Diagrama de Classes

José Osvano da Silva, PMP

Sumário

- Introdução
- Atributos
- Operações
- Responsabilidades
- Relacionamentos
  - Associação

Sumário

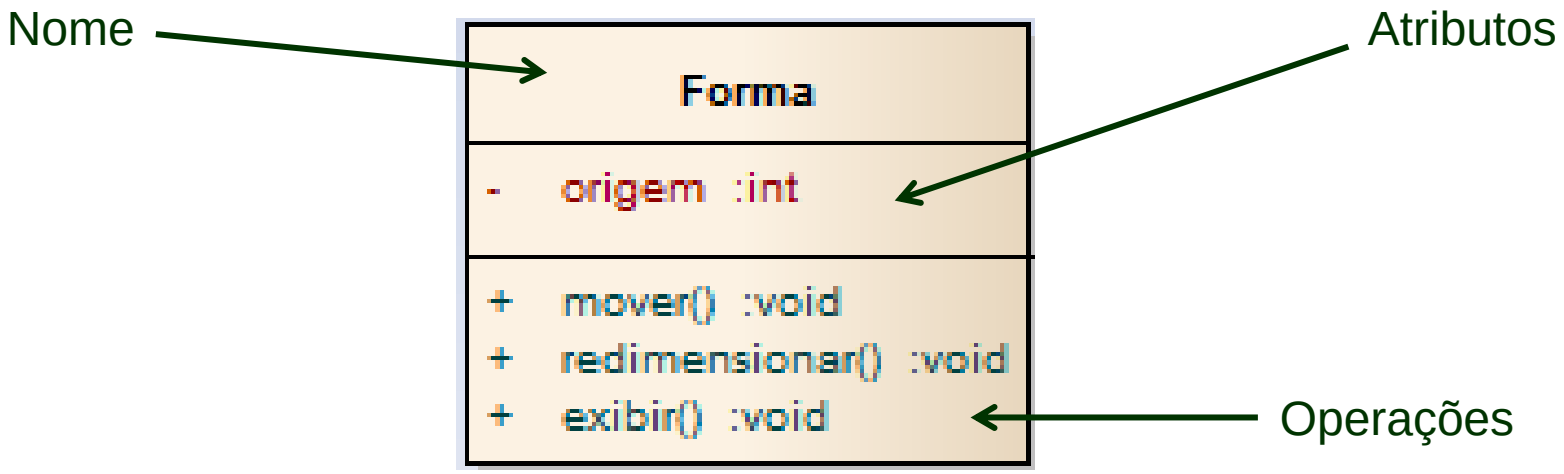
- Relacionamentos
  - Agregação
  - Composição
  - Generalização
  - Especialização
  - Realização
  - Dependência
- Exercícios
- Referências

Introdução

- As classes são os blocos de construção mais importante de qualquer sistema orientado a objetos;
- Uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica;
- Uma classe implementa uma ou mais interfaces.

Introdução

- Uma classe é representada graficamente por um retângulo;



Atributos

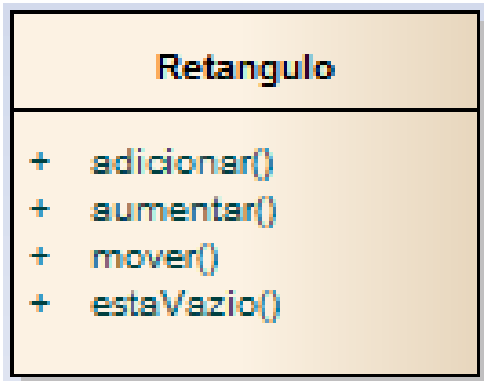
- Um atributo é uma **propriedade** nomeada de uma classe que descreve um intervalo de valores;
- Uma classe pode ter qualquer número de atributos;
- Ou mesmo nenhum atributo;
- Tipicamente aparece como maiúsculo o primeiro caractere de cada palavra existente no seu nome, ex.: suporteDeCarga.

Operações

- Uma operação é a implementação de um **serviço** que pode ser solicitado por algum **objeto** da classe para modificar seu **comportamento**;
- Pode ter qualquer número de operações ou nenhuma;
- Tipicamente aparece como maiúsculo o primeiro caractere de cada palavra existente no seu nome, ex.: estaVazio.

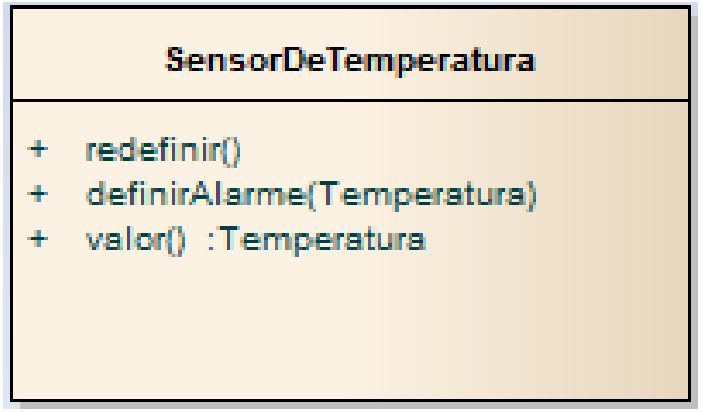
Operações

- As operações podem ser representadas, exibindo somente seus nomes.



Operações

- As operações podem ser especificadas indicando sua assinatura, que contem o nome, o tipo e o valor-padrão de todos os parâmetros

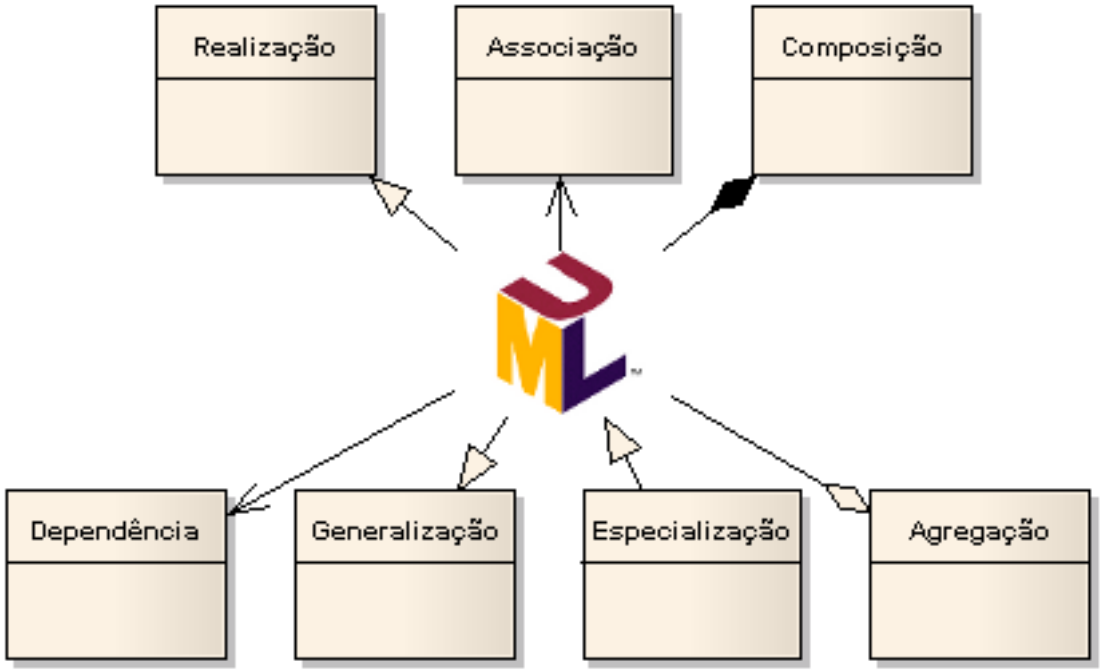


Responsabilidades

- Uma responsabilidade é um **contrato** ou obrigações de uma determinada classe;
- Ao criar uma classe está sendo criado **uma declaração** de que todos os objetos dela tem o mesmo tipo de estado e o mesmo tipo de comportamento;
- As **responsabilidades** são apenas texto em formato livre. Na prática uma única responsabilidade pode ser escrita como uma expressão, uma oração ou um breve parágrafo.

