"**Antes de imprimir** pense em sua responsabilidade
e compromisso com o **MEIO AMBIENTE**."

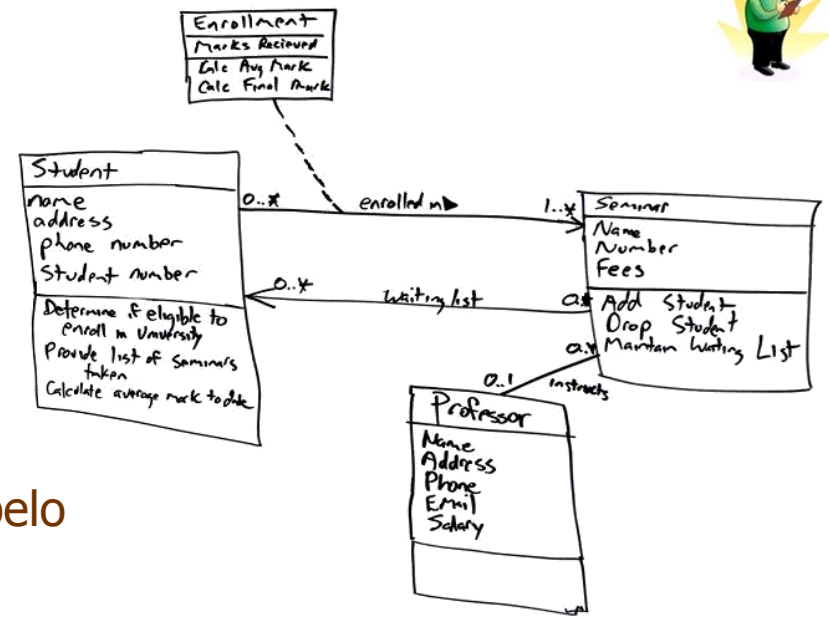
Engenharia de Software

Diagrama de Classe

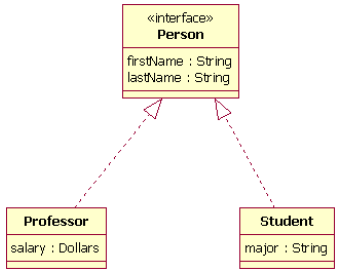
Givanaldo Rocha de Souza
givanaldo.rocha@ifrn.edu.br
http://docente.ifrn.edu.br/givanaldorocha

Material original gentilmente cedido pelo professor Fábio Procópio

Introdução



- Diagrama mais utilizado da UML.
- Permite a visualização das classes utilizadas pelo sistema e como elas se relacionam.
- Apresenta uma visão estática de como as classes estão organizadas a fim de definir sua estrutura lógica.
- Foi projetado para ser uma evolução (e não substituição) do Modelo Entidade-Relacionamento do Banco de Dados.

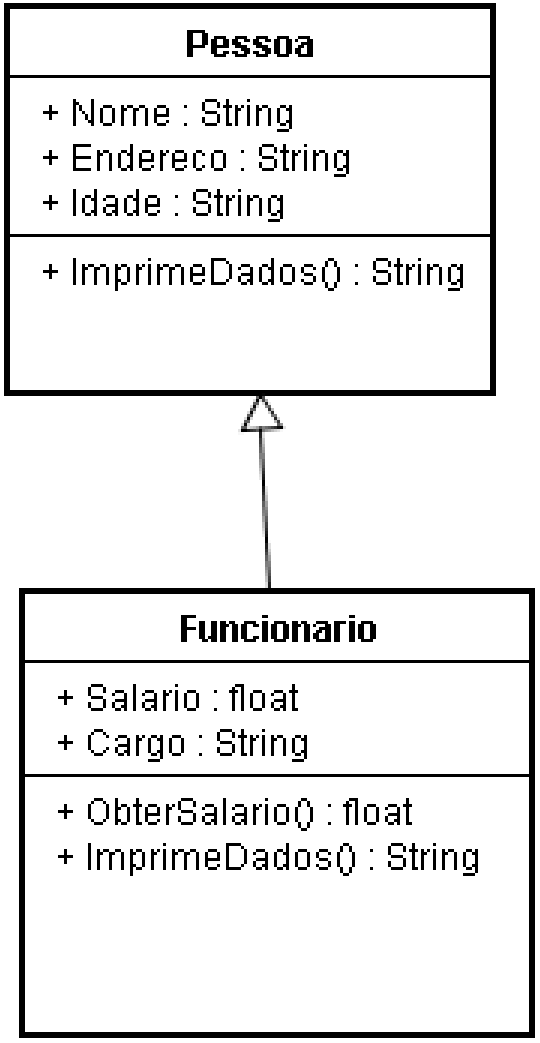


Introdução

- Uma classe não corresponde, obrigatoriamente, a uma tabela em um banco de dados. (Exemplos: *classes de interface, classes de controle*)

- Eventualmente, os atributos de uma classe correspondem aos atributos de uma tabela, porém uma classe não é uma tabela.

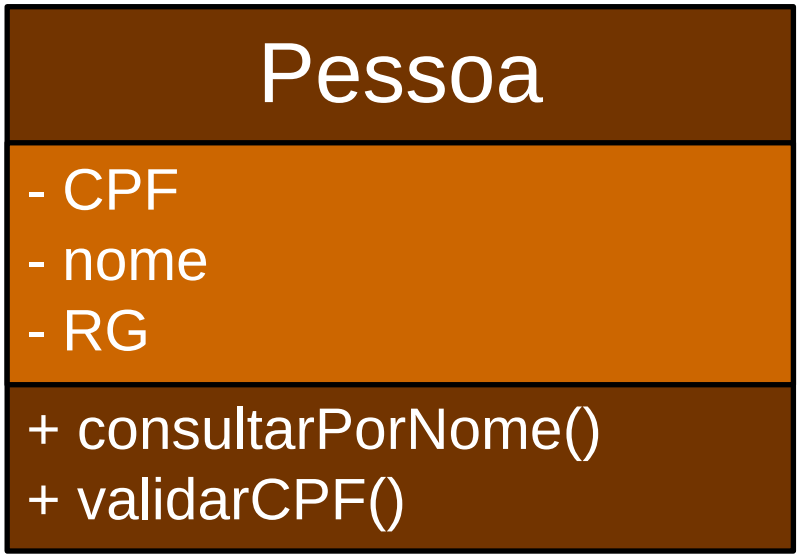
- Em um modelo lógico de Banco de Dados, os métodos de uma classe podem corresponder às operações realizadas sobre uma tabela (incluir, alterar, excluir, consultar).



Classe

- É uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica.

- Representada por um retângulo que pode possuir até três divisões:
  - Nome da classe
  - Atributos da classe
  - Métodos da classe



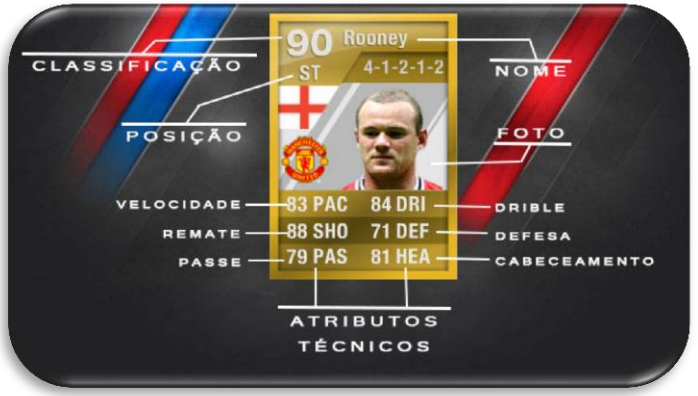
Nome

Atributos
(características)

Métodos
(comportamento)

Relembrando...

