass); add: (OORDBMScroller).
shifter type: (OORDBMScroller verticalType).
ret world addMorph: (shifter).
shifter topLeft: (Point x: (ret world width − (shifter extent x)) y: 0).
shifter extent: (Point x: (shifter extent x) y: (ret world height − (shifter extent x))).
shifter ← OORDBMScroller new.
shifter classesToIgnore add: (ret class toolbarClass); add: (OORDBMScroller).
shifter type: (OORDBMScroller horizontalType).
ret world addMorph: (shifter).
shifter topLeft: (Point x: (flap bounds corner x) + 5 y: (ret world height − (shifter extent x))).
shifter extent: (Point x: (ret world width − ((shifter extent x) + (shifter topLeft x))) y: (shifter extent x)).
ret renameTo: aName.
↑ ret.
```

novoProjetoComNomeGerado

```
| numeroProjeto |
numeroProjeto ← 1.
[ (self named: 'Projeto' , (numeroProjeto asString)) isNil ] whileFalse: [ numeroProjeto ← numeroProjeto + 1].
↑ self newProjectNamed: 'Projeto' , (numeroProjeto asString).
```

promptForNewDrawing

```
↑'Nome do diagrama:'
```

Instance protocol

Methods for category: as yet unclassified

Methods for category: converting

exportarComoXMI

```
"Exporta os elements do projeto como um documento XMI"
↑ self world exportarComoXMI!
```

gerarProjeto

```
"Se ainda nao houver sido, instancia uma tela de geracao de projeto para
este projeto. Caso jah tenha sido criada, apenas exibe a mesma."
self validar ifTrue: [
```

```
telaGeracaoProjeto isNil
    ifTrue: [telaGeracaoProjeto ← OORDBMMorphTelaGeracaoProjeto new xmi: self exportarComoXMI;
                openInSystemWindow.
        telaGeracaoProjeto resistsRemoval: true]
    ifFalse: [telaGeracaoProjeto client xmi: self exportarComoXMI.
        telaGeracaoProjeto openInWorld: self world.
        telaGeracaoProjeto show] ]!
```

Methods for category: displaying

buildBasicToolbar

```
"Project current buildBasicToolbar openInHand"
"Personalizado para o projeto"
| newBar button |
newBar ← self class toolbarClass new width: 100;
        wrapDirection: #none;
        listDirection: #topToBottom;
        hResizing: #rigid;
        color: (Color
            r: 0.742
            g: 0.742
            b: 0.742).
"Novo"
button ← newBar class
        buttonTarget: self class
        action: #enterNewProject
        arguments: {}
        icon: 'icons/folder←new.gif'
        label: 'Novo '.
newBar addMorphBack: button.
"Trocar"
button ← newBar class
        buttonTarget: self class
        action: #jumpToProject
        arguments: {}
        icon: 'icons/contents.gif'
        label: 'Trocar'.
newBar addMorphBack: button.
"Abrir"
button ← newBar class
        buttonTarget: self
        action: #loadFromFile
        arguments: {}
        icon: 'icons/fileopen.gif'
        label: 'Abrir'.
newBar addMorphBack: button.
"Salvar"
button ← newBar class
        buttonTarget: self
        action: #saveToFile
        arguments: {}
        icon: 'icons/filesave.gif'
        label: 'Salvar'.
newBar addMorphBack: button.
"Salvar imagem"
button ← newBar class
        buttonTarget: self
        action: #exportAsGIF
        arguments: {}
        icon: 'icons/colorize.gif'
        label: 'Exportar imagem'.
newBar addMorphBack: button.
"Gerar"
button ← newBar class
        buttonTarget: self
        action: #gerarProjeto
        arguments: {}
```

```
            icon: 'icons/colorize.gif'
            label: 'Gerar projeto'.
    newBar addMorphBack: button.
    "Alinhar"
    button ← newBar class
            buttonTarget: NCBroomMorph
            action: #newTransientInHand
            arguments: {}
            icon: 'icons/broom.gif'
            label: 'Alinhar'.
    newBar addMorphBack: button.
    "Sair"
    button ← newBar class
            buttonTarget: self class
            action: #quitSession
            arguments: {}
            icon: 'icons/exit.gif'
            label: 'Fechar'.
    newBar addMorphBack: button.
    newBar
        bounds: (0 @ 0 extent: 90 @ (60 × newBar submorphs size)).
    newBar
        bounds: (newBar bounds outsetBy: 1).
    "force layout again"
    ↑ newBar!
```

Methods for category: file in/out

loadFromFile

```
    "Load a drawing from a file and put the morphs into the
    correctly named project, or into a new project"
    "NCDrawingProject loadFromFile"
    | stream morphs project parsed projectName bytes byteStream |
    stream ← FileList2 modalFileSelectorForSuffixes: self class fileSuffixes.
    stream ifNil: [ ↑nil ].
    parsed ← self class parseProjectFileName: stream localName.
    projectName ← parsed first.
    project ← self.
    (project notNil and: [ project isEmpty not ])
        ifTrue: [ | response |
            response ← (SelectionMenu
                labels: #('Sobrescrever' 'Fundir' 'Carregar como um novo projeto' 'Cancelar')
                selections: #('overwrite' 'merge' 'new' 'cancel'))
                startUpWithCaption: 'Ja existe um projeto aberto'.
            response = 'cancel' ifTrue: [ ↑nil ].
            response = 'new'
                ifTrue: [ project ← self class newProjectNamed: ((self class newProjectName) ifNil: [ ↑nil ]) ].
            response = 'overwrite' ifTrue: [ project clear ].
        ]
        ifFalse: [ project ← self class newProjectNamed: projectName ].
    stream binary.
    bytes ← (GZipReadStream on: stream) binary; contents.
    byteStream ← (RWBinaryOrTextStream with: bytes).
    byteStream reset; binary.
    morphs ← byteStream fileInObjectAndCode.
    project world addAllMorphs: (morphs select: [ :ea | ea isMorph ]).
    project extraData: (morphs reject: [ :ea | ea isMorph ]).
    (project ∼∼ self class current)
        ifTrue: [ project enter ].
    ↑stream localName.
```

Methods for category: initialization

initMorphic

```
    "Written so that Morphic can still be removed. Note that #initialize is never actually called for a morphic project −− see the senders
of this method."
    Smalltalk verifyMorphicAvailability ifFalse: [↑ nil].
    changeSet ← ChangeSet new.
```

```
transcript ← TranscriptStream new.
displayDepth ← Display depth.
parentProject ← CurrentProject.
isolatedHead ← false.
world ← OORDBMMorphDiagrama newWorldForProject: self.
self initializeProjectPreferences “Do this ×after× a world is installed so that the project will be recognized as a morphic one."
```

Methods for category: private

clear

```
world submorphs do: [ :ea | (self class nonSavedClasses anySatisfy: [ :cls | ea isKindOf: cls ]) ifFalse: [ ea delete ] ].
```

Methods for category: testing

validar

```
↑ self world validar.
```

NCConnectorMorph subclass: OORDBMMorphAssociacao

Category: OORDBM-Widgets
Instance variable names: labelOrigem labelDestino labelDefinicao ultimoErro

Componente que representa uma associacao entre classes.

Instance protocol

Methods for category: *connectors-queries

wantsToAttachEnd: *index* toMorph: *aMorph*

```
"Answer true if I want my end with index 'index' (1 or 2) to attach to
the given Morph."
"Impede que conecte−se a elementos que nao sao classes"
(aMorph isKindOf: OORDBMMorphClasse)
   ifFalse: [↑ false].
(index = 1
      and: [labelOrigem isNil])
   ifTrue: [labelOrigem ← self addLabel: '1' at: self line firstVertex.
      labelOrigem extent: 80 @ 5.
      labelOrigem wrapFlag: false.
      labelOrigem wrapFlag: true.
      labelOrigem centered].
(index = 2
      and: [labelDestino isNil])
   ifTrue: [labelDestino ← self addLabel: '1' at: self line lastVertex.
      labelDestino extent: 80 @ 5.
      labelDestino wrapFlag: false.
      labelDestino wrapFlag: true.
      labelDestino centered].
labelDefinicao isNil
   ifTrue: [labelDefinicao ← self addLabel: 'descricao' near: (self line midpoint)].
↑ true!
```

Methods for category: accessing

classeDestino

```
↑ self destinationMorph
```

classeOrigem

```
↑ self sourceMorph
```

defineDescricao

```
| novaDescricao |
novaDescricao ← (labelDefinicao contents asString isNil) ifTrue: [' '] ifFalse: [labelDefinicao contents asString].
novaDescricao ← FillInTheBlankMorph request: 'Qual a descricao da relacao?' initialAnswer: novaDescricao.
novaDescricao isEmpty ifFalse: [ self labelDefinicao contents: novaDescricao ].
```

defineGrandezaDestino

```
| novaGrandeza |
novaGrandeza ← (self obtemGrandezasDe: self labelDestino contents asString) isNil
      ifTrue: [' ']
      ifFalse: [self obtemGrandezasDe: self labelDestino contents asString].
novaGrandeza ← FillInTheBlankMorph request: 'Qual a grandeza do destino da relacao?' initialAnswer: novaGrandeza.
novaGrandeza isEmpty
   ifFalse: [self grandezaDestino: novaGrandeza]!
```

defineGrandezaOrigem

```
| novaGrandeza |
novaGrandeza ← (self obtemGrandezasDe: self labelOrigem contents asString) isNil
      ifTrue: [' ']
      ifFalse: [self obtemGrandezasDe: self labelOrigem contents asString].
novaGrandeza ← FillInTheBlankMorph request: 'Qual a grandeza da origem da relacao?' initialAnswer: novaGrandeza.
novaGrandeza isEmpty
   ifFalse: [self grandezaOrigem: novaGrandeza]!
```

definePapelDestino