Relacionamentos

- Os relacionamentos ou associações definem as dependências e ligações entre as classes, objetos, pacotes, tabelas, entre outros.



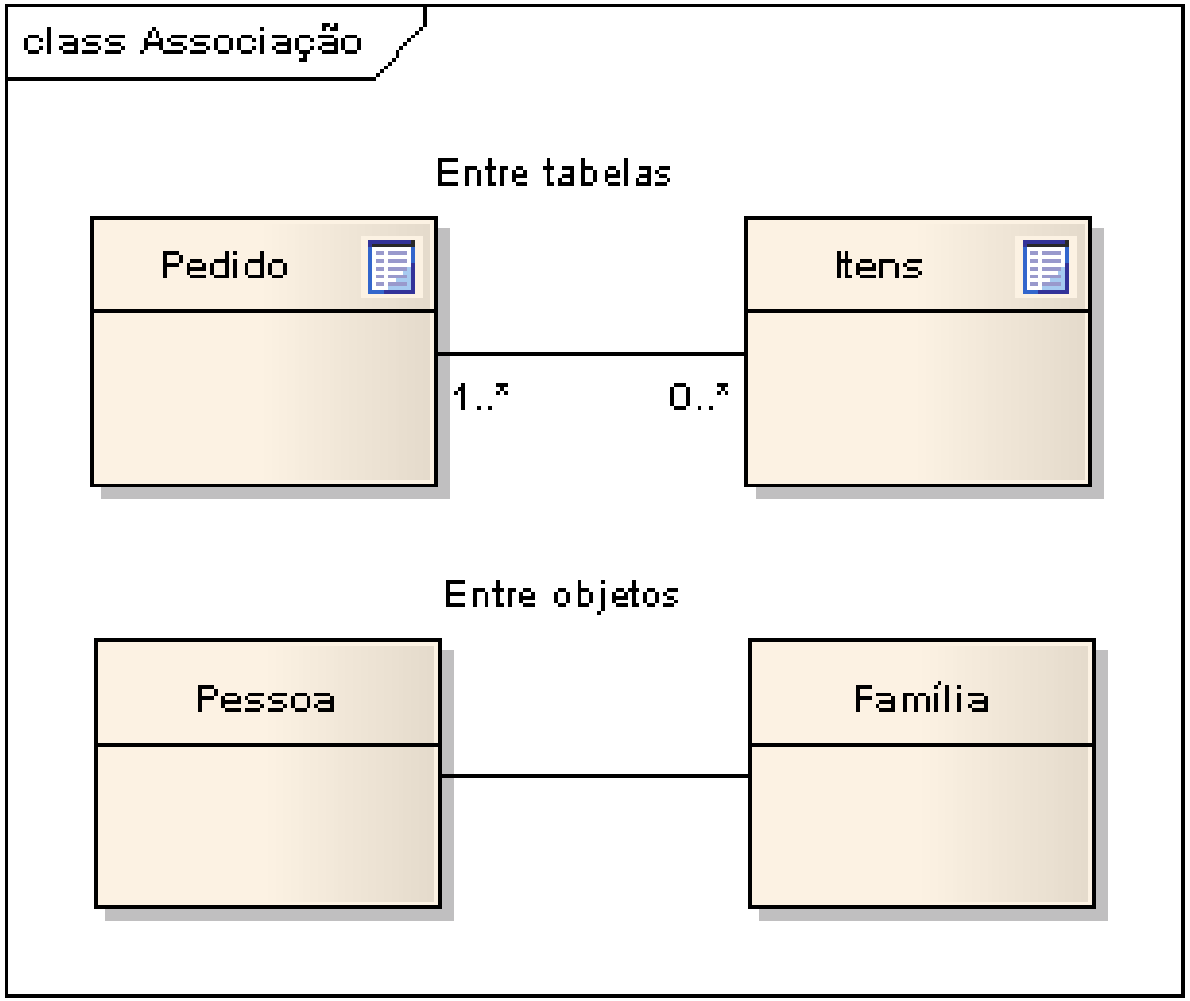
Relacionamentos - Associação

- É uma conexão entre as classes, tabelas e objetos. Um relacionamento;
- Em UML uma associação representa um relacionamento que descreve as ligações entre os objetos ligados;
- Uma associação deve ter sempre duas pontas, onde uma é o objeto de início e a outra o objeto final.

Relacionamentos - Associação

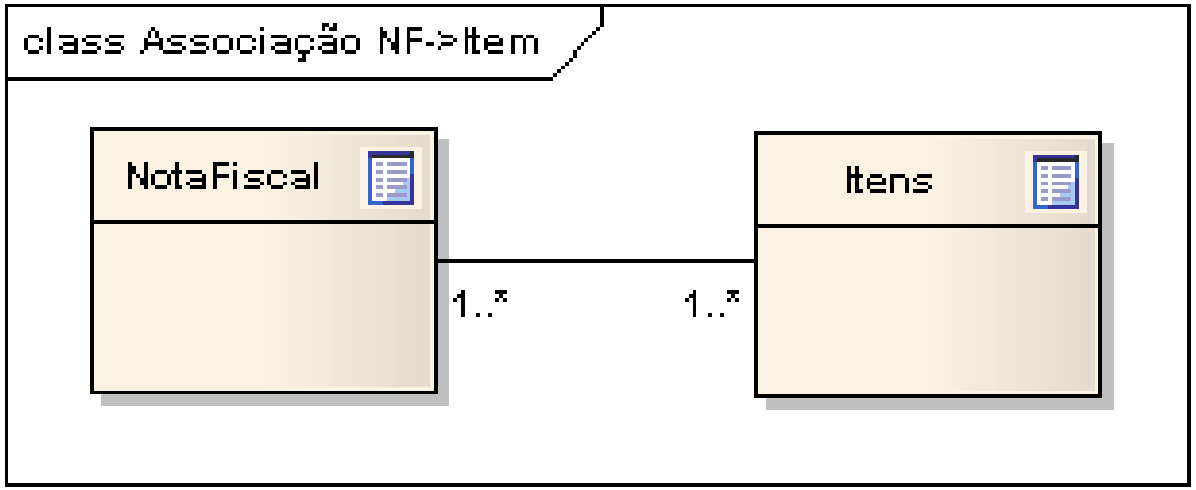
- Uma associação pode representar a multiplicidade entre os objetos.
- Podemos ter as seguintes representações de multiplicidade:
  - 0 (zero)
  - 1 (um)
  - 0…1 (zero ou um)
  - 0…* (zero ou mais)
  - 1…* (um para muitos)
  - * (muitos)