- Atributo
  - Representa características de uma classe.
  - Exemplo: Jogador (nome, sexo, idade etc.).



- Método
  - Representa atividades que um objeto de uma classe pode executar.
  - Exemplo: Jogador (correr, driblar, chutar).

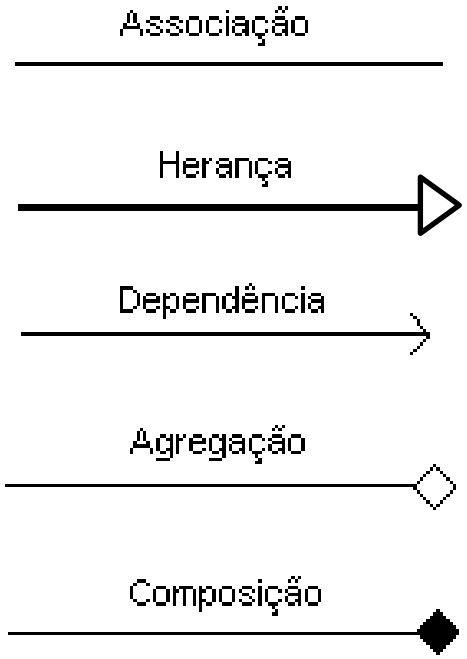
- Visibilidade
  - Indica o nível de acessibilidade de um atributo ou método.
  - Tipos: Pública (+), Privada (-) e Protegida (#).

Relacionamento

- Permite compartilhar informações e colaborar com a execução dos processos do sistema.

- Descreve um vínculo que ocorre, normalmente, entre os objetos de uma ou mais classes.

- Os tipos de relacionamentos são:
  - Associação
    - ✓ Agregação
    - ✓ Composição
  - Especialização/Generalização
  - Dependência



Associação

- Descreve um conjunto de vínculos entre elementos de modelo.

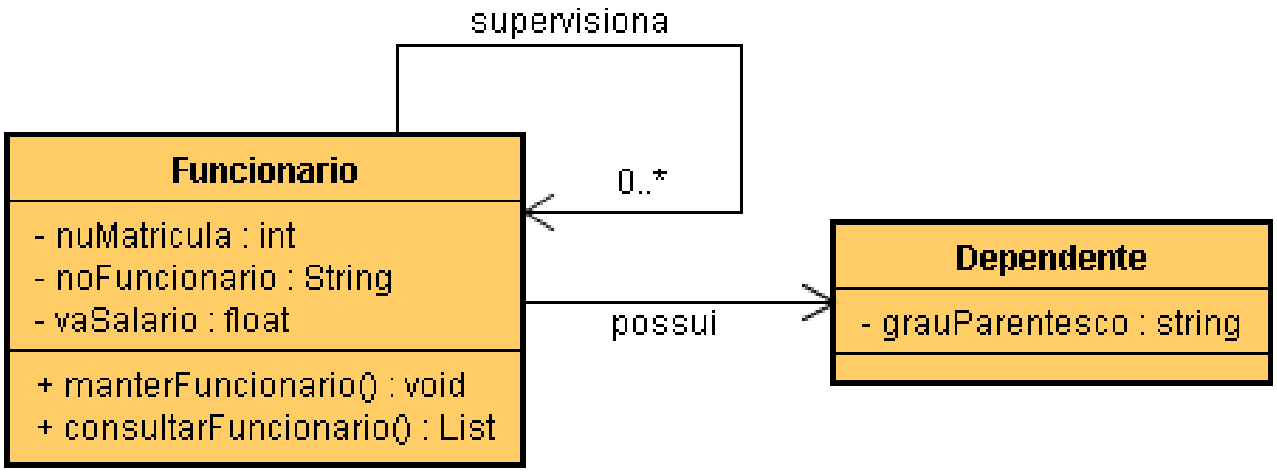
- Relacionamento estrutural que especifica objetos de um item conectados a objetos de outro item:

  - Associação binária – quando há duas classes envolvidas na associação de forma direta de uma para outra.
    - ✓ Relacionamento entre duas classes (tipo mais comum).
    - ✓ Podem possuir títulos para determinar o tipo de vínculo.

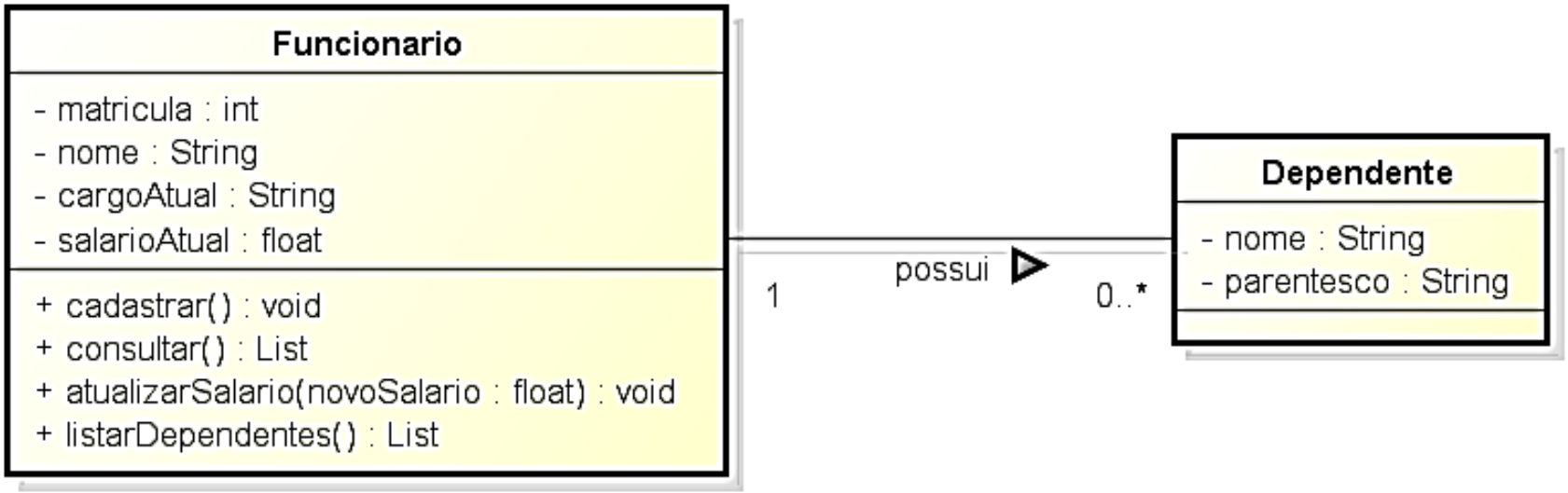
  - Associação unária – quando há um relacionamento de uma classe consigo mesma. Se comparada ao modelo ER, seria um auto-relacionamento.

Associação unária (ou reflexiva)

- Ocorre quando há um relacionamento de um objeto de uma classe com objetos da mesma classe;
- No exemplo abaixo, percebe-se que um objeto da classe Funcionário pode (ou não) supervisionar outros objetos dessa mesma classe;
- Para o relacionamento ficar mais claro, pode-se informar a sua multiplicidade.