```
| novoPapel |
novoPapel ← (self papelDestino isNil) ifTrue: [' '] ifFalse: [self papelDestino].
novoPapel ← FillInTheBlankMorph request: 'Qual o papel da entidade destino da relacao?' initialAnswer: novoPapel.
novoPapel isEmpty ifFalse: [ self papelDestino: novoPapel ].
```

definePapelOrigem

```
| novoPapel |
novoPapel ← (self papelOrigem isNil) ifTrue: [' '] ifFalse: [self papelOrigem].
novoPapel ← FillInTheBlankMorph request: 'Qual o papel da entidade origem da relacao?' initialAnswer: novoPapel.
novoPapel isEmpty ifFalse: [ self papelOrigem: novoPapel ].
```

grandezaDestino

```
| pontaAssoc infoAssoc |
pontaAssoc ← OORDBMMorphAssociacaoLimite new.
infoAssoc ← self obtemGrandezasDe: self labelDestino contents asString.
↑ self completaPontaAssociacao: pontaAssoc com: infoAssoc!
```

grandezaDestino: *umaGrandeza*

```
self setGrandeza: self labelDestino para: umaGrandeza!
```

grandezaOrigem

```
| pontaAssoc infoAssoc |
pontaAssoc ← OORDBMMorphAssociacaoLimite new.
infoAssoc ← self obtemGrandezasDe: self labelOrigem contents asString.
↑ self completaPontaAssociacao: pontaAssoc com: infoAssoc!
```

grandezaOrigem: *umaGrandeza*

```
self setGrandeza: self labelOrigem para: umaGrandeza!
```

papelDestino

```
| papel |
papel ← self papel: self labelDestino.
↑papel withBlanksTrimmed.“ isEmpty
    ifTrue: [self grandezaDestino limiteSuperior ≠ 1
        ifTrue: [↑ self classeOrigem nomeDaClasse decapitalized , 's']
        ifFalse: [↑ self classeOrigem nomeDaClasse decapitalized] ]
    ifFalse: [↑ papel]"!
```

papelDestino: *umPapel*

```
self setPapel: self labelDestino para: umPapel    .
```

papelOrigem

```
| papel |
papel ← self papel: self labelOrigem.
↑papel withBlanksTrimmed “isEmpty
    ifTrue: [self grandezaOrigem limiteSuperior ≠ 1
        ifTrue: [↑ self classeDestino nomeDaClasse decapitalized , 's']
        ifFalse: [↑ self classeDestino nomeDaClasse decapitalized] ]
    ifFalse: [↑ papel]"!
```

papelOrigem: *umPapel*

```
self setPapel: self labelOrigem para: umPapel    .
```

ultimoErro

```
↑ ultimoErro
```

Methods for category: menus

addBasicMenuItemsTo: *aCustomMenu* **event:** *evt*

```
owner
    ifNotNil: [aCustomMenu
        add: 'Mover'
        target: evt hand
        selector: #attachMorph:
        argument: self;
        add: 'Excluir'
        target: self
        selector: #delete;
        add: 'Redimensionar'
        target: self
        selector: #resizeMorph:
        argument: evt;
add: 'Descricao da relacao'
        target: self
        selector: #defineDescricao;
        add: 'Papel da classe origem na relacao'
```

```
            target: self
            selector: #definePapelOrigem;
            add: 'Papel da classe destino na relacao'
            target: self
            selector: #definePapelDestino;
          add: 'Grandeza da origem na relacao'
            target: self
            selector: #defineGrandezaOrigem;
            add: 'Grandeza do destino na relacao'
            target: self
            selector: #defineGrandezaDestino]!
```

addYellowButtonMenuItemsTo: *aMenu* **event:** *evt*

```
self addBasicMenuItemsTo: aMenu event: evt.
↑ aMenu.
```

Methods for category: printing

asString

```
↑ '(' , self classeOrigem asString , ')(' , self papelOrigem , ')<−−−−−−−−−→(' , self papelDestino , ')(' , self classeDestino asString , ')'!
```

Methods for category: private

completaPontaAssociacao: *pontaAssoc* **com:** *infoAssoc*

```
"Recebe um OORDBMMorphAssociacaoLimite(pontaAssoc) e completa o
mesmo com as informacoes presentes no componente − papeis(roles) de
cada ponta da associacao e as cardinalidades dos papeis"
| tmpInfoAssoc |
infoAssoc isEmpty
  ifTrue: [pontaAssoc limiteInferior: 0.
    pontaAssoc limiteSuperior: −1]
  ifFalse: [infoAssoc withBlanksTrimmed isAllDigits
      ifTrue: [(pontaAssoc limiteSuperior: infoAssoc asInteger)
          limiteInferior: infoAssoc asInteger]
      ifFalse: [infoAssoc withBlanksTrimmed = '×'
          ifTrue: [pontaAssoc limiteSuperior: −1;
              limiteInferior: 0]
          ifFalse: [tmpInfoAssoc ← infoAssoc findTokens: ':'.
            tmpInfoAssoc at: 1 put: (tmpInfoAssoc at: 1) withBlanksTrimmed.
            tmpInfoAssoc at: 2 put: (tmpInfoAssoc at: 2) withBlanksTrimmed.
            (tmpInfoAssoc at: 1) isAllDigits
              ifTrue: [pontaAssoc limiteInferior: (tmpInfoAssoc at: 1) asInteger]
              ifFalse: [pontaAssoc limiteInferior: −1].
            (tmpInfoAssoc at: 2) isAllDigits
              ifTrue: [pontaAssoc limiteSuperior: (tmpInfoAssoc at: 2) asInteger]
              ifFalse: [pontaAssoc limiteSuperior: −1] ]] ].
↑ pontaAssoc!
```

labelDefinicao

```
↑ labelDefinicao
```

labelDefinicao: *umLabel*

```
labelDefinicao ← umLabel.
```

labelDestino

```
labelDestino isNil ifTrue: [ ↑ self error: 'Voc ainda nao conectou o destino da relacao a uma classe!' ].
↑ labelDestino.
```

labelOrigem

```
labelOrigem isNil ifTrue: [ ↑ self error: 'Voc ainda nao conectou a origem da relacao a uma classe!' ].
↑ labelOrigem.
```

obtemGrandezasDe: *aString*

```
| nString |
nString ← aString withBlanksTrimmed.
nString isEmpty ifTrue: [ ↑ ' ' ] ifFalse: [ ↑ ((nString findTokens: (Character cr)) at: 1) asString. ].
```

papel: *label*

```
| papel |
papel ← label contents asString withBlanksTrimmed.
papel isEmpty ifTrue: [ ↑ ' ' ].
```

```
  papel ← papel findTokens: (Character cr).
  papel size < 2 ifTrue: [ ↑ ' ' ] ifFalse: [ ↑ papel at: 2 ].
  ↑ ' '.
setGrandeza: label para: umaGrandeza
  | grandeza |
  grandeza ← label contents asString withBlanksTrimmed.
  grandeza isEmpty ifTrue: [ label contents: umaGrandeza ] ifFalse: [
    grandeza ← grandeza findTokens: (Character cr).
    grandeza size = 2 ifTrue: [ label contents: umaGrandeza , (Character cr asString) , (grandeza at: 2).
  ] ifFalse: [ label contents: umaGrandeza ] ].
setPapel: label para: umPapel
  | papel |
  papel ← label contents asString withBlanksTrimmed.
  papel isEmpty ifTrue: [ label contents: (Character cr asString) , umPapel ] ifFalse: [
    papel ← papel findTokens: (Character cr).
    label contents: (papel at: 1) , (Character cr asString) , umPapel.
  ].
validarGrandeza: umaGrandeza
  | tokens |
  umaGrandeza isAllDigits
    ifTrue: [↑ true].
  umaGrandeza = '×'
    ifTrue: [↑ true].
  tokens ← umaGrandeza findTokens: ':'.
  tokens size ≠ 2
    ifTrue: [ultimoErro ← 'Cardinalidade/grandeza informada invalida!'. ↑ false].
  tokens
    do: [:token | (token asString withBlanksTrimmed isAllDigits
          or: [token asString withBlanksTrimmed = '×'])
        ifFalse: [ultimoErro ← 'Cardinalidade/grandeza informada invalida!'.↑ false] ].
  ↑ true!