Relacionamentos - Associação



Relacionamentos - Associação

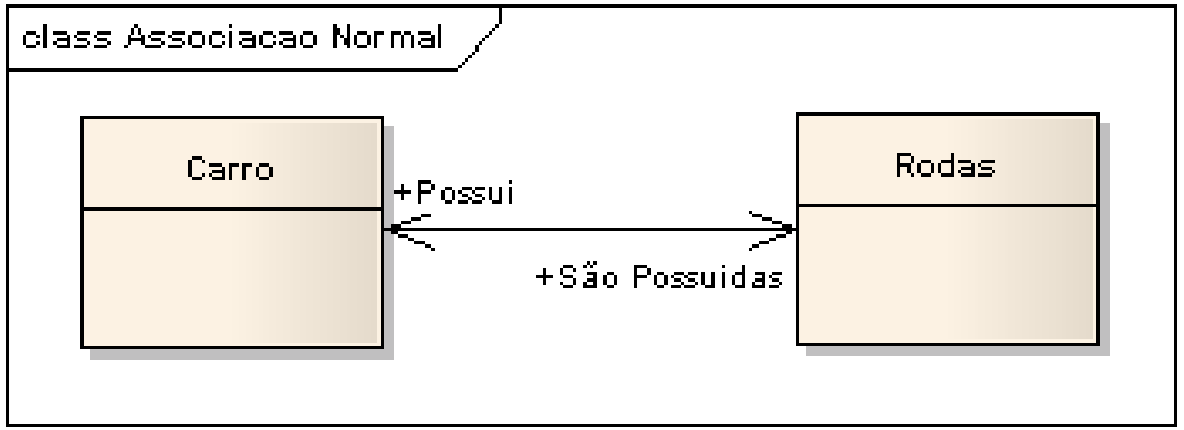
- Por exemplo, podemos pegar uma Nota Fiscal e seus Itens.



- Uma nota fiscal, vários itens.

Relacionamentos - Associação

- Normais:
  - É o tipo mais comum, é **representada por uma linha sólida entre as classes**. Possui um nome em cada ponta que identifica o que cada uma representa, normalmente é um **verbo**. Mas fica a critério de cada um para facilitar o entendimento.



Relacionamentos - Associação

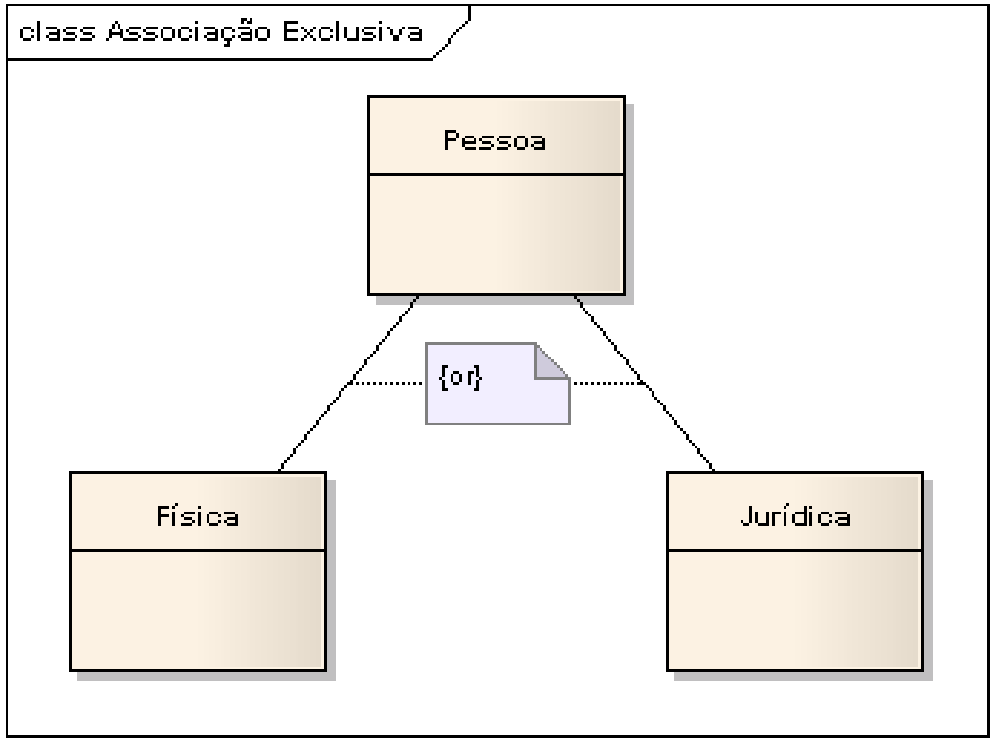
- Recursiva:
  - Indica que uma classe pode conectar-se a ela mesma. Por exemplo. Pegamos o objeto Pessoa. Ele é um objeto que pode ser um Pai, um Filho, um Fornecedor, um Cliente, um Usuário. Mas todos estes são “Pessoas”.

Relacionamentos - Associação

- Exclusiva:
  - Em alguns momentos precisamos representar que um objeto pode participar apenas de um tipo em determinado momento.
  - Uma associação exclusiva é representada por um linha tracejada entre as associações, entre elas existe a especificação {ou}.

Relacionamentos - Associação

- Exclusiva:
  - A pessoa não pode ser Jurídica e Física ao mesmo tempo.

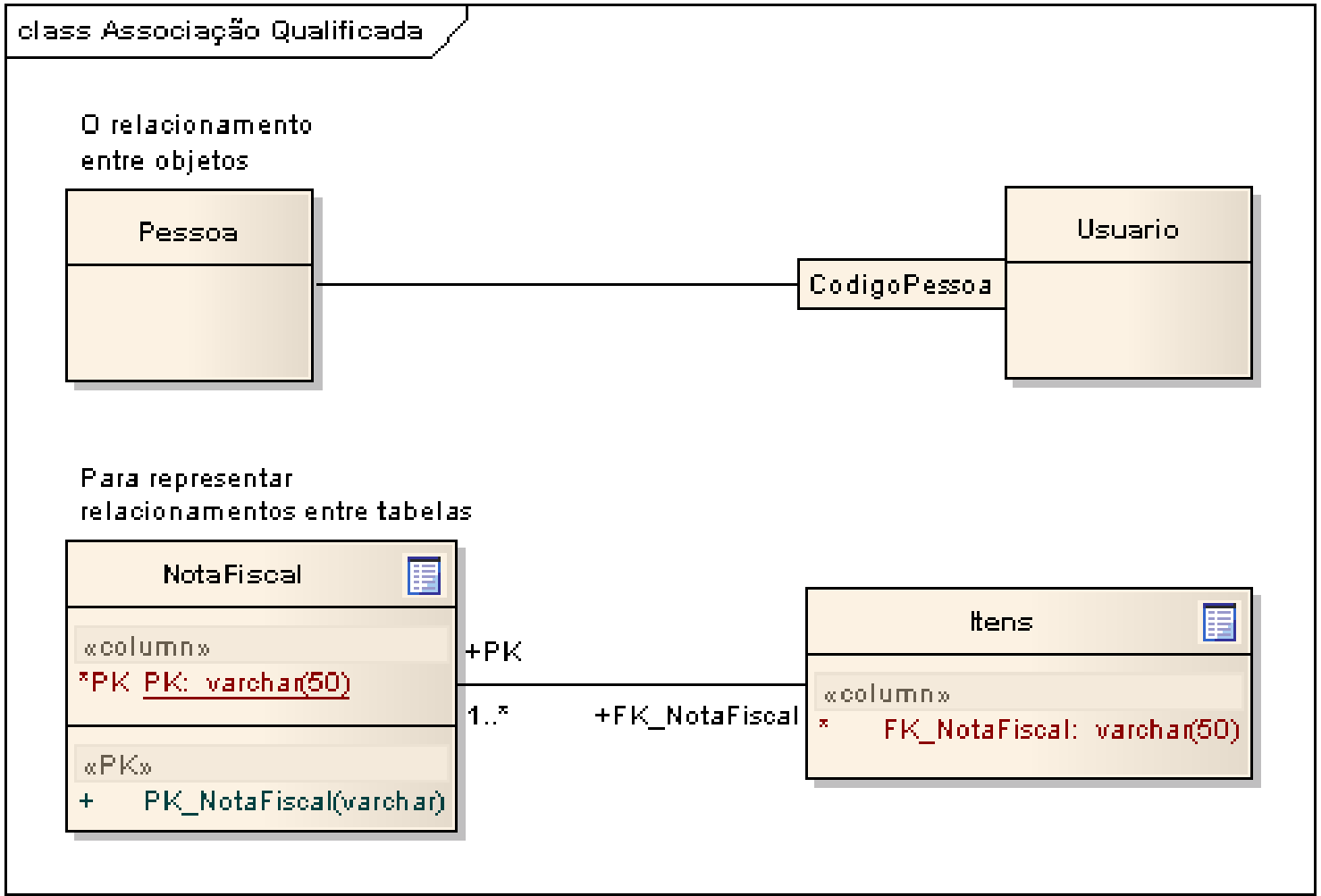


Relacionamentos - Associação

- Qualificada:
  - As associações qualificadas representam as ligações de 1…* (um para muitos) ou * (muitos).
    O “qualificador” identifica no final da associação qual o objeto está identificado.