Associação binária

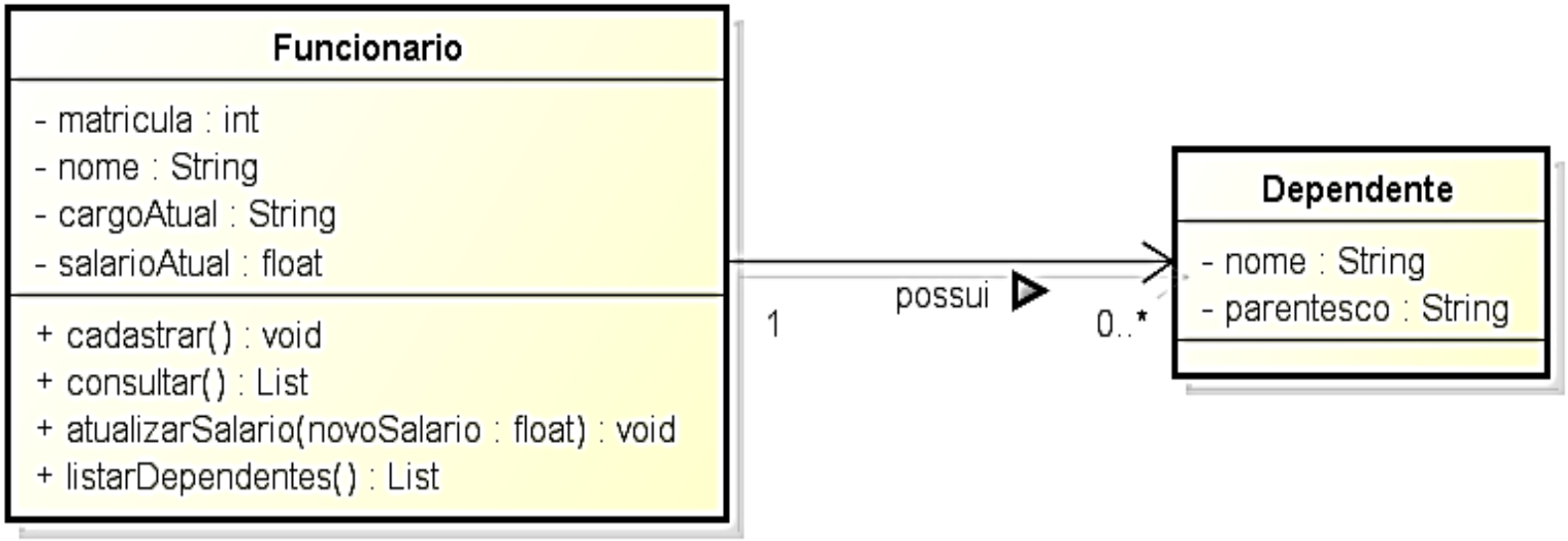



```
public class Funcionario {
    private int matricula;
    ...
    private Dependente[] dependentes;

    // métodos
    ...
}
```

```
public class Dependente {
    private String nome;
    private String parentesco;
    private Funcionario funcionario;
}
```

Associação binária




```
public class Funcionario {
    private int matricula;
    ...
    private Dependente[] dependentes;

    // métodos
    ...
}
```

```
public class Dependente {
    private String nome;
    private String parentesco;
}
```

Multiplicidade

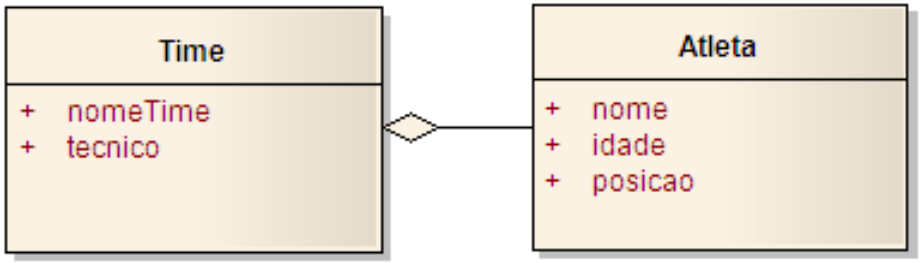
Multiplicidade	Significado
0..1	No mínimo zero e no máximo um. Os objetos não precisam estar relacionados, porém se houver relacionamento deve ser de no máximo 1.
1..1	Um e somente um
0..*	No mínimo nenhum e no máximo muitos.
*	Muitos
1..*	No mínimo um e no máximo muitos.
3..5	No mínimo 3 e no máximo 5.

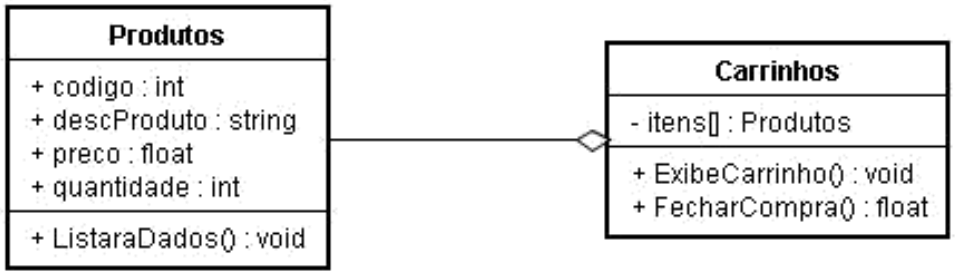
Agregação

- Tipo especial de associação que tenta demonstrar que as informações de um objeto-todo precisam ser complementadas pelas informações contidas em um (ou mais) objetos-parte.

- A existência do objeto-parte faz sentido mesmo não existindo o objeto-todo.

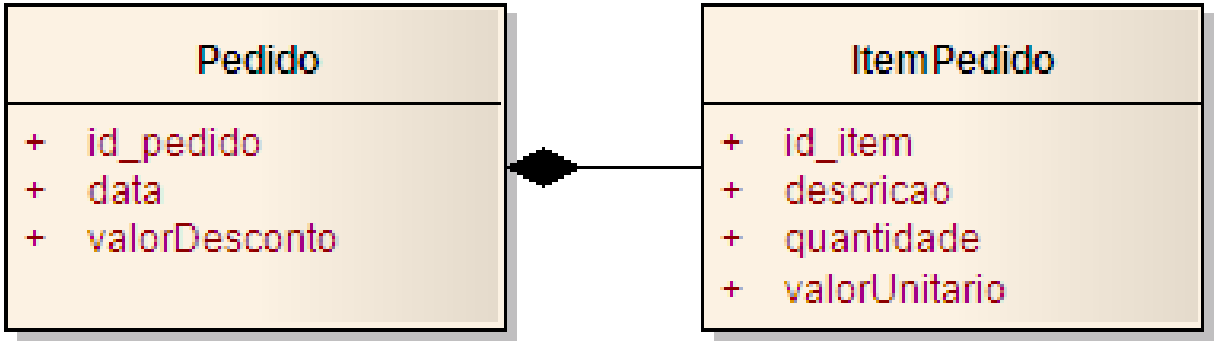
- A associação de agregação pode, em muitos casos, ser substituída por uma associação binária simples, dependendo da visão de quem faz a modelagem.