validarGrandezas
  | lbOrigem lbDestino |
  lbOrigem ← self obtemGrandezasDe: self labelOrigem contents asString.
  lbDestino ← self obtemGrandezasDe: self labelDestino contents asString.
  (self validarGrandeza: lbOrigem) ifFalse: [ ↑ false. ].
  (self validarGrandeza: lbDestino) ifFalse: [ ↑ false. ].
  ↑ true.
validarPapeis
  (self grandezaOrigem limiteSuperior = 1
      and: [self grandezaDestino limiteSuperior = 1])
    ifTrue: [(self papelOrigem = ' '
          and: [self papelDestino = ' '])
        ifTrue: [ultimoErro ← 'A relacao ' , self asString , ' tem que ter pelo menos um papel definido'.
          ↑ false] ].
  (self grandezaOrigem limiteSuperior ≠ 1
      and: [self grandezaDestino limiteSuperior ≠ 1])
    ifTrue: [(self papelOrigem = ' '
          or: [self papelDestino = ' '])
        ifTrue: [ultimoErro ← 'A relacao ' , self asString , ' tem que ter os dois papeis definidos'.
          ↑ false] ].
  (self grandezaOrigem limiteSuperior = 1
      and: [self grandezaDestino limiteSuperior ≠ 1])
    ifTrue: [self papelOrigem = ' '
        ifTrue: [ultimoErro ← 'A relacao ' , self asString , ' tem que ter o papel do lado unitario definido!!'.
          ↑ false] ].
  (self grandezaOrigem limiteSuperior ≠ 1
      and: [self grandezaDestino limiteSuperior = 1])
    ifTrue: [self papelDestino = ' '
        ifTrue: [ultimoErro ← 'A relacao ' , self asString , ' tem que ter o papel do lado unitario definido!'.
          ↑ false] ].
  (self papelOrigem ≠ ' ') ifTrue: [ (ValidadorDeClasses validarAtributo: (self papelOrigem)) ifFalse: [ ultimoErro ← 'Um dos
papeis esta incorreto segundo a sintaxe da linguagem para nomes de atributos'. ↑ false ] ].
  (self papelDestino ≠ ' ') ifTrue: [ (ValidadorDeClasses validarAtributo: (self papelDestino)) ifFalse: [ ultimoErro ← 'Um dos
papeis esta incorreto segundo a sintaxe da linguagem para nomes de atributos'. ↑ false ] ].
  ↑ true!
```

Methods for category: testing

validar

(self classeOrigem isNil not
and: [self classeDestino isNil not])
ifTrue: [↑(self validarGrandezas and: [self validarPapeis])] ifFalse: [ultimoErro ← 'Um dos terminais da associacao nao esta conectado!'].
↑ false!

Object subclass: OORDBMMorphAssociacaoLimite

Category: OORDBM-Widgets
Instance variable names: limiteSuperior limiteInferior

Representa a cardinalidade de uma ponta de uma associacao.

Instance protocol

Methods for category: accessing

limiteInferior

"Answer the value of limiteInferior"
↑ limiteInferior

limiteInferior: *anObject*

"Set the value of limiteInferior"
((limiteSuperior isNil not) and: [limiteSuperior $\neq -1$]) ifTrue: [limiteSuperior $<$ anObject ifTrue: [self error: 'Limite inferior nao deve ser maior que o superior']].
limiteInferior ← anObject!

limiteSuperior

"Answer the value of limiteSuperior"
↑ limiteSuperior

limiteSuperior: *anObject*

"Set the value of limiteSuperior"
((limiteInferior isNil not) and: [anObject $\neq -1$]) ifTrue: [limiteInferior $>$ anObject
ifTrue: [self error: 'Limite inferior nao deve ser maior que o superior']].
limiteSuperior ← anObject!

Methods for category: printing

asString

↑ limiteInferior asString , ':' , limiteSuperior asString.

NCTextRectangleMorph subclass: OORDBMMorphClasse

Category: OORDBM-Widgets
Instance variable names: morphNomeClasse morphAtributosClasse morphMetodosClasse ultimoErro

Representa uma classe, com nome, metodos e atributos

Class protocol

Methods for category: as yet unclassified

newUMLClassSymbolWithThreeBlocks

```
"Retorna uma instancia pronta para o uso"
"Armazena internamente para que serve cada um dos blocos de textos componentes"
↑ self newUMLClassSymbol.
```

Instance protocol

Methods for category: accessing

listaDeAtributos

```
|atributos resultado attPartes|
atributos ← (morphAtributosClasse contents asString withBlanksTrimmed) findTokens: (Character cr).
resultado ← OrderedCollection new.
atributos do: [:attr |
   attPartes ← (attr withBlanksTrimmed) findTokens: ':'.
   attPartes size < 2 ifTrue: [ attPartes addLast: 'Texto' ].
   resultado add: ((OORDBMMorphClasseAtributo new nome: ((attPartes at: 1) withBlanksTrimmed)) nomeDoTipo: ((attPartes
at: 2) withBlanksTrimmed)).
].
↑ resultado.
```

listaDeMetodos

```
|atributos resultado |
atributos ← (morphMetodosClasse contents asString withBlanksTrimmed) findTokens: (Character cr).
resultado ← OrderedCollection new.
resultado addAll: atributos.
↑ resultado.
```

nomeDaClasse

```
↑ morphNomeClasse contents asString.
```

Methods for category: initialization

initialize

```
| id fonts |
super initialize.
morphNomeClasse ← self textMorphAt: 1.
id ← self addTextBlock: ' '.
fonts ← TextStyle named: #ComicBold.
fonts ifNotNil: [
   (self textMorphAt: id)
      beAllFont: (fonts fontOfSize: 12)
].
(self textMorphAt: id) centered.
(self textMorphAt: id) contentsWrapped: 'Atributos'.
(self textMorphAt: id) readOnly: true.
id ← self addTextBlock: ' '.
morphAtributosClasse ← self textMorphAt: id.
id ← self addTextBlock: ' '.
fonts ifNotNil: [
   (self textMorphAt: id)
      beAllFont: (fonts fontOfSize: 12)
].
(self textMorphAt: id) contentsWrapped: 'Metodos'.
(self textMorphAt: id) centered.
(self textMorphAt: id) readOnly: true.
id ← self addTextBlock: ' '.
morphMetodosClasse ← self textMorphAt: id.
```

Methods for category: menus

addBasicMenuItemsTo: *aCustomMenu* event: *evt*

```
owner ifNotNil: [
   aCustomMenu
```

```
        add: 'Mover' target: evt hand selector: #attachMorph: argument: self;
        add: 'Excluir' target: self selector: #delete;
        add: 'Redimensionar' target: self selector: #resizeMorph: argument: evt;
        addLine
    ].
```

addYellowButtonMenuItemsTo: *aCustomMenu* **event:** *evt*

```
    self addBasicMenuItemsTo: aCustomMenu event: evt.
    self adicionarMenuDeClasseA: aCustomMenu event: evt.
"   self addTextMenuItemsTo: aCustomMenu event: evt."
```

adicionarMenuDeClasseA: *aCustomMenu* **event:** *evt*

```
    owner ifNotNil: [
      aCustomMenu
        addLine    ;
        add: 'Incluir atributo' target: self selector: #incluirAtributo;
        add: 'Excluir atributo' target: self selector: #excluirAtributo;
        addLine;
        add: 'Incluir metodo' target: self selector: #incluirMetodo;
        add: 'Excluir metodo' target: self selector: #excluirMetodo.
    ].
```

excluirAtributo

```
    |atributos atributo novosAtributos|
    atributo ← FillInTheBlank request: 'Qual o atributo a excluir?' initialAnswer: ' '.
    self inform: 'Este atributo sera excluido ' , atributo.
    atributos ← (morphAtributosClasse contents asString withBlanksTrimmed) findTokens: (Character cr).
    novosAtributos ← String new.
    atributos do: [:att |
      att = atributo ifFalse: [ novosAtributos ← novosAtributos , (Character cr asString) , att asString. ].
    ].
    morphAtributosClasse contentsWrapped: (novosAtributos copyWithoutFirst asString).
```

excluirMetodo

```
    |metodos metodo novoMetodos|
    metodo ← FillInTheBlank request: 'Qual o metodo a excluir?' initialAnswer: ' '.
    self inform: 'Este metodo sera excluido ' , metodo.
    metodos ← (morphMetodosClasse contents asString withBlanksTrimmed) findTokens: (Character cr).
    novoMetodos ← String new.
    metodos do: [:met |
      met = metodo ifFalse: [ novoMetodos ← novoMetodos , (Character cr asString) , met asString. ].
    ].
    morphMetodosClasse contentsWrapped: (novoMetodos copyWithoutFirst asString).
```

incluirAtributo

```
    | tela |
    tela ← OORDBMMorphTelaCriarAtributo
            new: 'Criar atributo para classe: ' , self nomeDaClasse
            blocoDeCodigoParaCriarAtributo: [:nome :tipo | | quebra | nome withBlanksTrimmed isEmpty
                  ifFalse: [tipo withBlanksTrimmed isEmpty ifFalse: [morphAtributosClasse contents asString withBlanksTrimmed size
> 0
                        ifTrue: [quebra ← Character cr asString]
                        ifFalse: [quebra ← ' '].
                    morphAtributosClasse contentsWrapped: morphAtributosClasse contents asString , quebra , nome , ':' , tipo] ]].
    tela tiposDeAtributos addAll: OORDBMTiposBasicos tiposUsuario.
    tela openInSystemWindow!
```

incluirMetodo

```
    |metodo quebra|
    metodo ← FillInTheBlank request: 'Qual o metodo a incluir?' initialAnswer: ' '.
    self inform: 'Este metodo sera incluido: ' , metodo.
    (morphMetodosClasse contents asString withBlanksTrimmed size > 0) ifTrue: [quebra ← Character cr asString] ifFalse: [quebra
← ' '].
    morphMetodosClasse contentsWrapped: (morphMetodosClasse contents asString , quebra, metodo).
```

Methods for category: printing

asString

```
    ↑ self nomeDaClasse.!
```

Methods for category: private

ultimoErro

```
↑ ultimoErro
```

validarAtributos

```
| definicaoValida tipoValido |
definicaoValida ← ValidadorDeClasses
        validarAtributos: (self listaDeAtributos
              collect: [:atributo | atributo nome]).
definicaoValida
   ifFalse: [ultimoErro ← 'O formato de definicao dos atributos esta incorreto'.
      ↑ false].
tipoValido ← (self listaDeAtributos
        collect: [:atributo | atributo nomeDoTipo])
        allSatisfy: [:tipo | OORDBMTiposBasicos ehTipoBasico: tipo].
tipoValido
   ifFalse: [ultimoErro ← 'Um dos tipos dos atributos esta incorreto'.
      ↑ false].
↑ true!
```

validarMetodos

```
(ValidadorDeClasses validarAssinaturasDeMetodos: self listaDeMetodos) ifFalse: [ ultimoErro ← 'A definicao dos metodos esta incorreta segundo a sintaxe da linguagem'. ↑ false] ifTrue: [ ↑ true ].!
```

validarNome