Relacionamentos - Associação

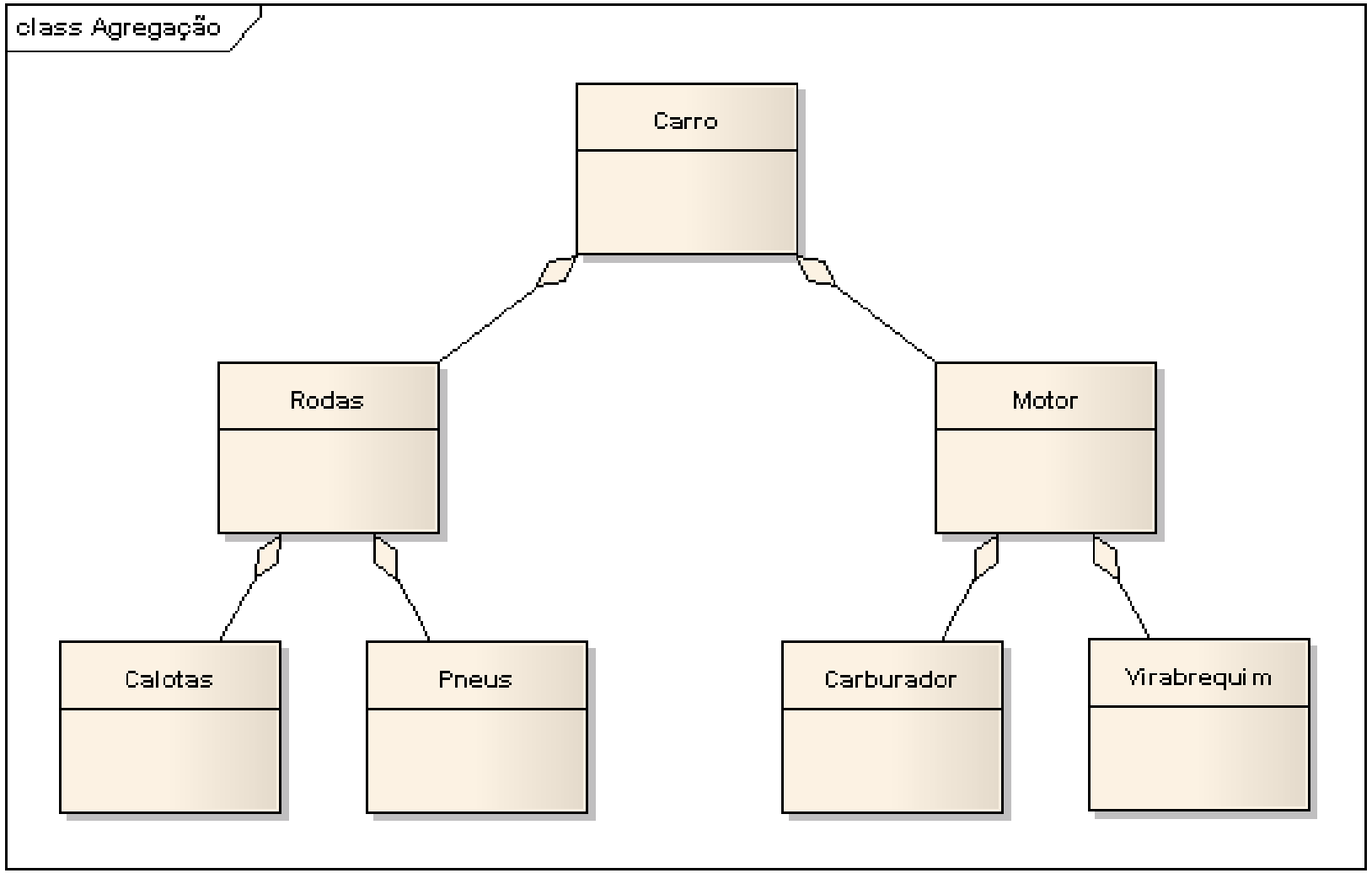
- Qualificada:



Relacionamentos - Agregação

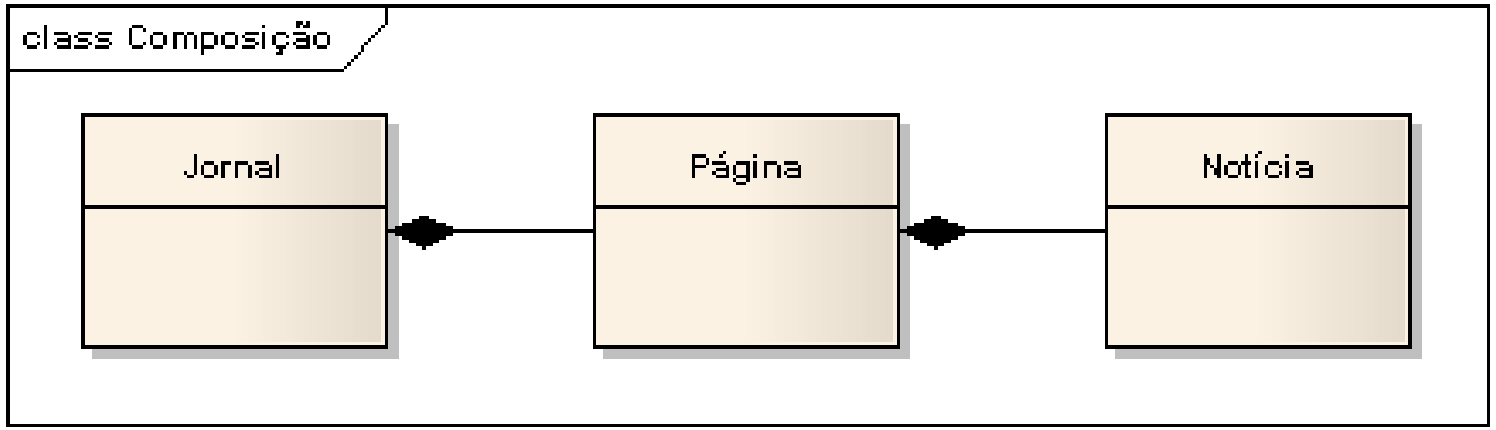
- A agregação é um tipo de associação onde o todo está relacionado com suas partes. É representada com o símbolo de um diamante junto a classe agregadora.