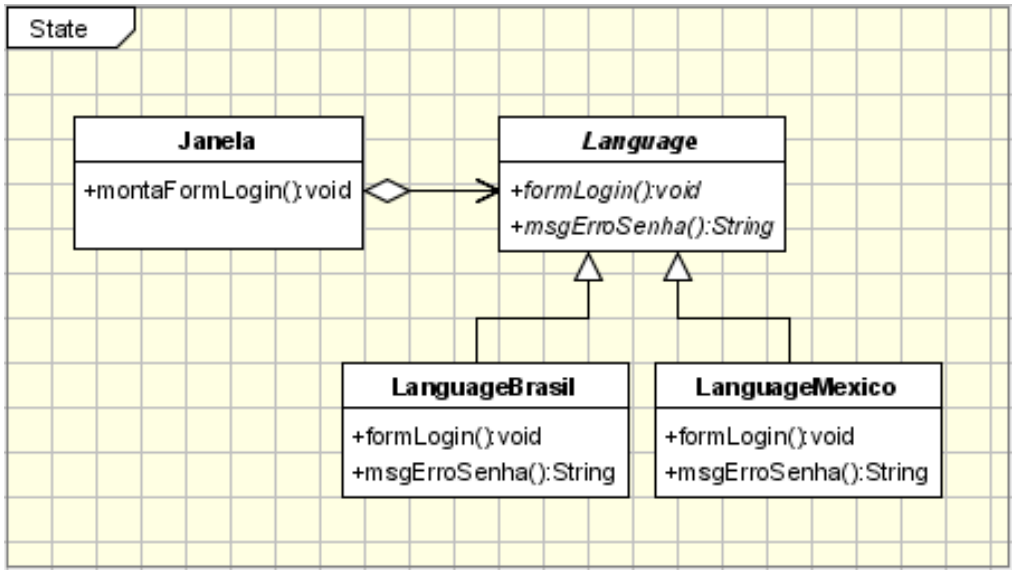
Composição

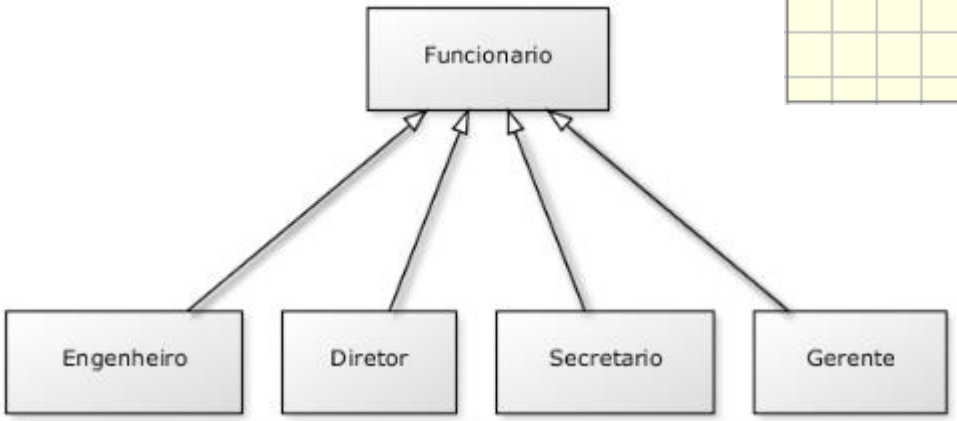
- É uma variação da agregação e considerada mais "forte".
- O objeto-parte não pode existir sem o objeto-todo.
- Se o objeto-todo for destruído, o objeto-parte também será.



Especialização/Generalização

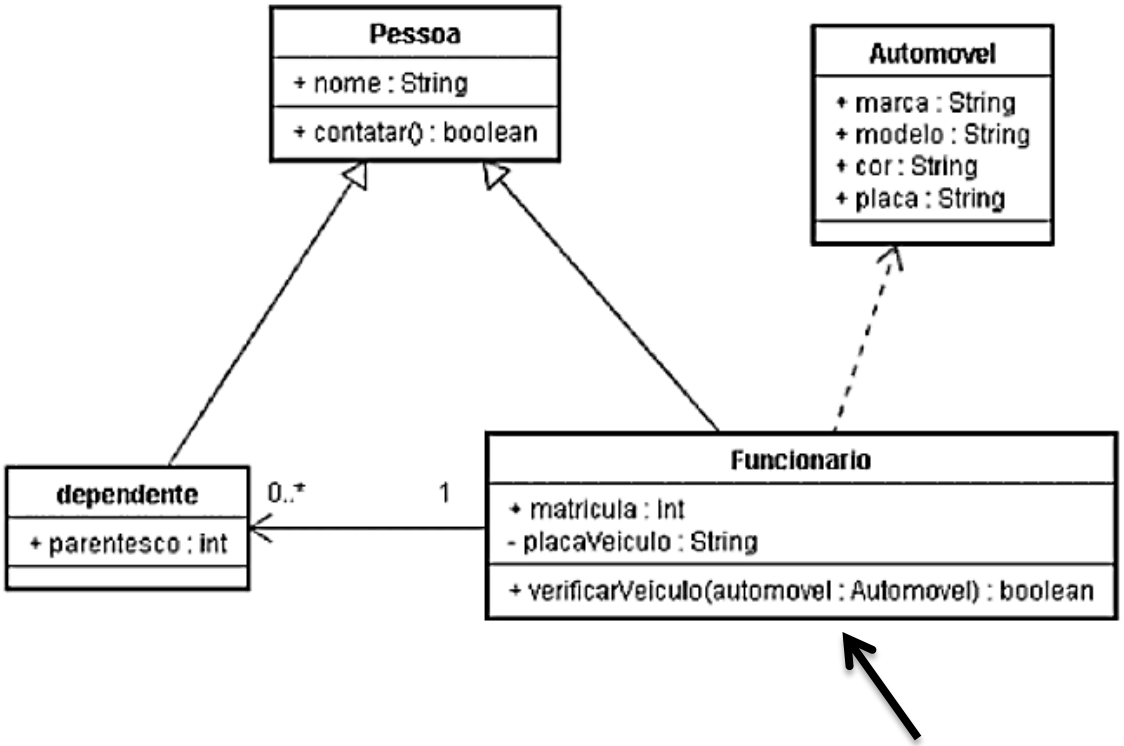
- Tem como objetivo identificar classes-mãe, denominadas de gerais, e classes-filha chamadas de especializadas;
- São chamados de relacionamentos "é um tipo de".





Dependência

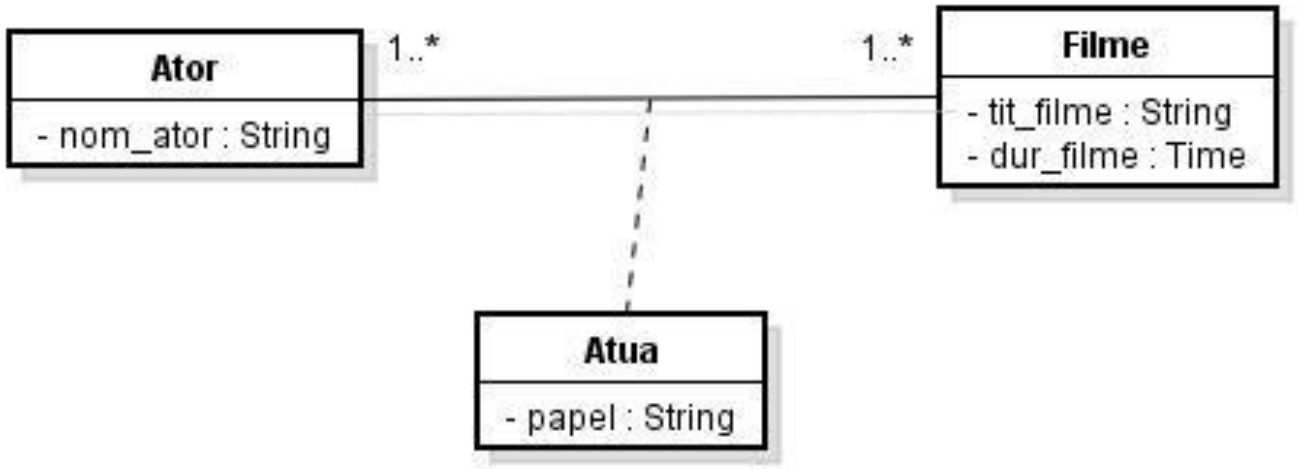
- Como o nome sugere, indica um grau de dependência entre uma classe e outra.
- Uma dependência difere de uma associação porque a conexão entre as classes é temporária.
- Representada por uma seta tracejada entre duas classes.