```
(ValidadorDeClasses validarNome: self nomeDaClasse) ifFalse: [ ultimoErro ← 'O nome da classe esta incorreto segundo a sintaxe da linguagem'. ↑ false] ifTrue: [↑ true].!
```

Methods for category: testing

validar

```
↑ ((self validarNome) and: [self validarAtributos]) and: [self validarMetodos].
```

Object subclass: OORDBMMorphClasseAtributo

Category: OORDBM-Widgets

Instance variable names: nome nomeDoTipo ultimoErro

Representa um atributo de uma classe

Instance protocol

Methods for category: accessing

nome
"Answer the value of nome"
↑ nome

nome: *anObject*
"Set the value of nome"
nome ← anObject

nomeDoTipo
"Answer the value of nomeDoTipo"
↑ nomeDoTipo

nomeDoTipo: *anObject*
"Set the value of nomeDoTipo"
nomeDoTipo ← anObject

Methods for category: comparing

= *umAtributo*
↑ (self nome = umAtributo nome) and: [self nomeDoTipo = umAtributo nomeDoTipo].

Methods for category: printing

asString
↑ self printString

printString
↑ '<' , self nome , ',' , self nomeDoTipo , '>'

Methods for category: testing

validarX
↑ (ValidadorDeClasses validarAtributo: nome)
and: [ValidadorDeClasses validarNome: self nomeDoTipo]!

NCWorldMorph subclass: OORDBMMorphDiagrama
Category: OORDBM-Widgets
Instance variable names: ultimoErro

Componente principal da janela de edicao de diagramas de classe − contem todos os outros componentes utilizados na criacao do diagrama, alem de rotinas para validar os dados constantes nos diagramas feitos.

Instance protocol

Methods for category: accessing

classes

```
↑ self submorphs
   select: [:morph | morph isKindOf: OORDBMMorphClasse]!
```

herancas

```
↑ self submorphs
   select: [:morph | morph isKindOf: OORDBMMorphHeranca]!
```

relacoes

```
↑ self submorphs
   select: [:morph | morph isKindOf: OORDBMMorphAssociacao]!
```

Methods for category: as yet unclassified

atributosDeRelacionamentoDe: *umOORDBMMorphClasse*

```
| relacionamentos atributos |
relacionamentos ← self relacionamentosDaClasse: umOORDBMMorphClasse.
atributos ← OrderedCollection new.
relacionamentos do: [:relacionamento || papel |
   relacionamento classeOrigem == umOORDBMMorphClasse ifTrue: [ papel ← relacionamento papelOrigem ] ifFalse: [ papel
← relacionamento papelDestino ].
   papel withBlanksTrimmed isEmpty ifFalse: [      atributos add: papel.      ].
].
↑ atributos.
!
```

classesSuperiores

```
| herancas |
herancas ← self herancas.
↑ self classes
        select: [:classe | (herancas
             detect: [:heranca | heranca classeDescendente == classe]
             ifNone: [ ]) isNil]!
```

hierarquiaAPartirDaClasse: *umOORDBMMorphClasse*

```
| retorno |
retorno ← OrderedCollection new.
retorno add: umOORDBMMorphClasse.
(self herancas
   select: [:heranca | heranca classeSuperior == umOORDBMMorphClasse]) do: [:heranca |
   retorno addAll: (self hierarquiaAPartirDaClasse: heranca classeDescendente).
].
↑ retorno.!
```

relacionamentosDaClasse: *umOORDBMMorphClasse*

```
↑ self relacoes select: [:relacao | relacao classeOrigem == umOORDBMMorphClasse
                or: [relacao classeDestino == umOORDBMMorphClasse] ].!
```

ultimoErro

```
↑ ultimoErro
```

Methods for category: converting

exportarComoXMI

```
| documentoXMI |
documentoXMI ← XMIDocumento new.
self exportarClassesEm: documentoXMI.
self exportarHerancasEm: documentoXMI.
self exportarAssociacoesEm: documentoXMI.
↑ documentoXMI!
```

Methods for category: private

exportarAssociacoesEm: *documentoXMI*

```
| associacoes xmiAssoc |
associacoes ← self submorphs
        select: [:morph | morph isKindOf: OORDBMMorphAssociacao].
associacoes
   do: [:assoc |
      xmiAssoc ← XMIAssociacao new.
      xmiAssoc
         classeOrigem: (documentoXMI classeParaNome: assoc classeOrigem nomeDaClasse).
      xmiAssoc
         classeDestino: (documentoXMI classeParaNome: assoc classeDestino nomeDaClasse).
      xmiAssoc
         multiplicidadeOrigem: (XMIMultiplicidade de: assoc grandezaOrigem limiteInferior ate: assoc grandezaOrigem limiteSuperior).
      xmiAssoc
         multiplicidadeDestino: (XMIMultiplicidade de: assoc grandezaDestino limiteInferior ate: assoc grandezaDestino limiteSuperior).
      xmiAssoc papelOrigem: assoc papelOrigem.
      xmiAssoc papelDestino: assoc papelDestino.
      documentoXMI associacoes add: xmiAssoc]
```

exportarClassesEm: *documentoXMI*

```
| classes |
classes ← self classes.
classes
   do: [:classe | documentoXMI classes
        add: (XMIClasse new nome: classe nomeDaClasse)].
classes
   do: [:classe |
      classe listaDeAtributos
         do: [:atributo | (documentoXMI criarClasseParaNome: classe nomeDaClasse) atributos
              add: ((XMIAtributo new nome: atributo nome)
                   tipo: (documentoXMI criarClasseParaNome: atributo nomeDoTipo))].
      classe listaDeMetodos
         do: [:metodo | (documentoXMI criarClasseParaNome: classe nomeDaClasse) metodos
              add: (XMIMetodo new assinatura: metodo)] ]!
```

exportarHerancasEm: *documentoXMI*

```
| herancas |
herancas ← self submorphs
        select: [:morph | morph isKindOf: OORDBMMorphHeranca].
herancas
   do: [:heranca | documentoXMI herancas
        add: (XMIHeranca
             new: (documentoXMI classeParaNome: heranca classeDescendente nomeDaClasse)
             herdaDe: (documentoXMI classeParaNome: heranca classeSuperior nomeDaClasse))]!
```

validarAtributosClasses

```
| grafo |
grafo ← Digrafo new.
self classes
   do: [:classe | grafo adicionaNodo: classe].
self herancas
   do: [:heranca | grafo conecta: heranca classeSuperior a: heranca classeDescendente].
(grafo nodos
   select: [:nodo | (grafo arestasChegandoA: nodo) isEmpty])
   do: [:classe | (self validarAtributosDaHierarquiaIniciandoEm: classe considerandoGrafo: grafo)
        ifFalse: [self inform: 'Existem atributos duplicados na hierarquia'.
            ↑ false] ].
↑ true.!
```

validarAtributosDaHierarquiaIniciandoEm: *classe* **considerandoGrafo:** *grafo*

```
| lista nodosPesquisando classePesquisando |
lista ← OrderedCollection new.
nodosPesquisando ← OrderedCollection with: classe.
[nodosPesquisando isEmpty]
   whileFalse: [classePesquisando ← nodosPesquisando removeFirst.
```

```
        classePesquisando listaDeAtributos
            do: [:atributo | (lista includes: atributo)
                    ifTrue: [↑ false]
                    ifFalse: [lista addLast: atributo] ].
            nodosPesquisando
            addAll: (grafo sucessoresDe: classePesquisando)].
    ↑ true!
```

validarClasses

```
    | classes |
    classes ← self classes.
    classes
        do: [:morph | morph validar
            ifFalse: [self inform: 'A classe ' , morph nomeDaClasse , ' esta com problemas em sua definicao! ' , String cr , String cr ,
morph ultimoErro.
                ↑ false] ].
    classes
        do: [:morph | (classes
                detect: [:morphPesq | morph nomeDaClasse = morphPesq nomeDaClasse
                    and: [(morph == morphPesq) not] ]
                ifNone: [ ]) isNil
            ifFalse: [self inform: 'A classe ' , morph nomeDaClasse , ' esta com nome duplicado'.
                ↑ false] ].
    ↑ true!
```

validarHerancas

```
    self herancas
        do: [:morph | morph validar
            ifFalse: [self inform: 'A heranca ' , morph asString , ' possui problemas: ' , String cr , String cr , morph ultimoErro.
                ↑ false] ].
    ↑ true!
```

validarMetodosClasses

```
    | grafo |
    grafo ← Digrafo new.
    self classes
        do: [:classe | grafo adicionaNodo: classe].
    self herancas
        do: [:heranca | grafo conecta: heranca classeSuperior a: heranca classeDescendente].
    (grafo nodos
        select: [:nodo | (grafo arestasChegandoA: nodo) isEmpty])
        do: [:classe | (self validarMetodosDaHierarquiaIniciandoEm: classe considerandoGrafo: grafo)
            ifFalse: [self inform: 'Existem metodos duplicados na hierarquia'.
                ↑ false] ].
↑ true.!
```

validarMetodosDaHierarquiaIniciandoEm: *classe* **considerandoGrafo:** *grafo*

```
    | lista nodosPesquisando classePesquisando |
    lista ← OrderedCollection new.
    nodosPesquisando ← OrderedCollection with: classe.
    [nodosPesquisando isEmpty]
        whileFalse: [classePesquisando ← nodosPesquisando removeFirst.
            classePesquisando listaDeMetodos
                do: [:metodo | (lista includes: (metodo findTokens: $ ))
                        ifTrue: [↑ false]
                        ifFalse: [lista addLast: (metodo findTokens: $ )] ].
            nodosPesquisando
                addAll: (grafo sucessoresDe: classePesquisando)].
    ↑ true!
```

validarRelacoes