Relacionamentos - Agregação



Relacionamentos - Composição

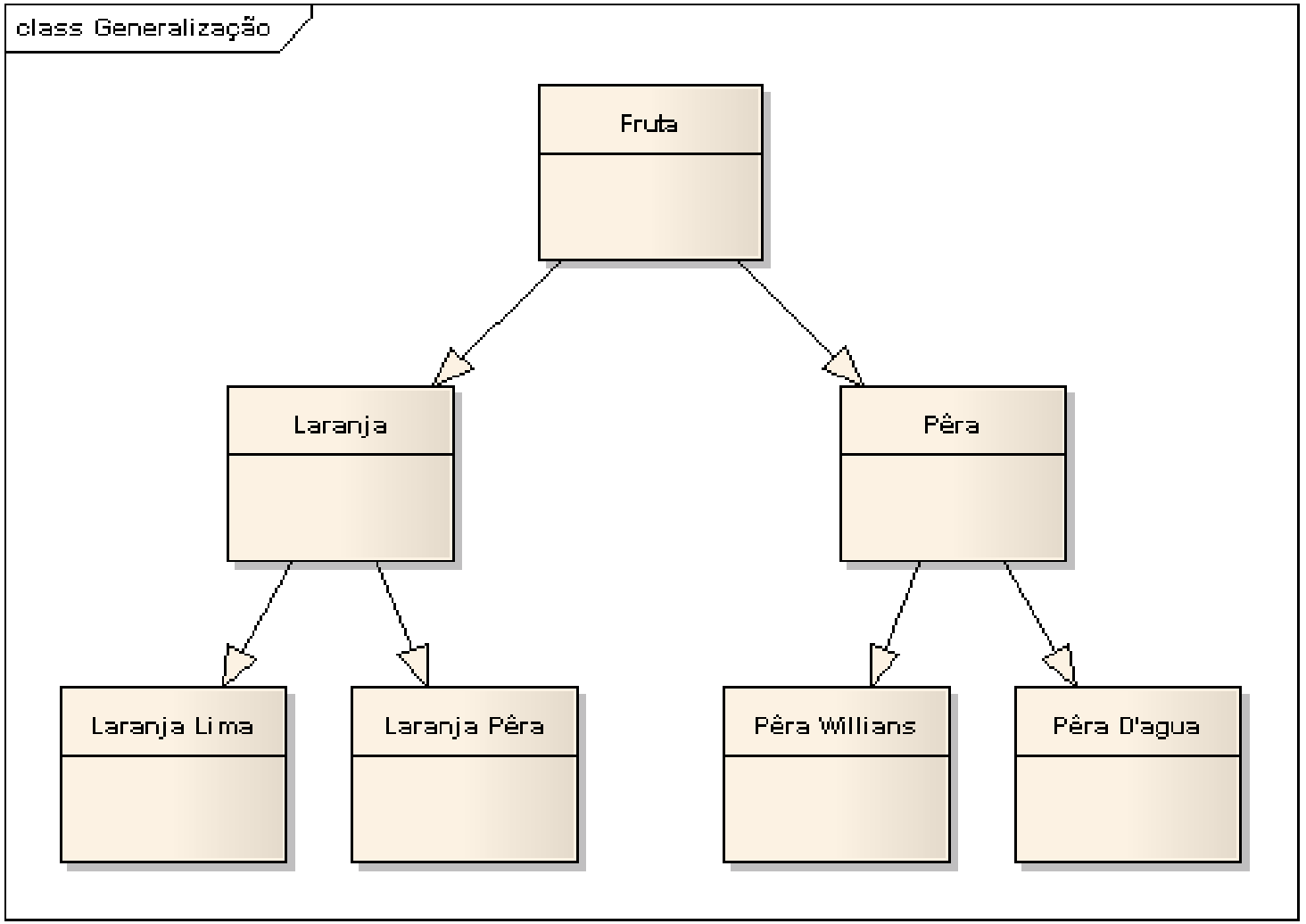
- A composição é muito semelhante a agregação, o objeto “Parte” pode pertencer somente ao objeto “Todo” e normalmente o objeto “Todo” vive e morre com suas “Partes”.
- É representada pelo símbolo de um diamante preto na classe agregadora.



Relacionamentos - Generalização

- Generalização é a capacidade de identificar as similaridades entre várias classes, com isso criamos um supertipo que encapsula todas as funcionalidades comuns as demais classes filhas.
- Podemos usar a generalização para agrupar os nossos objetos em um tipo comum.

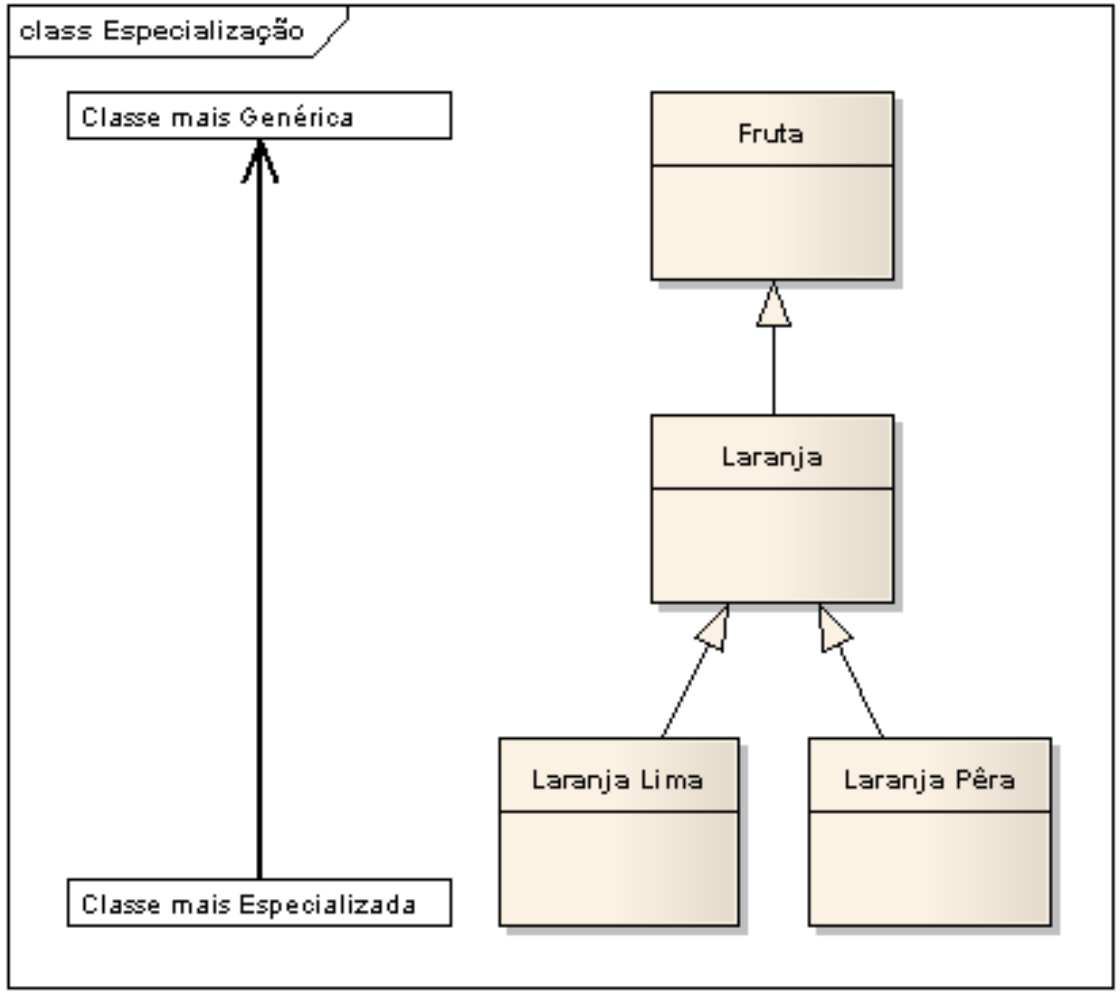
Relacionamentos - Generalização



Relacionamentos - Especialização

- A especialização cria uma classe herdada da generalização onde refina o processo definido na classe pai. Como o próprio nome diz, especializa a classe pai para um tipo específico.
- Usando a mesma imagem da generalização, podemos identificar quem é a generalização “subindo” e quem é a especialização “descendo”.