Funcionário não instancia um Automóvel, apenas usa-o como parâmetro de um método.

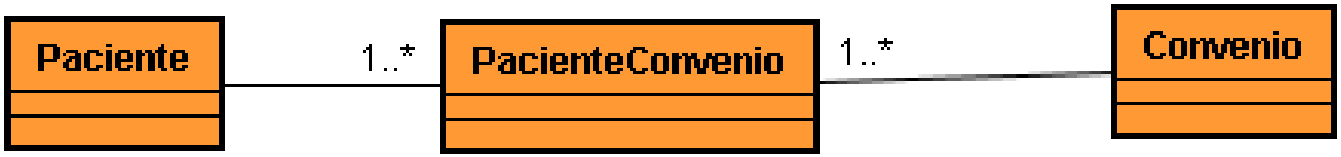
Classe associativa

- Utilizada quando ocorrem associações que possuem multiplicidade muitos para muitos em todas as suas extremidades;
- Armazena os atributos transmitidos pela associação;
- Pode possuir seus próprios atributos;
- Representada por uma reta tracejada partindo do meio da associação até uma classe.



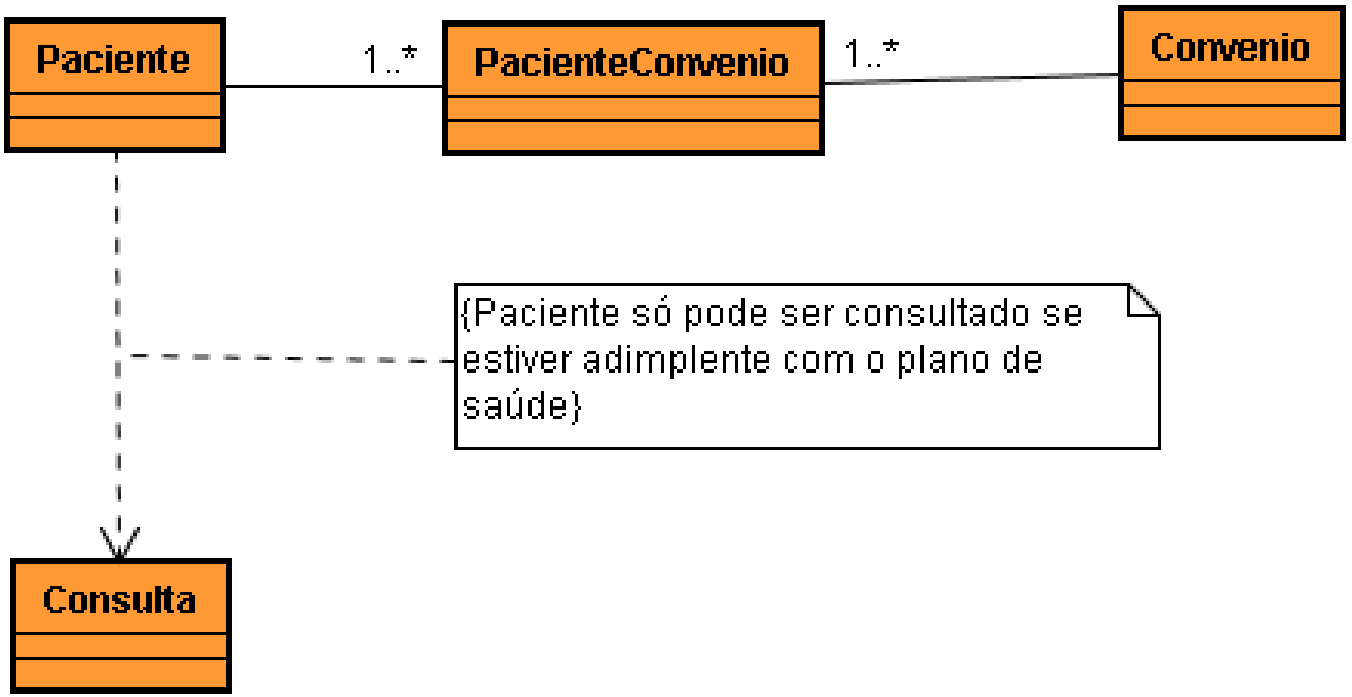
Classe intermediária

- Substitui as classes associativas;
- Apresenta, exatamente, a mesma função da classe associativa;.
- Pode possuir seus próprios atributos;



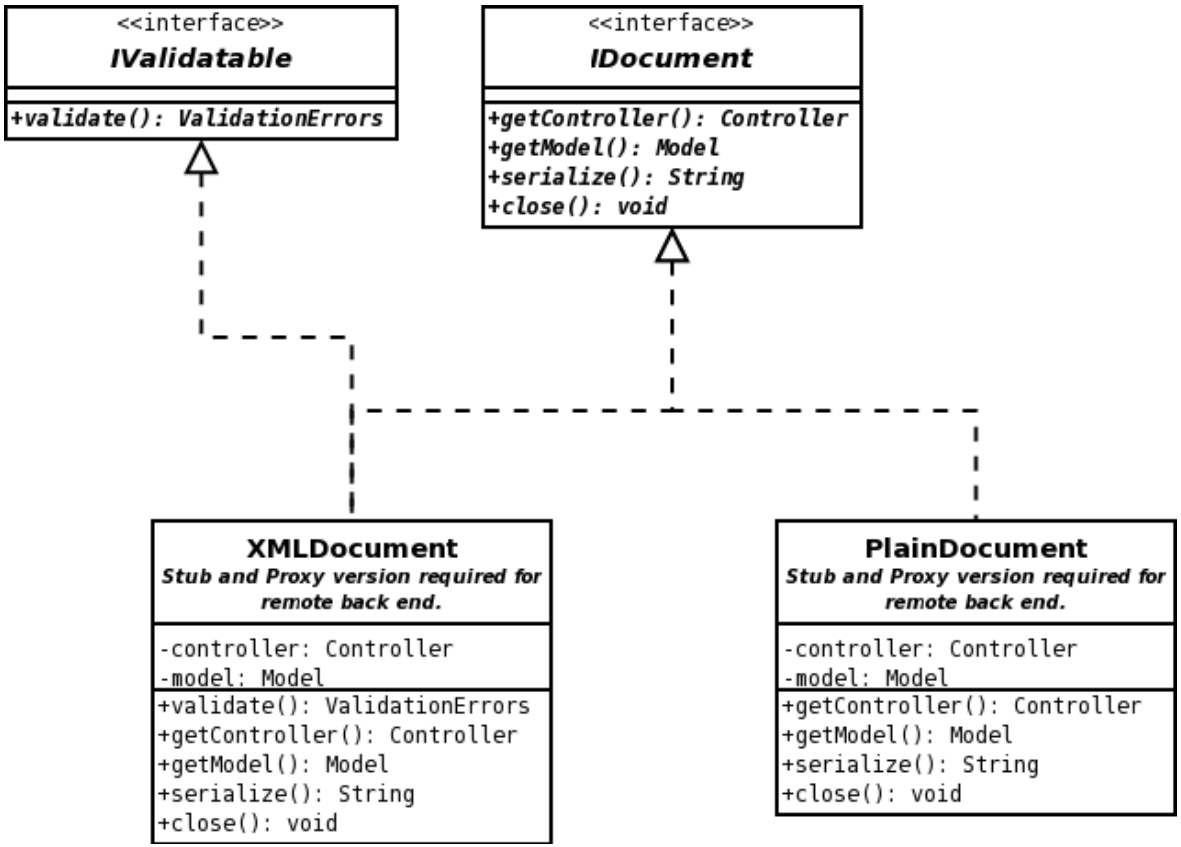
Restrição

- Informações extras que definem condições a serem validadas durante a implementação dos métodos de uma classe, das associações entre as classes ou mesmo de seus atributos;
- Representadas por textos limitados por chaves.