```
    (self submorphs
        select: [:morph | morph isKindOf: OORDBMMorphAssociacao])
        do: [:morph | morph validar
            ifFalse: [self inform: 'A relacao ' , morph asString , ' possui problemas: ' , String cr , String cr , morph ultimoErro.
                ↑ false] ].
    ↑ self validarRelacoesNasHerancas!
```

validarRelacoesNasHerancas

```
  self classesSuperiores
     do: [:classeSuperior |
        | relacionamentos |
        relacionamentos ← Set new.
“          (self atributosDeRelacionamentoDe: classeSuperior) do: [:atributo | (relacionamentos includes: atributo) ifFalse: [
        relacionamentos add: atributo] ifTrue: [self inform: ’Existem papeis de relacionamento duplicados na hierarquia de classes’.
↑ false.] ]."
        (self hierarquiaAPartirDaClasse: classeSuperior)
           do: [:classeDescendente |
(self atributosDeRelacionamentoDe: classeDescendente)do: [:atributo | (relacionamentos includes: atributo) ifFalse: [
        relacionamentos add: atributo] ifTrue: [self inform: ’Existem papeis de relacionamento duplicados na hierarquia de classes’.
↑ false.] ].] ].
↑ true.!
```

Methods for category: testing

validar

```
  ↑ (((self validarClasses
     and: [self validarRelacoes]) and: [self validarHerancas]) and: [self validarAtributosClasses]) and: [self validarMetodosClasses].!
```

NCConnectorMorph subclass: OORDBMMorphHeranca

Category: OORDBM-Widgets
Instance variable names: ultimoErro

Representa uma heranca.

Instance protocol

Methods for category: accessing

classeDescendente
```
↑ self sourceMorph
```
classeSuperior
```
↑ self destinationMorph
```

Methods for category: as yet unclassified

ultimoErro
```
↑ ultimoErro
```

Methods for category: menus

addBasicMenuItemsTo: *aCustomMenu* **event:** *evt*
```
owner ifNotNil: [
   aCustomMenu
      add: 'Mover' target: evt hand selector: #attachMorph: argument: self;
      add: 'Excluir' target: self selector: #delete;
      add: 'Redimensionar' target: self selector: #resizeMorph: argument: evt.
].
```

addYellowButtonMenuItemsTo: *aMenu* **event:** *evt*
```
self addBasicMenuItemsTo: aMenu event: evt.
↑ aMenu.
```

Methods for category: printing

asString
```
↑ self classeSuperior asString , ' → ' , self classeInferior asString!
```

Methods for category: private

validaDescendenciaDe: *aMorph* **considerando:** *herancas*
```
"Answer true if I want my end with index 'index' (1 or 2) to attach to the given Morph."
| heranca morphSuperior |
"Impede 'loops'"
(aMorph == self classeSuperior) ifTrue: [ ↑ false. ].
"Se estou tentando conectar a classe descendente e jah existe uma heranca conectada a mesma, aborte"
"Tambem jah valida se vai gerar loop na heranca"
herancas do: [ :hera |
   (hera classeDescendente == aMorph) ifTrue: [   ↑ false.].
].
"Se nao foi conectada a classe superior ainda, deixe conectar de qq forma"
(self classeSuperior isNil) ifTrue: [ ↑ true. ].
"Obtem heranca da classe superior"
morphSuperior ← self classeSuperior.
heranca ← herancas detect: [:hera | hera classeDescendente == morphSuperior] ifNone: [ ↑true ].
[heranca isNil] whileFalse: [
   (heranca classeSuperior isNil) ifTrue: [ ↑ true. ].
   (heranca classeSuperior == aMorph) ifTrue: [ ↑ false ] ifFalse: [
      heranca ← (herancas detect: [:morph | morph classeDescendente == heranca classeSuperior] ifNone: [ ↑ true ]).
   ].
].
↑ true.
```
validaHerancaPara: *aMorph* **considerando:** *herancas*
```
"Answer true if I want my end with index 'index' (1 or 2) to attach to the given Morph."
| morphSuperior heranca |
"Impede 'loops'"
(aMorph == self classeDescendente) ifTrue: [ ↑false. ].
morphSuperior ← aMorph.
heranca ← herancas detect: [:hera | hera classeDescendente == morphSuperior] ifNone: [ ↑true ].
[heranca isNil] whileFalse: [
   (heranca classeSuperior isNil) ifTrue: [ ↑ true. ].
   (heranca classeSuperior == self classeDescendente) ifTrue: [ ↑ false ] ifFalse: [
      heranca ← (herancas detect: [:morph | morph classeDescendente == heranca classeSuperior] ifNone: [ ↑ true ]).
```

```
    ].
  ].
  ↑ true.
```

Methods for category: testing

validar

```
  (self classeSuperior isNil not
    and: [self classeDescendente isNil not]) ifTrue: [ ↑ true ] ifFalse: [ ultimoErro ← 'Um dos terminais da heranca esta desconectado'. ↑ false ].!
```

wantsToAttachEnd: *index* toMorph: *aMorph*

```
  "Answer true if I want my end with index 'index' (1 or 2) to attach to the given Morph."
  | herancas |
  "Impede que conecte−se a elementos que nao sao classes"
  ((aMorph species = OORDBMMorphClasse) or: [aMorph species inheritsFrom: OORDBMMorphClasse]) ifFalse: [  ↑ false. ].
  "Obtem herancas do diagrama"
  herancas ← self world submorphs select: [:morph | (morph species = OORDBMMorphHeranca) or: [morph species inheritsFrom: OORDBMMorphHeranca] ].
  (index = 1) ifTrue: [ ↑ self validaDescendenciaDe: aMorph considerando: herancas].
  (index = 2) ifTrue: [ ↑ self validaHerancaPara: aMorph considerando: herancas].
```

NCNoteMorph subclass: OORDBMMorphNotaDiagrama

Category: OORDBM-Widgets

Anotacao no diagrama de classes(sem valor semantico)

Instance protocol

Methods for category: as yet unclassified

validar

```
  ↑ true.
```

Methods for category: menus

addBasicMenuItemsTo: *aCustomMenu* event: *evt*

```
  owner ifNotNil: [
    aCustomMenu
      add: 'Mover' target: evt hand selector: #attachMorph: argument: self;
      add: 'Excluir' target: self selector: #delete;
      add: 'Redimensionar' target: self selector: #resizeMorph: argument: evt.
  ].
```

addYellowButtonMenuItemsTo: *aMenu* event: *evt*

```
  self addBasicMenuItemsTo: aMenu event: evt.
  ↑ aMenu.
```

PrefabManager subclass: OORDBMMorphTelaCriarAtributo

Category: OORDBM-Widgets

Instance variable names: janela tiposDeAtributos codigoParaGerarAtributo titulo

Tela onde os tipos de atributos sao disponibilizados ao usuario, permitindo que ele escolha um deles e crie um novo atributo para a classe

Class protocol

Methods for category: as yet unclassified

new: *umTitulo*

```
↑ self new titulo: umTitulo!
```

new: *umTitulo* **blocoDeCodigoParaCriarAtributo:** *umBlocoDeCodigo*

```
↑ self new titulo: umTitulo;
   codigoParaGerarAtributo: umBlocoDeCodigo!
```

teste

```
| tela tt |
tela ← self new.
tela tiposDeAtributos addAll: #('Inteiro' 'Texto' 'Data' 'Hora' 'ValorMonetario' ).
tela codigoParaGerarAtributo: [:valor1 :valor2 | tt ← valor1. tt inspect. ].
tela openInSystemWindow!
```

Instance protocol

Methods for category: as yet unclassified

buildOn: *aManager*

```
"Caution−−automatically−generated method."
aManager
   extent: 259@325;
   color: (Color r: 0.851 g: 0.851 b: 0.851);
   foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
   tag: #corpoTelaCriarAtributo;
   minSize: 10@10;
   maxSize: 1073741823@1073741823.
aManager addClient: (PrefabButton new
   extent: 103@32;
   color: (Color r: 0.851 g: 0.851 b: 0.851);
   foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
   tag: #botaoCancelar;
   text: 'Cancelar';
   action: [janela delete];
   contentOrientation: #centered;
   yourself)
   atRelPosition: 40@278.
aManager addClient: (PrefabLabel new
   extent: 133@14;
   color: (Color r: 0.851 g: 0.851 b: 0.851);
   foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
   text: 'Tipos de dado possiveis';
   contentOrientation: #centered;
   yourself)
   atRelPosition: 11@6.
aManager addClient: (PrefabEntry new
   extent: 130@18;
   color: (Color r: 0.065 g: 0.065 b: 0.065);
   foregroundColor: (Color r: 1.0 g: 1.0 b: 0.935);
   tag: #nomeDoAtributo;
   yourself)
   atRelPosition: 116@233.
aManager addClient: (PrefabLabel new
   extent: 105@43;
   color: (Color r: 0.851 g: 0.851 b: 0.851);
   foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
   text: 'Nome do atributo:';
   contentOrientation: #centered;
   yourself)
   atRelPosition: 9@222.
aManager addClient: (PrefabButton new
```