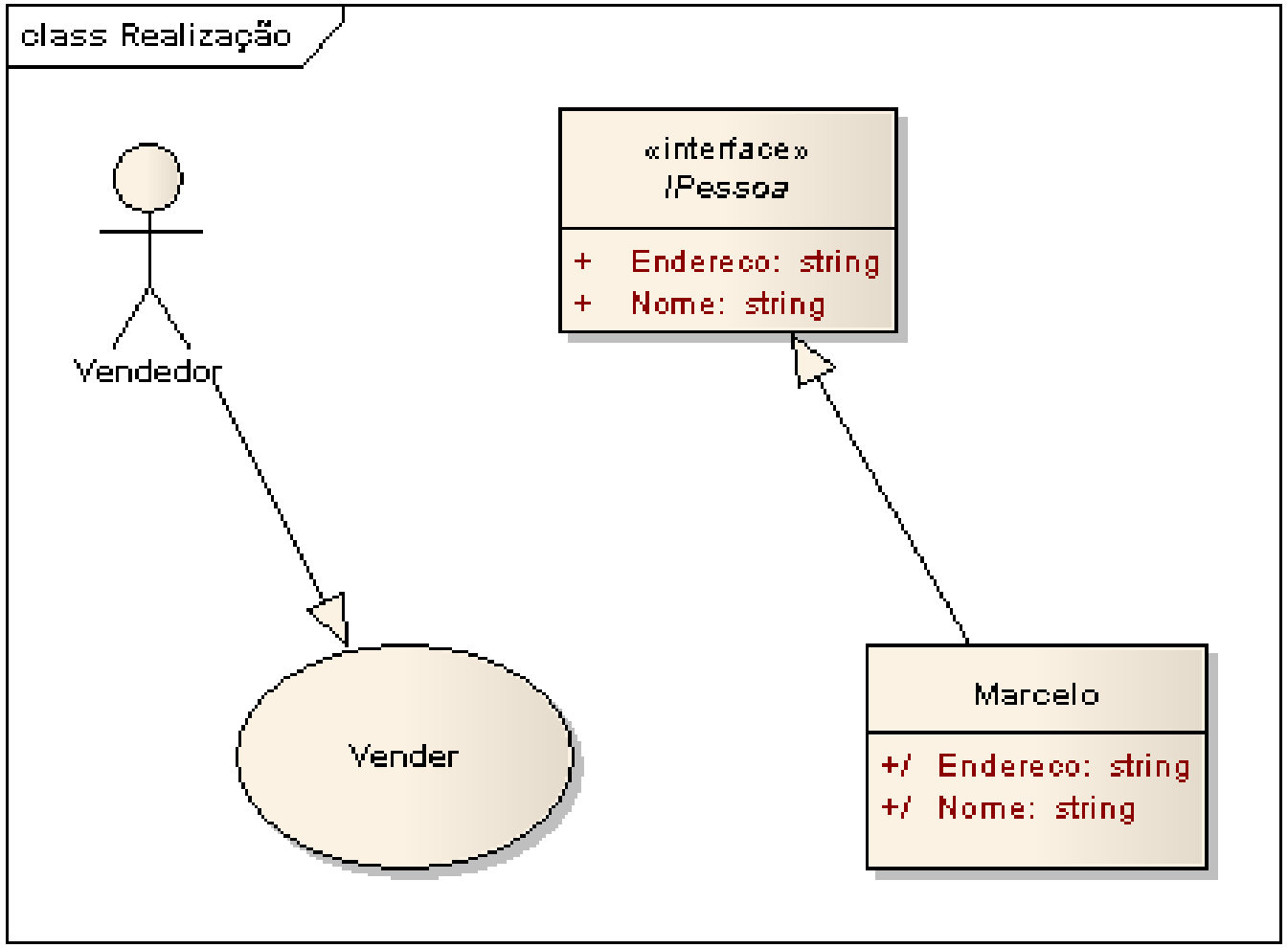
Relacionamentos - Especialização



Relacionamentos - Realização

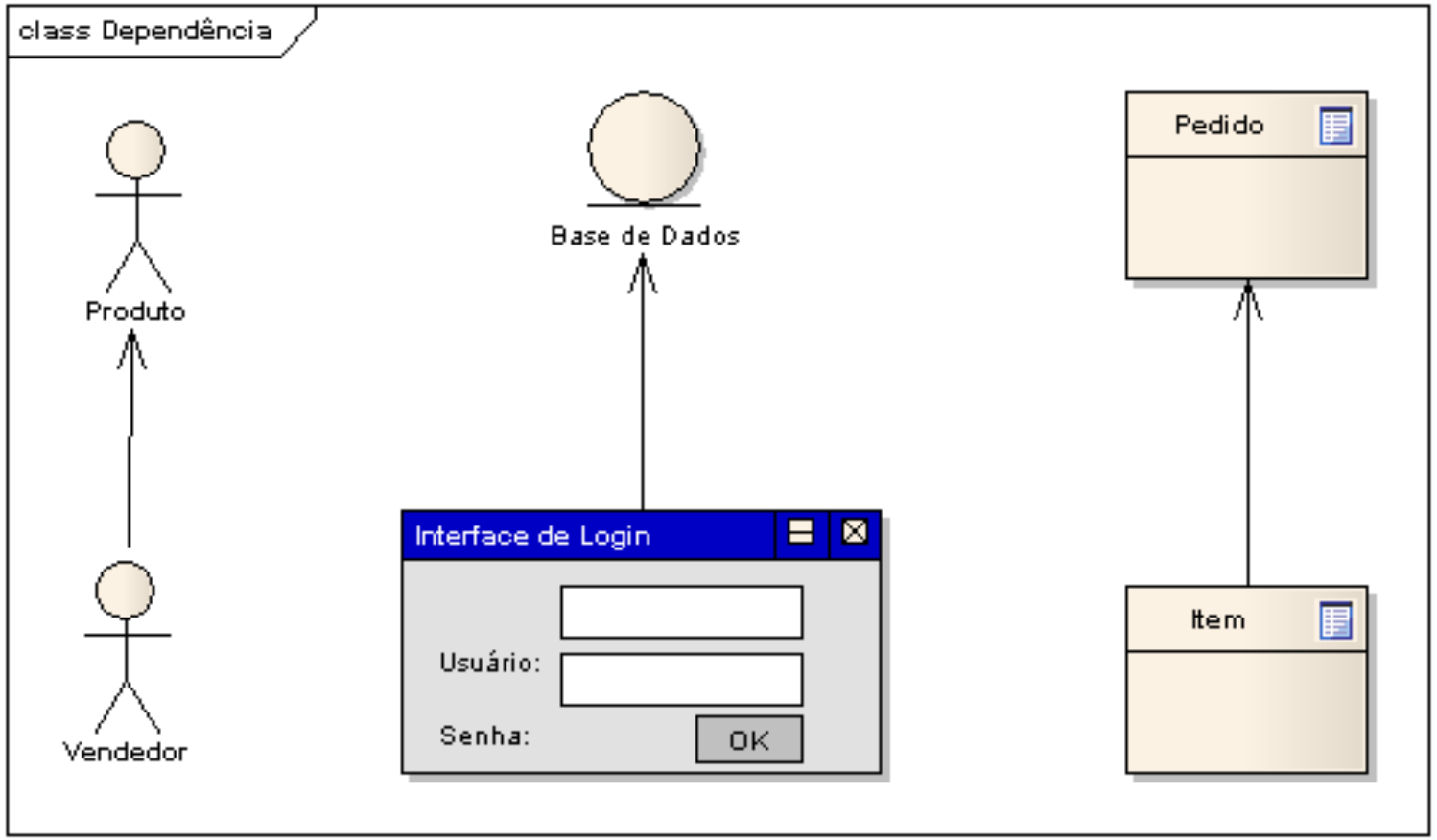
- A realização é um relacionamento entre os itens que implementa o comportamento especificado por outro.
- Um exemplo disso seria as classes abstratas e as interfaces que definem que o objeto “filho” deverá realizar algum método, propriedade no momento da herança.