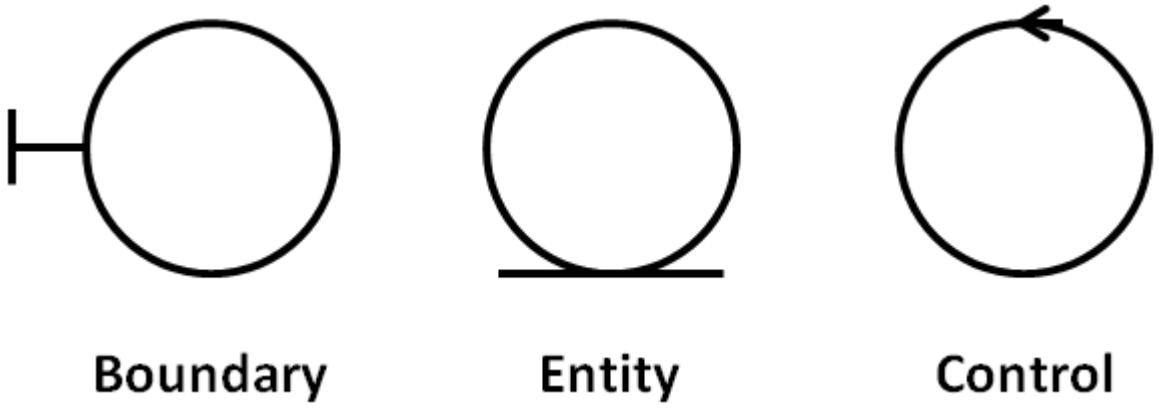
Interface

- Tipo especial de classe a qual não pode ser *instanciada*.
- Serve apenas para especificar operações externamente visíveis para uma outra classe implementar.

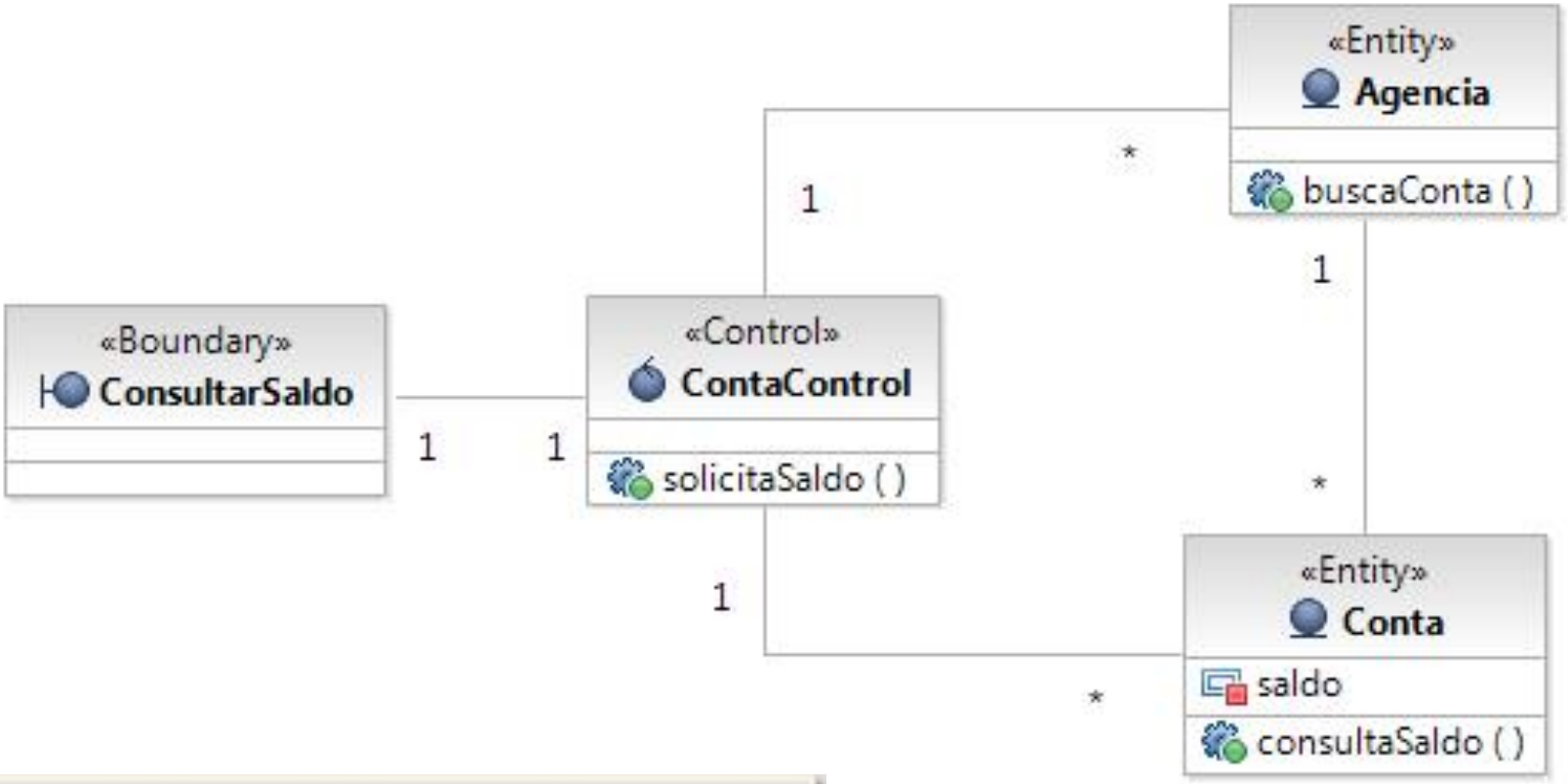


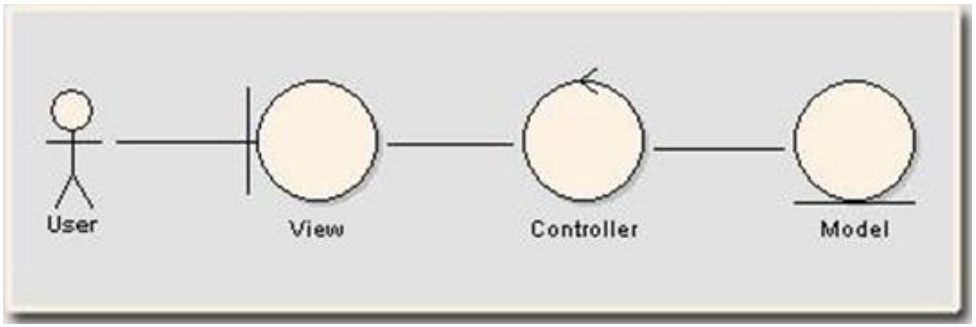
Boundary, Control e Entity

- Classes estereotipadas... Página 270.
- Boundary: classe de fronteira, geralmente interfaces gráficas.
- Control: classe de controle, geralmente implementa as regras de negócio.
- Entity: classe de entidade, geralmente implementa os objetos persistentes.