```
      extent: 103@32;
      color: (Color r: 0.851 g: 0.851 b: 0.851);
      foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
      tag: #botaoCriarAtributo;
      text: 'Criar atributo';
      contentOrientation: #centered;
      action: [self criarAtributo];
      yourself)
      atRelPosition: 143@278.
   aManager addClient: (PrefabListBox new
      extent: 191@172;
      color: (Color r: 0.851 g: 0.851 b: 0.851);
      foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
      elementHeight: 12;
      selectColor: (Color r: 0.771 g: 0.771 b: 0.771);
      selectForeground: (Color r: 1.0 g: 0.0 b: 0.0);
      selectionMode: #single;
      yourself)
      atRelPosition: 32@22.
   aManager addClient: (PrefabListBox new
      extent: 240@173;
      color: (Color r: 0.851 g: 0.851 b: 0.851);
      foregroundColor: (Color r: 0.0 g: 0.0 b: 0.0);
      tag: #listaDeAtributos;
      elementHeight: 12;
      selectColor: (Color r: 0.771 g: 0.771 b: 0.771);
      selectForeground: (Color r: 1.0 g: 0.0 b: 0.0);
      selectionMode: #single;
      yourself)
      atRelPosition: 7@22.
```

codigoParaGerarAtributo: *umBlocoDeCodigo*

```
   codigoParaGerarAtributo ← umBlocoDeCodigo
```

construirListaDeAtributos

```
   | lista |
   lista ← (self clientWithTag: #listaDeAtributos).
   (self tiposDeAtributos) do: [:tipo | lista addLast: tipo. ].
```

criarAtributo

```
   | nomeAtributo tipoAtributo |
   codigoParaGerarAtributo isNil ifFalse: [
nomeAtributo ← ((self clientWithTag: #nomeDoAtributo) text).
   (self clientWithTag: #listaDeAtributos) selection do: [:tipo | tipoAtributo ← tipo text.
codigoParaGerarAtributo value: nomeAtributo value: tipoAtributo ].].
```

initialize

```
   super initialize.
   self buildOn: self.
```

openInSystemWindow

```
   "Put this Manger into a system window. Return the system window."
   "If this Manager is already open (i.e. in a World), close it."
   | nm |
   self world
      ifNotNil: [self delete].
   title isNil
      ifTrue: [self title: self titulo].
   (nm ← Prefab managerFrameClass new) client: self;
      setLabel: title;
      openInWorld.
   janela ← nm.
   janela
      color: (Color
            r: 0.851
            g: 0.851
            b: 0.851).
   janela setLabel: self titulo.
   (self clientWithTag: #nomeDoAtributo)
```

```
    text: 'nomeDoNovoAtributo'.
  self construirListaDeAtributos.
  ↑ nm!

tiposDeAtributos
  tiposDeAtributos isNil ifTrue: [ tiposDeAtributos ← OrderedCollection new ].
  ↑ tiposDeAtributos.
titulo
  titulo isNil ifTrue: [ titulo ← 'Criar atributo' ].
  ↑ titulo!

titulo: umTitulo
  titulo ← umTitulo.!
```

PrefabManager subclass: OORDBMMorphTelaGeracaoProjeto
Category: OORDBM-Widgets
Instance variable names: janela xmi

Tela onde sao definidos os parametros e dado inicio a geracao do sistema modelado.

Instance protocol

Methods for category: API

openInSystemWindow

```
"Put this Manger into a system window. Return the system window."
| nm |
"If this Manager is already open (i.e. in a World), close it."
self world
   ifNotNil: [self delete].
title isNil
   ifTrue: [self title: 'Prefab Manager'].
(nm := Prefab managerFrameClass new)
   client: self;
   setLabel: title;
   openInWorld.
janela ← nm.
janela color: (Color r: 0.851 g: 0.851 b: 0.851).
janela setLabel: 'Geracao de projeto'.
↑nm
```

Methods for category: accessing

bancoDeDados

```
↑ (self clientWithTag: #bancoDeDados) text.
```

categoria

```
↑ (self clientWithTag: #categoriaProjeto) text.
```

gerarNavegadores

```
↑ (self clientWithTag: #gerarNavegadores) isSelected.!
```

persistencia

```
↑ (self clientWithTag: #gerarSuporteAPersistencia) isSelected.!
```

prefixo

```
↑ (self clientWithTag: #prefixoClasses) text.!
```

senha

```
↑ (self clientWithTag: #senha) text.
```

servidor

```
↑ (self clientWithTag: #caminhoServidor) text.
```

usuario

```
↑ (self clientWithTag: #usuario) text.
```

xmi: *umXMI*

```
xmi ← umXMI.
```

Methods for category: creation

gerar

```
| projeto |
self world validar
   ifTrue: [
      janela hide.
      projeto ← OORDBMProjetoTabelaUnica new.
      projeto prefixo: self prefixo.
      projeto categoria: self categoria.
      projeto hostDB: self servidor.
      projeto nomeDB: self bancoDeDados.
      projeto usuario: self usuario.
      projeto senha: self senha.
      projeto gerarNavegadores: self gerarNavegadores.
      projeto xmi: xmi.
      projeto gerarTabelasBD: self gerarTabelas.
      projeto gerarAsClasses: self gerarClasses.
      projeto gerarSuportePersistencia: self persistencia.
      projeto gerarProjeto]!
```

Methods for category: initialization

buildOn: *aManager*

```
"Caution−−automatically−generated method."
aManager extent: 376 @ 396;
   color: (Color
        r: 0.851
        g: 0.851
        b: 0.851);
   foregroundColor: (Color
        r: 0.0
        g: 0.0
        b: 0.0);
   minSize: 10 @ 10;
   maxSize: 1073741823 @ 1073741823.
aManager addClient: (PrefabFrame new extent: 360 @ 31;
      color: (Color
           r: 0.851
           g: 0.851
           b: 0.851);
      foregroundColor: (Color
           r: 0.0
           g: 0.0
           b: 0.0);
      style: #inset;
      yourself) atRelPosition: 8 @ 176.
aManager addClient: (PrefabFrame new extent: 360 @ 31;
      color: (Color
           r: 0.851
           g: 0.851
           b: 0.851);
      foregroundColor: (Color
           r: 0.0
           g: 0.0
           b: 0.0);
      style: #inset;
      yourself) atRelPosition: 8 @ 70.
aManager addClient: (PrefabFrame new extent: 360 @ 31;
      color: (Color
           r: 0.851
           g: 0.851
           b: 0.851);
      foregroundColor: (Color
           r: 0.0
           g: 0.0
           b: 0.0);
      style: #inset;
      yourself) atRelPosition: 8 @ 211.
aManager addClient: (PrefabFrame new extent: 360 @ 31;
      color: (Color
           r: 0.851
           g: 0.851
           b: 0.851);
      foregroundColor: (Color
           r: 0.0
           g: 0.0
           b: 0.0);
      style: #inset;
      yourself) atRelPosition: 8 @ 141.
aManager addClient: (PrefabFrame new extent: 360 @ 31;
      color: (Color
           r: 0.851
           g: 0.851
           b: 0.851);
      foregroundColor: (Color
           r: 0.0
           g: 0.0
           b: 0.0);
```

```
        style: #inset;
        yourself) atRelPosition: 8 @ 106.
    aManager addClient: (PrefabFrame new extent: 360 @ 31;
        color: (Color
            r: 0.851
            g: 0.851
            b: 0.851);
        foregroundColor: (Color
            r: 0.0
            g: 0.0
            b: 0.0);
        style: #inset;
        yourself) atRelPosition: 8 @ 247.
    aManager addClient: (PrefabLabel new extent: 135 @ 18;
        color: (Color
            r: 1.0
            g: 1.0
            b: 1.0);
        foregroundColor: (Color
            r: 0.0
            g: 0.0
            b: 0.0);
        text: 'Categoria:';
        contentOrientation: #centered;
        yourself) atRelPosition: 13 @ 78.
    aManager addClient: (PrefabEntry new extent: 207 @ 18;
        color: (Color
            r: 0.0
            g: 0.0
            b: 0.0);
        foregroundColor: (Color
            r: 1.0
            g: 0.968
            b: 0.903);
        tag: #categoriaProjeto;
        text: 'ProjetoTeste';
        yourself) atRelPosition: 153 @ 77.
    aManager addClient: (PrefabLabel new extent: 135 @ 18;
        color: (Color
            r: 1.0
            g: 1.0
            b: 1.0);
        foregroundColor: (Color
            r: 0.0
            g: 0.0
            b: 0.0);
        text: 'Banco de dados:';
        contentOrientation: #centered;
        yourself) atRelPosition: 13 @ 251.
    aManager addClient: (PrefabCheckButton new extent: 200 @ 30;
        color: (Color
            r: 0.851
            g: 0.851
            b: 0.851);
        foregroundColor: (Color
            r: 0.0
            g: 0.0
            b: 0.0);
        tag: #gerarSuporteAPersistencia;
        text: 'Gerar suporte a persistncia';
        contentOrientation: #left;
        yourself) atRelPosition: 137 @ 293.
    aManager addClient: (PrefabButton new extent: 103 @ 32;
        color: (Color
            r: 0.851
            g: 0.851
            b: 0.851);
```

```
        foregroundColor: (Color
                r: 0.0
                g: 0.0
                b: 0.0);
        text: 'Cancelar';
        action: [janela hide];
        contentOrientation: #centered;
        yourself) atRelPosition: 154 @ 349.
aManager addClient: (PrefabEntry new extent: 207 @ 18;
        color: (Color
                r: 0.0
                g: 0.0
                b: 0.0);
        foregroundColor: (Color
                r: 1.0
                g: 0.968
                b: 0.903);
        tag: #bancoDeDados;
        text: 'projeto';
        yourself) atRelPosition: 150 @ 252.
aManager addClient: (PrefabLabel new extent: 135 @ 18;
        color: (Color
                r: 1.0
                g: 1.0
                b: 1.0);
        foregroundColor: (Color
                r: 0.0
                g: 0.0
                b: 0.0);
        text: 'Usuario:';
        contentOrientation: #centered;
        yourself) atRelPosition: 14 @ 180.
aManager addClient: (PrefabLabel new extent: 287 @ 23;
        color: (Color
                r: 1.0
                g: 1.0
                b: 0.935);
        foregroundColor: (Color
                r: 0.0
                g: 0.0
                b: 0.0);
        text: 'Geracao de projeto';
        contentOrientation: #centered;
        yourself) atRelPosition: 48 @ 14.
aManager addClient: (PrefabButton new extent: 103 @ 32;
        color: (Color
                r: 0.851
                g: 0.851
                b: 0.851);
        foregroundColor: (Color
                r: 0.0
                g: 0.0
                b: 0.0);
        text: 'Gerar';
        contentOrientation: #centered;
        action: [self gerar];
        yourself) atRelPosition: 262 @ 349.
aManager addClient: (PrefabLabel new extent: 135 @ 18;
        color: (Color
                r: 1.0
                g: 1.0
                b: 1.0);
        foregroundColor: (Color
                r: 0.0
                g: 0.0
                b: 0.0);
        text: 'Caminho do servidor:';
```