Relacionamentos - Realização

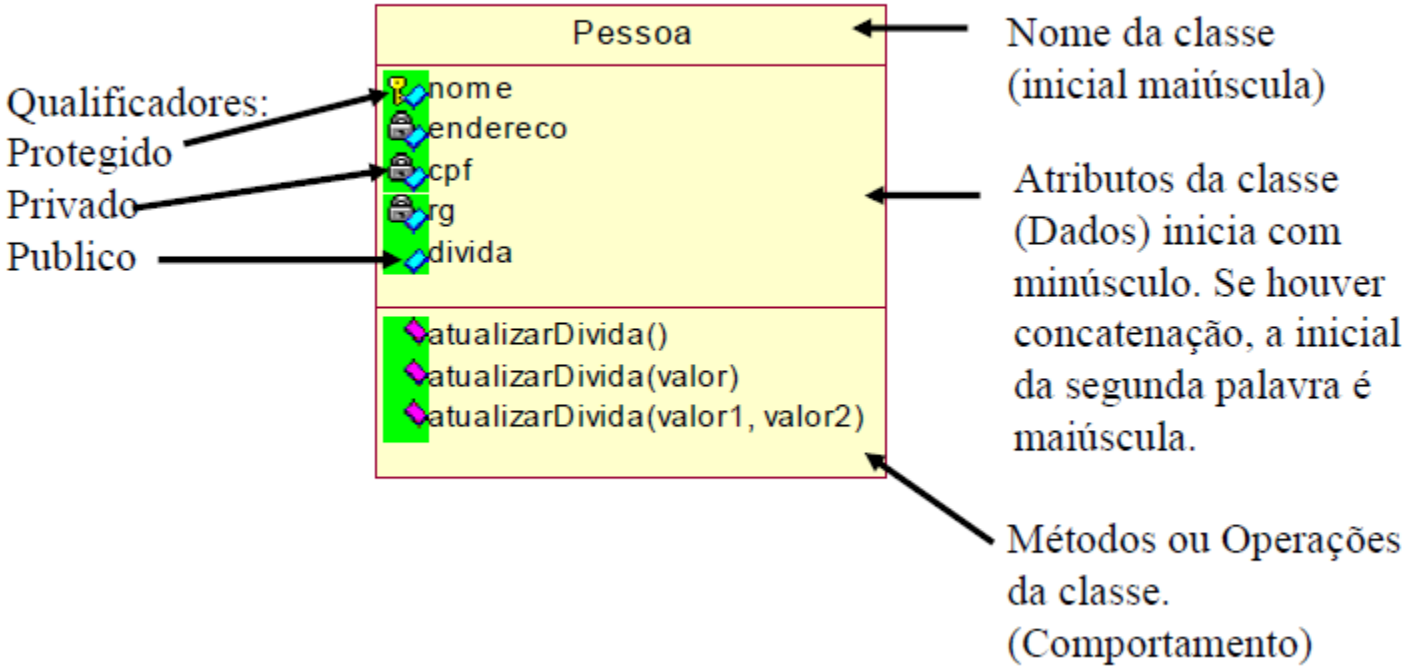


Relacionamentos - Dependência

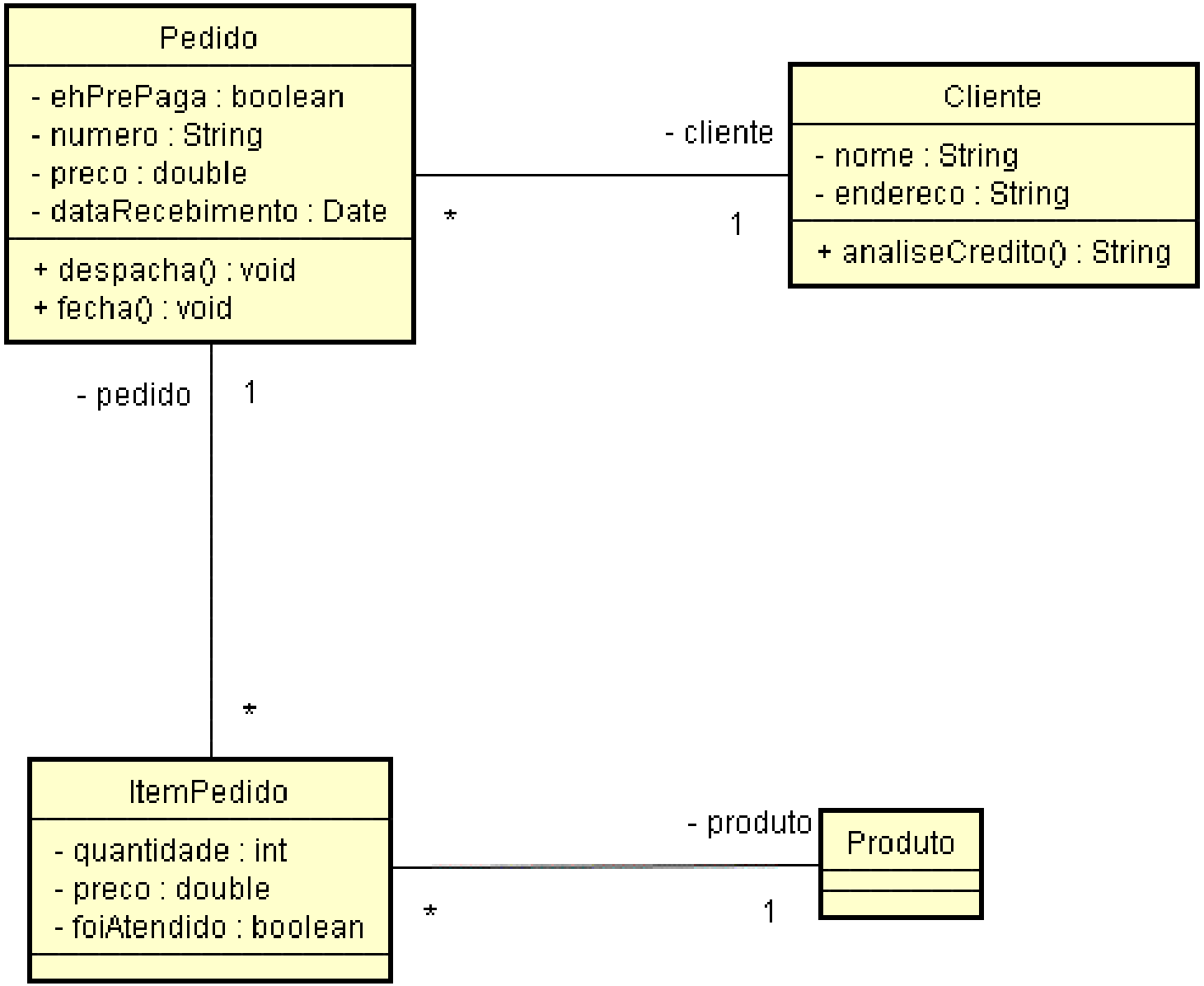
- Este é o mais simples, é um item (objeto) que depende de outro.