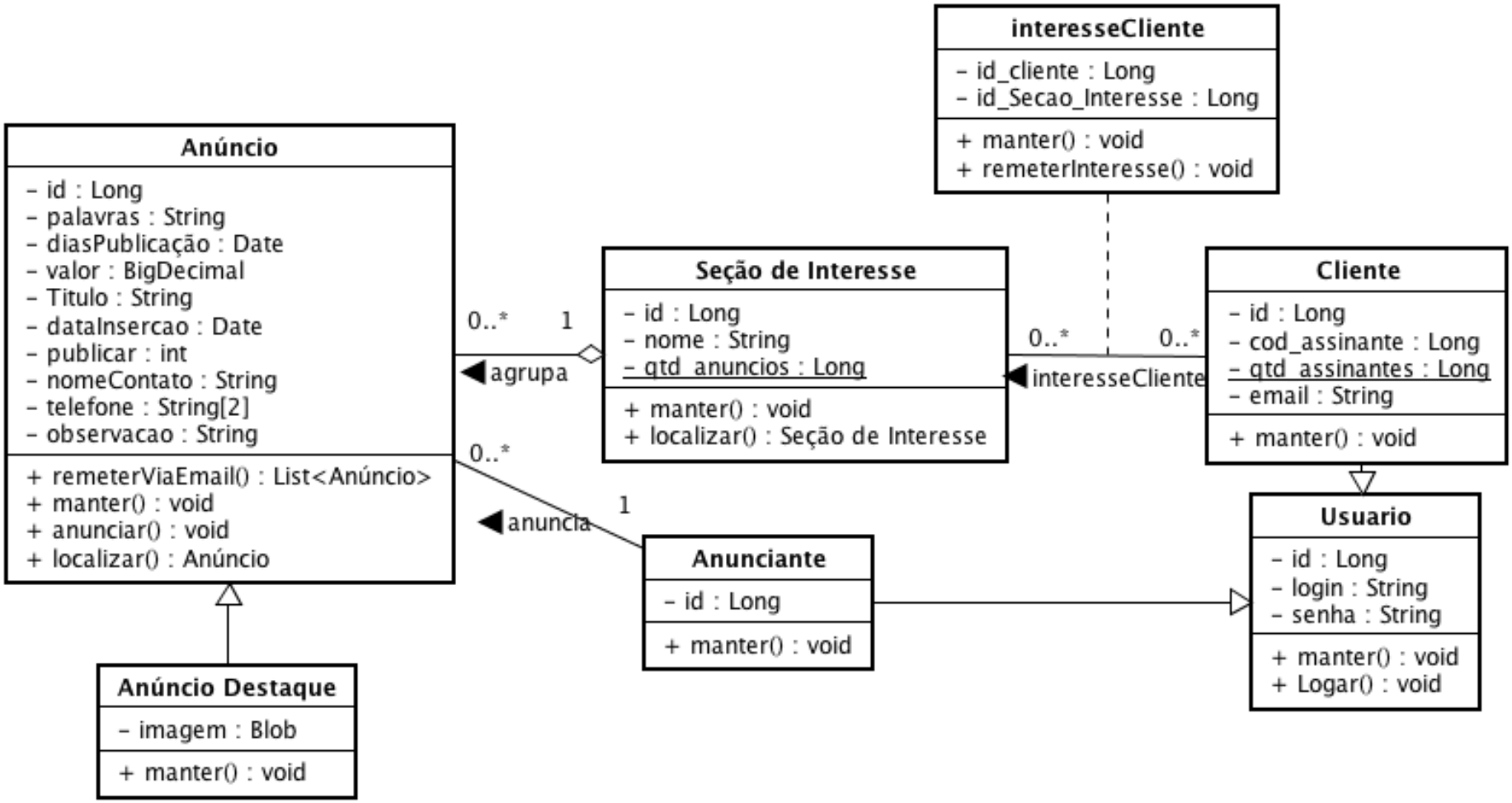


Boundary, Control e Entity

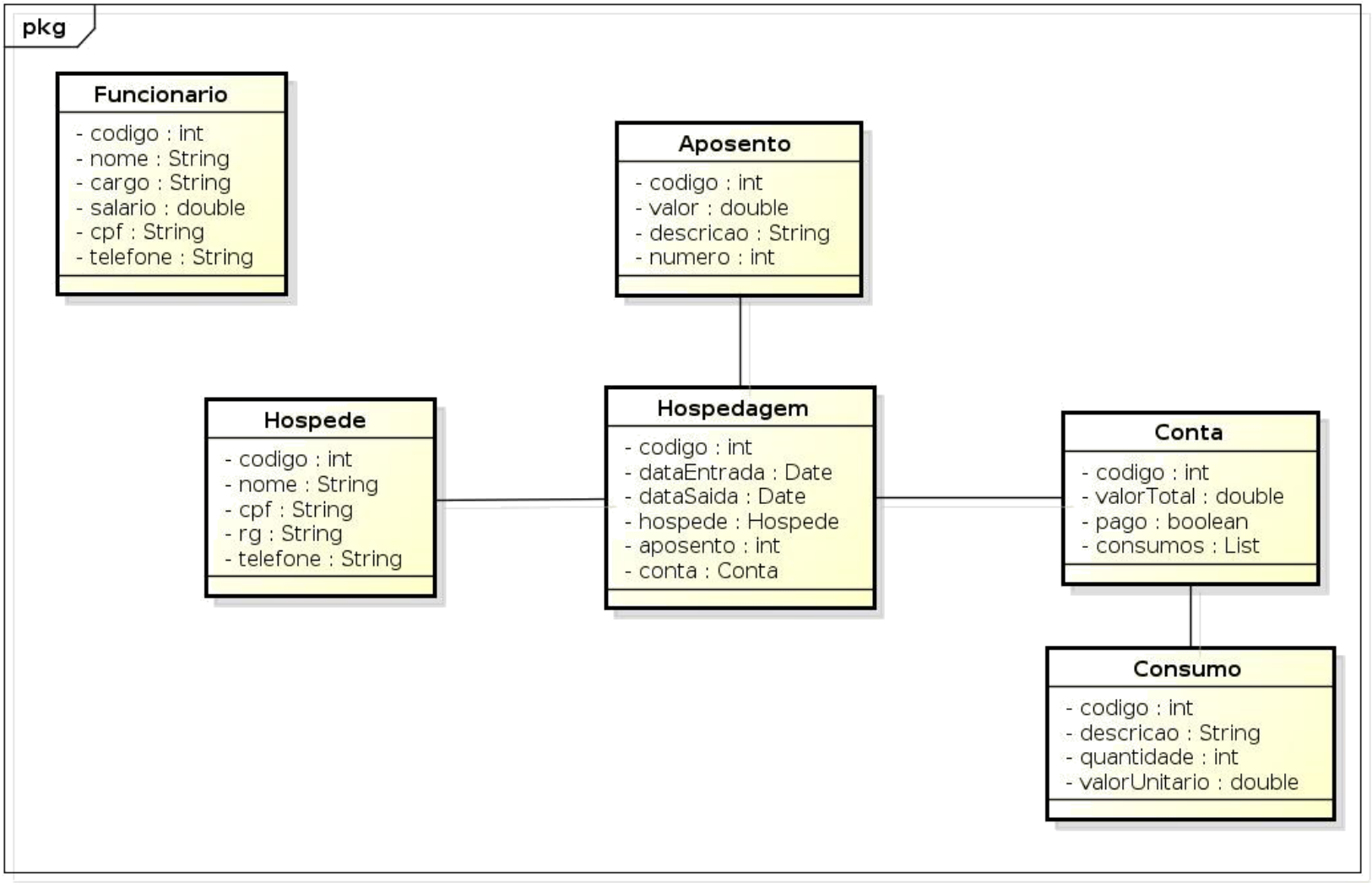


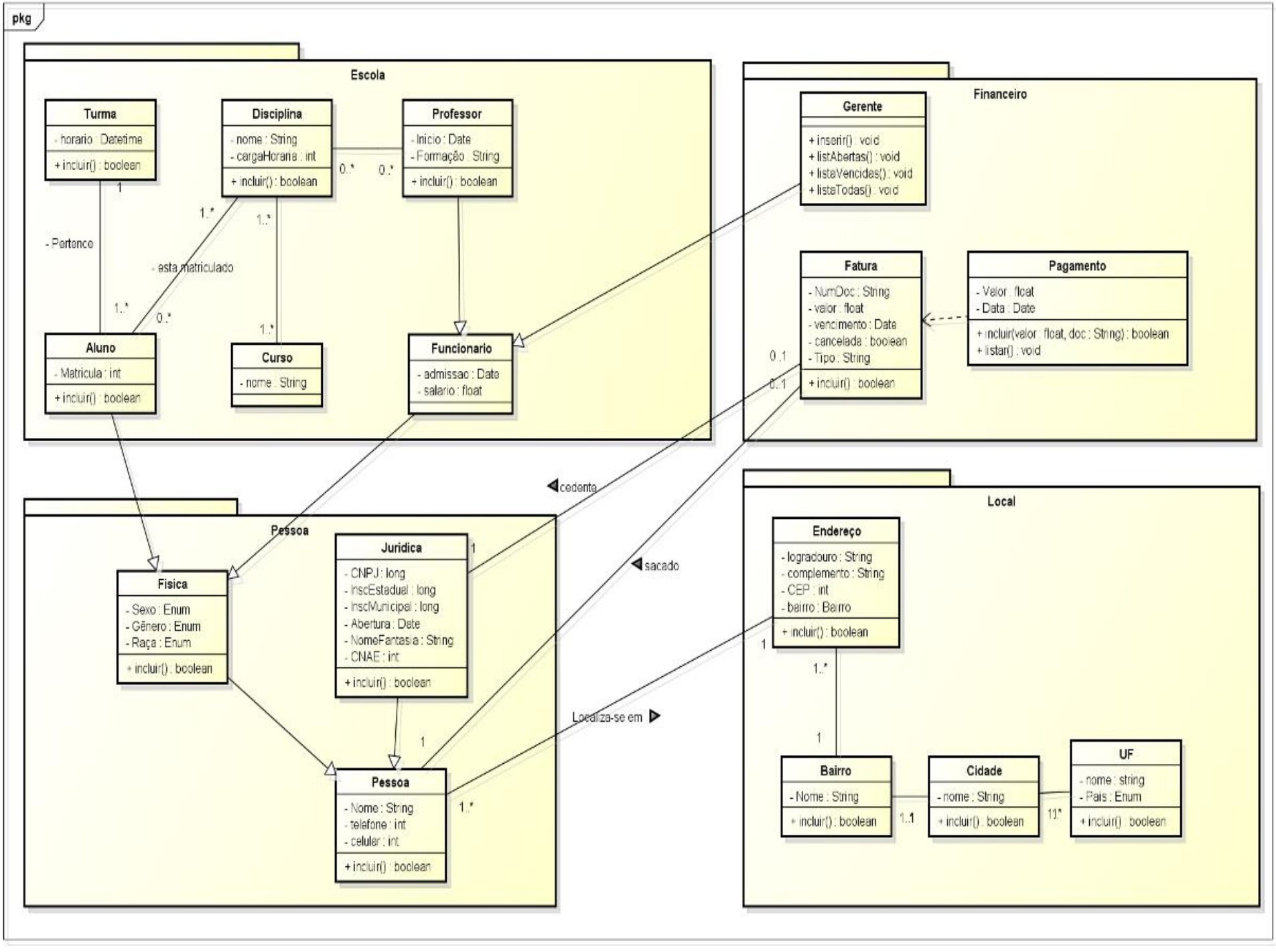


Exemplos de Diagramas de Classe



Exemplos de Diagramas de Classe




pkg
Escola
Turma
- horario : Datetime
+ incluir() : boolean
Disciplina
- nome : String
- cargaHoraria : int
+ incluir() : boolean
Professor
- Inicio : Date
- Formação : String
+ incluir() : boolean
0..*
0..*
1
1..*
1..*
- Pertence
- esta matriculado
1..*
0..*
1..*
Aluno
- Matricula : int
+ incluir() : boolean
Curso
- nome : String
Funcionario
- admissao : Date
- salario : float
Financeiro
Gerente
+ inserir() : void
+ listAbertas() : void
+ listeVencidas() : void
+ listaTodas() : void
Fatura
- NumDoc : String
- valor : float
- vencimento : Date
- cancelada : boolean
- Tipo : String
+ incluir() : boolean
Pagamento
- Valor : float
- Data : Date
+ incluir(valor : float, doc : String) : boolean
+ listar() : void
0..1
0..1
cedente
sacado
Pessoa
Fisica
- Sexo : Enum
- Gênero : Enum
- Raça : Enum
+ incluir() : boolean
Juridica
- CNPJ : long
- InscEstadual : long
- InscMunicipal : long
- Abertura : Date
- NomeFantasia : String
- CNAE : int
+ incluir() : boolean
1
1
Pessoa
- Nome : String
- telefone : int
- celular : int