```
        contentOrientation: #centered;
        yourself) atRelPosition: 14 @ 146.
aManager addClient: (PrefabEntry new extent: 207 @ 18;
        color: (Color
                r: 0.0
                g: 0.0
                b: 0.0);
        foregroundColor: (Color
                r: 1.0
                g: 0.968
                b: 0.903);
        tag: #caminhoServidor;
        text: 'localhost';
        yourself) atRelPosition: 151 @ 146.
aManager addClient: (PrefabEntry new extent: 207 @ 18;
        color: (Color
                r: 0.0
                g: 0.0
                b: 0.0);
        foregroundColor: (Color
                r: 1.0
                g: 0.968
                b: 0.903);
        tag: #usuario;
        text: 'teste';
        yourself) atRelPosition: 152 @ 181.
aManager addClient: (PrefabLabel new extent: 135 @ 18;
        color: (Color
                r: 1.0
                g: 1.0
                b: 1.0);
        foregroundColor: (Color
                r: 0.0
                g: 0.0
                b: 0.0);
        text: 'Senha:';
        contentOrientation: #centered;
        yourself) atRelPosition: 14 @ 216.
aManager addClient: (PrefabCheckButton new extent: 186 @ 30;
        color: (Color
                r: 0.851
                g: 0.851
                b: 0.851);
        foregroundColor: (Color
                r: 0.0
                g: 0.0
                b: 0.0);
        tag: #gerarBancoDeDados;
        text: 'Gerar banco de dados';
        contentOrientation: #left;
        yourself) atRelPosition: 11 @ 316.
aManager addClient: (PrefabCheckButton new extent: 186 @ 30;
        color: (Color
                r: 0.851
                g: 0.851
                b: 0.851);
        foregroundColor: (Color
                r: 0.0
                g: 0.0
                b: 0.0);
        tag: #gerarNavegadores;
        text: 'Gerar navegadores';
        contentOrientation: #left;
        yourself) atRelPosition: 180 @ 316.
aManager addClient: (PrefabEntry new extent: 207 @ 18;
        color: (Color
                r: 0.0
```

```
            g: 0.0
            b: 0.0);
        foregroundColor: (Color
            r: 1.0
            g: 0.968
            b: 0.903);
        tag: #senha;
        text: 'senha';
        yourself) atRelPosition: 152 @ 216.
    aManager addClient: (PrefabCheckButton new extent: 121 @ 30;
        color: (Color
            r: 0.851
            g: 0.851
            b: 0.851);
        foregroundColor: (Color
            r: 0.0
            g: 0.0
            b: 0.0);
        tag: #gerarClasses;
        text: 'Gerar classes';
        contentOrientation: #left;
        yourself) atRelPosition: 11 @ 292.
    aManager addClient: (PrefabEntry new extent: 207 @ 18;
        color: (Color
            r: 0.0
            g: 0.0
            b: 0.0);
        foregroundColor: (Color
            r: 1.0
            g: 0.968
            b: 0.903);
        tag: #prefixoClasses;
        text: 'PT';
        yourself) atRelPosition: 151 @ 112.
    aManager addClient: (PrefabLabel new extent: 135 @ 18;
        color: (Color
            r: 1.0
            g: 1.0
            b: 1.0);
        foregroundColor: (Color
            r: 0.0
            g: 0.0
            b: 0.0);
        text: 'Prefixo das classes:';
        contentOrientation: #centered;
        yourself) atRelPosition: 13 @ 111!
```

initialize

```
    super initialize.
    self buildOn: self.
```

Methods for category: private

gerarClasses

```
    ↑ (self clientWithTag: #gerarClasses) isSelected!
```

gerarTabelas

```
    ↑ (self clientWithTag: #gerarBancoDeDados) isSelected!
```

validarEntradas

```
    ↑ true.
```

ScrollBar subclass: OORDBMScroller

Category: OORDBM-Widgets
Instance variable names: lastValue classesToIgnore type

Scrollbar utilizada por OORDBMDiagramaProjeto para fazer o scroll de OORDBMMorphDiagrama

Class protocol

Methods for category: as yet unclassified

horizontalType
```
↑ 1.
```
new
```
↑ super new initialize.
```
verticalType
```
↑ 0.
```

Instance protocol

Methods for category: accessing

classesToIgnore
```
↑ classesToIgnore.
```
type
```
↑ type.
```
type: *aType*
```
type ← aType.
self descending: (type = self species horizontalType).
```

Methods for category: event handling

hideOrShowScrollBar!

Methods for category: initialization

initialize
```
super initialize.
classesToIgnore ← OrderedCollection new.
lastValue ← value.
self type: self species horizontalType.
self interval: 10.
self model: self.
```

Methods for category: input events

MenuButtonPressed: *arg1*
```
“Automatically generated null response."
“Add code below for appropriate behavior..."
```

Methods for category: private

moveWorld: *aValue*
```
| delta morphsToMove valueX valueY |
delta := aValue − lastValue.
valueX ← type = self species horizontalType ifTrue: [(self owner bounds width × delta) negated] ifFalse: [0].
valueY ← type = self species horizontalType ifFalse: [(self owner bounds height × delta) negated] ifTrue: [0].
delta ← Point x: valueX y: valueY.
self owner isNil    ifTrue: [↑ self].
lastValue ← aValue.
delta isZero ifFalse: [
    morphsToMove := self owner submorphs reject: [:m | m == self or: [m isFlapOrTab] or: [classesToIgnore includes: (m species)] ].
    morphsToMove do: [:m | m position: (m position translateBy: delta) ]
].
```

Methods for category: scrolling

Value: *aValue*
```
“Scrolls the world according to the type and the value received"
self moveWorld: aValue.

  rm
```

validarEntradas
```
↑ true.
```

ScrollBar subclass: OORDBMScroller
Category: OORDBM-Widgets
Instance variable names: lastValue classesToIgnore type

Scrollbar utilizada por OORDBMDiagramaProjeto para fazer o scroll de OORDBMMorphDiagrama

Class protocol

Methods for category: as yet unclassified

horizontalType
```
↑ 1.
```

new
```
↑ super new initialize.
```

verticalType
```
↑ 0.
```

Instance protocol

Methods for category: accessing

classesToIgnore
```
↑ classesToIgnore.
```

type
```
↑ type.
```

type: *aType*
```
type ← aType.

self descending: (type = self species horizontalType).
```

Methods for category: event handling

hideOrShowScrollBar!

Methods for category: initialization

initialize
```
super initialize.

classesToIgnore ← OrderedCollection new.
lastValue ← value.
self type: self species horizontalType.
self interval: 10.
self model: self.
```

Methods for category: input events

MenuButtonPressed: *arg1*
“Automatically generated null response."
“Add code below for appropriate behavior..."

Methods for category: private

moveWorld: *aValue*

```
| delta morphsToMove valueX valueY |

delta := aValue − lastValue.

valueX ← type = self species horizontalType ifTrue: [(self owner bounds width × delta) negated] ifFalse: [0].
valueY ← type = self species horizontalType ifFalse: [(self owner bounds height × delta) negated] ifTrue: [0].
delta ← Point x: valueX y: valueY.
self owner isNil    ifTrue: [↑ self].
lastValue ← aValue.
delta isZero ifFalse: [

    morphsToMove := self owner submorphs reject: [:m | m == self or: [m isFlapOrTab] or: [classesToIgnore includes: (m species)] ].
    morphsToMove do: [:m | m position: (m position translateBy: delta) ]
].
```

Methods for category: scrolling

Value: *aValue*