+ incluir() : boolean
1..*
Localiza-se em
Local
Endereço
- logradouro : String
- complemento : String
- CEP : int
- bairro : Bairro
+ incluir() : boolean
1
1..*
1
Bairro
- Nome : String
+ incluir() : boolean
1..1
Cidade
- nome : String
+ incluir() : boolean
1..*
UF
- nome : string
- Pais : Enum
+ incluir() : boolean

Referências

SIERRA, Katy; BATES, Bert. **Use a cabeça JAVA**. Ed 2, Editora Altabooks.

GUEDES, Gilleanes. **UML Uma Abordagem Prática**. Editora Novatec. São Paulo, 2007.

FURLAN, José. **Modelagem de Objetos através da UML**. Editora Makron Books.

CASTRO, Maurício. **Orientação a Objetos**. Solis/Univates (internet).

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML Guia do Usuário**. Editora Campus.

LIMA, Gleydson. **Diagrama de Classes**. Curso de Especialização em Sistemas Corporativos, FARN/2008.

MACEDO, José Alexandre. **Modelando objetos com cores**. Disponível em: http://jamacedo.com/tag/uml-em-cores/. Acessado em: 16 mai. 2011.

MENDES, Ricardo. **UML: composição x agregação**. Disponível em: http://imasters.com.br/artigo/18901/uml/uml_composicao_x_agregacao/. Acessado em: 26 mai. 2011

TONSIG, Sérgio Luiz. **Engenharia de Software: Análise e Projeto de Sistemas**. 2ª edição. Rio de Janeiro: Editora Ciência Moderna, 2008.