```
"Scrolls the world according to the type and the value received"

self moveWorld: aValue.
```

10.3 Anexo III - Artigo

Uma ferramenta para a aprendizagem de desenvolvimento de sistemas

Rodrigo Gonçalves

29 de novembro de 2004

Resumo

Apresenta-se aqui uma ferramenta para auxiliar o processo de aprendizagem sobre desenvolvimento de sistemas computacionais. Esta permite a modelagem visual destes através de diagramas de classes, segundo a notação UML. A partir destes, gera-se de forma automatizada o código correspondente (em ambiente Smalltalk), já incluindo o suporte a persistência dos dados envolvidos sobre um banco de dados relacional, através de um framework de persistência.

Palavras-chave: Ensino, Engenharia de Software, Banco de dados.

1 Introdução

O desenvolvimento de sistemas computacionais é uma atividade considerada complexa para alguém iniciante no assunto. Este fato é observado em alunos de disciplinas de programação de caráter introdutório, em cursos como Ciências da Computação e Sistemas de Informação. As dificuldades enfrentadas por estes alunos levaram ao desenvolvimento da ferramenta aqui apresentada.

Esta ferramenta visa auxiliar o aprendizado sobre o desenvolvimento de sistemas computacionais. Através dela é possível modelar visualmente sistemas computacionais de baixa complexidade e gerá-los de forma automatizada. O paradigma de programação orientado a objetos foi adotado para os sistemas modelados e gerados. Um suporte a persistência dos dados envolvidos nos sistemas modelados é fornecido, e este realiza-se sobre um banco de dados relacional.

2 A ferramenta desenvolvida

A ferramenta é composta por duas partes: uma visual, com o intuito de modelar a parte estática de um sistema computacional (denominada *"Modelador de Sistemas"*) e outra encarregada de gerar a estrutura básica do sistema modelado (denominada *"Gerador de Sistemas"*).

A modelagem é feita através do conceito de *"Modelo de Objetos"*, que permite a modelagem da parte estática de um sistema através da representação das entidades que o compõe e seus relacionamentos[Rumbaugh 1991]. É adotada como notação para a modelagem a UML. Desta fornece-se suporte apenas a um subconjunto de seus recursos, que considera-se adequados para a criação de sistemas simples: classes, atributos, métodos, relacionamentos (tanto associações quanto heranças).

Os atributos das classes na modelagem podem ser apenas de um conjunto pré-definido de tipos que visam aproximar-se dos tipos encontrados em sistemas reais, como "Inteiro", "Texto", "Imagem", "Real", "ValorMonetario", etc.

Para uma modelagem, a ferramenta pode gerar: um conjunto de classes, com seus atributos (e respectivos *"getters"* e *"setters"*); um esquema de banco de dados relacional para persistir os dados envolvidos; um sistema de mapeamento dos objetos para o banco relacional; e interfaces visuais simples para a manipulação dos dados envolvidos. Dentro dos atributos das classes, incluem-se aqueles originários de relacionamentos (associações).

Para os sistemas gerados adota-se a linguagem Smalltalk[Lount 2004], em sua implementação conhecida por Squeak[About Squeak 2004]. Esta escolha deve-se a aspectos da linguagem que facilitam tanto a geração dos sistema quanto o entendimento e manipulação dos mesmos pelos usuários.

Para a persistência dos dados utiliza-se o banco de dados PostgreSQL[1] e adota-se um framework existente para Smalltalk chamado GLORP[2]. Este apresenta recursos comuns a *frameworks* do gênero (como cache de objetos, operações baseadas em sessões, etc.) mas também recursos que interessam à ferramenta: um suporte avançado à heranças e polimorfismos; manipulação dos dados pesistidos (recuperação dos mesmos) através de blocos de código da própria linguagem Smalltalk, ao invés de uma linguagem particular. A partir dos blocos de código o GLORP é capaz de gerar uma cláusula *SQL* que recupera os objetos desejados.

[1] Disponível em http://www.postgresql.org/
[2] Disponível em http://www.glorp.org/

3 O Gerador de Sistemas

O *Gerador de Sistemas*, responsável por criar a estrutura básica de um sistema a partir de uma modelagem, é dividido em vários componentes, dos quais destaca-se: o Gerador de Esquema Relacional, o Gerador de Classes, O Gerador da Camada de Persistência e o Gerador de Interfaces.

O *Gerador de Esquemas Relacionais*, dado um diagrama de classes, gera um modelo relacional utilizando conceitos de mapeamento entre os paradigmas orientado a objetos e o paradigma relacional. Dentre as abordagens[W. Ambler 2003], adota-se a que mapeia cada hierarquia de classes[3] para uma única tabela do modelo relacional, utilizando um campo como atributo para determinar a classe na hierarquia de cada um dos objetos persistidos.

O *Gerador de Classes* cria, dentro do ambiente Squeak, as classes do sistema modelado, já estruturando corretamente as relações de herança e associações, além dos atributos de cada classe. As associações tratam os aspectos de multiplicidade através do uso de coleções especiais que limitam a quantidade de elementos que podem estar contidos nas mesmas.

O *Gerador da Camada de Persistência* estabelece no ambiente do Squeak um conjunto de classes responsável pela persistência dos dados do sistema modelado sobre o modelo relacional. Para tal utiliza o framework GLORP. Além disso ele também cria uma entidade denomidada ‘‘*Gerenciador de Persistência*” que encarrega-se de simplifcar o processo de manipulação da persistência através do fornecimento de uma interface (conjunto de métodos) simplificada, onde escondem-se aspectos como o estabelecimento de uma conexão ao banco de dados.

O *Gerador de Interfaces* cria ambientes visuais a partir dos quais as entidades presentes no sistema podem ser manipuladas tanto com relação ao valor de seus atributos quanto à sua persistência (inserir e excluir objetos).

Último elemento do Gerador de Sistemas, o *Gerador de Projetos* faz o papel do próprio Gerador de Sistemas, ou seja, é ele quem utiliza-se dos outros componentes do mesmo para gerar o sistema modelado. É possível fazer uma geração completa ou parcial (onde limita-se a geração à somente a estrutura de classes, por exemplo).

[3]Onde uma hierarquia de classes corresponde a classe mais alta na hierarquia e todas as suas subclasses

4 O Modelador de Sistemas

O Modelador de Sistemas permite modelar visualmente sistemas computacionais utilizando o conceito de “Modelos de Objetos”, conforme descrito anteriormente. Como destina-se a usuários leigos, busca a simplicidade e não a completude em termos de recursos.

A ferramenta tem a aparência mostrada na figura 1. Ela é composta por uma barra de ações (com elementos para salvar e carregar modelagens, iniciar uma nova modelagem e gerar um sistema a partir da modelagem atual), uma barra de elementos e uma área de modelagem.

Figura 1: A interface do Modelador de Sistemas

A barra de elementos contém as entidades que podem compor uma modelagem: classe, herança, associação e nota (sem valor semântico, apenas para permitir ao usuário fazer anotações a respeito da modelagem).

As classes são compostas de três partes principais, onde definem-se o nome, lista de atributos e lista de métodos da classe. A definição destes dá-se diretamente no elemento gráfico, sem a necessidade de acessar telas adicionais. Basta digitar os elementos (no caso dos atributos e métodos, na forma de uma lista vertical). Um atributo é definido através da nomenclatura: *“nomeDoAtributo : tipoDoAtributo”*. Os métodos seguem a sintaxe da linguagem Smalltalk para assinaturas de métodos.

Heranças e associações são definidas através da união das classes integrantes das mesmas. No caso de associações, define-se também os papéis e cardinalidades de cada terminal da associação (novamente diretamente no próprio elemento gráfico).

As validações referentes à modelagem, que incluem nomes de atributos e métodos, tipos de atributos, papéis e cardinalidades de associações, etc. são realizadas apenas quando gera-se o sistema, dando assim total liberdade para o usuário mexer na modelagem.

A geração do sistema funciona como a "ponte" entre o Modelador e o Gerador de Sistemas. Na geração o usuário pode definir aspectos como categoria do Smalltalk onde as classes serão geradas, um prefixo a adicionar ao nomes das mesmas, as configurações relativas ao banco de dados utilizado e por último que elementos serão gerados do sistema - desde as classes até o sistema de persistência.

5 Conclusão

Apresentou-se aqui uma ferramenta para modelagem de sistemas computacionais simples, destinada a leigos no assunto (por exemplo iniciantes em cursos de Ciências da Computação). Ela possui recursos para gerar a estrutura básica de um sistema computacional em Smalltalk (Squeak), assim como um esquema de banco de dados relacional, sobre o qual os dados são persistidos através de um *framework*.

Esta visa tornar-se uma ferramenta mais completa, onde trataria-se todo o processo de desenvolvimento de software, incluindo etapas comuns à maioria dos processos tradicionais ou fixando-se em um processo específico. Assim apresenta-se todo um processo de desenvolvimento de software a alguém iniciante no assunto, que segue o princípio do currículo invertido[Meyer 1993], onde apresenta-se o todo (no caso de Ciências da Computação, um sistema computacional completo) e começa-se a desmontar este todo. Dessa forma, à medida que é decomposto em partes menores, os conceitos e técnicas envolvidas começam a ficar mais claros, pois o aluno já teve uma visão do todo e pode perceber onde eles aplicam-se.

A implementação da ferramenta busca ser expansível, permitindo que a mesma possa vir a trabalhar com outro framework de persistência, de geração de interfaces e banco de dados (inclusive outros paradigmas de persistência). No momento atual a geração de interfaces para manipulação dos dados limitada-se aos atributos simples e não inclui os atributos oriundos de relacionamentos. As relações entre classes também limitam-se apenas a associações, não incluindo composições e

agregações.

Como evoluções a serem feitas na ferramenta inclui-se o suporte a classes presentes no ambiente Smalltalk durante a modelagem dos sistemas (atualmente conta-se apenas com as classes integrantes da modelagem) e uma melhora do gerador de interfaces. A criação de um "assistente" dentro da ferramenta para guiar seu uso também é um recurso que enriqueceria a mesma.

Um último aspecto sobre a ferramenta é que não foi feita uma análise pedagógica e ergonômica sobre a mesma. Estas poderiam ser feitas em conjunto com a aplicação da ferramenta a um conjunto reduzido de usuários, já como forma de teste.

Referências

[About Squeak 2004]ABOUT Squeak. 2004. Disponível em: <http://www.squeak.org/about/index.html>. Acesso em: 28/09/2004.

[Lount 2004]LOUNT, P. W. A brief introduction to smalltalk. 2004. Disponível em: <http://www.smalltalk.org/articles/article_20040000_11.html>. Acesso em: 28/09/2004.

[Meyer 1993]MEYER, B. Towards an object oriented curriculum. 1993.

[Rumbaugh 1991]RUMBAUGH, J. *Object-Oriented Modeling and Design*. [S.l.]: Prentice-Hall International, Inc., 1991.

[W. Ambler 2003]W. Ambler, S. The fundamentals of mapping objects to relational databases. 2003. Disponível em: <http://www.agiledata.org/essays/mappingObjects.html>. Acesso em: 20/05/2004.