Notação UML


Pessoa
nome
endereco
cpf
rg
divida
atualizarDivida()
atualizarDivida(valor)
atualizarDivida(valor1, valor2)
Qualificadores:
Protegido
Privado
Publico
Nome da classe (inicial maiúscula)
Atributos da classe (Dados) inicia com minúsculo. Se houver concatenação, a inicial da segunda palavra é maiúscula.
Métodos ou Operações da classe. (Comportamento)

Exemplo


Pedido
- ehPrePaga : boolean
- numero : String
- preco : double
- dataRecebimento : Date
+ despacha() : void
+ fecha() : void
- cliente
*
1
Cliente
- nome : String
- endereco : String
+ analiseCredito() : String
- pedido
1
*
ItemPedido
- quantidade : int
- preco : double
- foiAtendido : boolean
- produto
*
1
Produto

Exemplo

- O diagrama indica que um Pedido vem de um único cliente e que um Cliente pode fazer vários pedidos ao longo do tempo.
- Cada Pedido tem vários Itens de Pedido, cada um dos quais se refere a um único Produto. A associação tem dois lados, cada lado é ligado a uma classe indicando um relacionamento entre as classes.
- A terminação de uma associação pode ser explicitamente nomeada (cliente, pedido ou produto, no exemplo).

Exemplo

- Este nome é chamado de papel (*role*). Assim, um objeto da classe Produto exerce o papel de produto para a classe ItemPedido.
- Um objeto da classe Pedido exerce o papel pedido para a classe ItemPedido e assim por diante.
- A terminação de uma associação também tem uma multiplicidade, que é uma indicação do número de objetos que podem participar do relacionamento.

Exemplo

- O * no lado do Pedido na Associação Pedido-Cliente indica que um Cliente pode ter vários Pedidos associados com ele, enquanto que o **1** do outro lado indica que um Pedido vem de um único Cliente.

Exemplo

- A partir do diagrama, posso inferir a seguinte interface para as classes em Java:

```
Public class Pedido {
    private boolean ehPrePaga;
    private String numero;
    private double preco;
    private Date dataRecebimento;

    private Cliente cliente;              // associacao com Cliente
    private List itemPedido = new ArrayList();    // associacao com ItemPedido

    public void despacha() {
        // acrescentar código
    }

    public void fecha() {
        // acrescentar código
    }
}
```

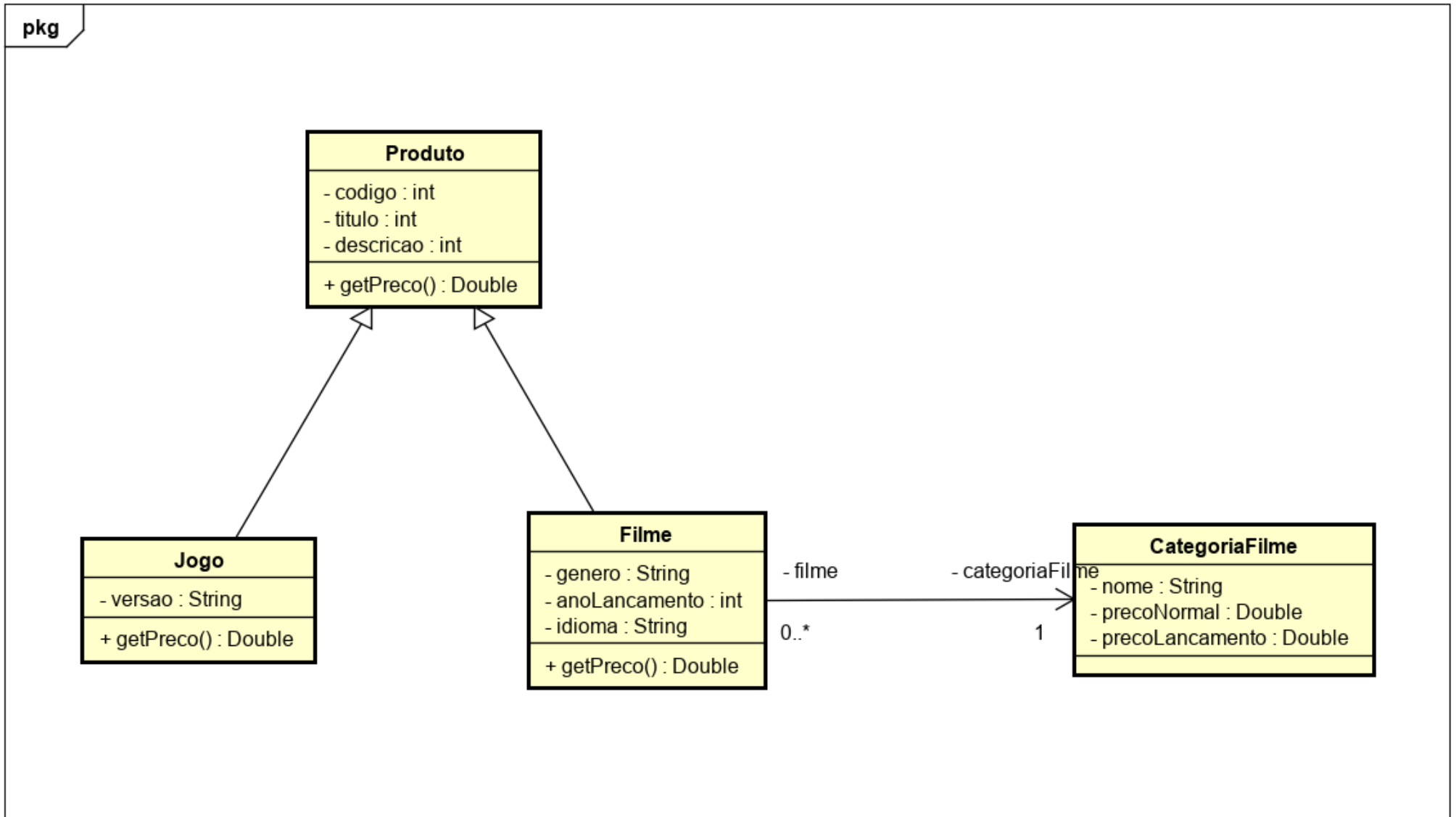
Exemplo

```
Public class ItemPedido {
    private int quantidade;
    private double preco;
    private boolean foiAtendido;

    private Pedido pedido; // associacao com Pedido
    private Produto produto; // associacao com Produto
}

Public class Cliente {
    private String nome;
    private String endereco;
    private List pedido = new ArrayList(); // associacao com Pedido

    public String analiseCredito() {
        // implementar o código depois
        return "";
    }
}
```

Exemplo

```
Public class Produto   {
        private List itemPedido = new ArrayList(); // associacao com ItemPedido
}
```

Exemplo 2 - Herança



Exemplo 2 - Herança

```
public class Produto {

        private int codigo;

        private int titulo;

        private int descricao;

        public Double getPreco() {
                return null;
        }

}
```

Exemplo 2 - Herança

```
public class Jogo extends Produto {

        private String versao;

        public Double getPreco() {
                return null;
        }

}
```

Exemplo 2 - Herança

```
public class Filme extends Produto {

        private String genero;

        private int anoLancamento;

        private String idioma;

        private CategoriaFilme categoriaFilme;

        public Double getPreco() {
                return null;
        }

}
```

Exemplo 2 - Herança

```
public class CategoriaFilme {

        private String nome;

        private Double precoNormal;

        private Double precoLancamento;

}
```

Exercício de Fixação

1) Dadas as características, comportamentos e classes organize-as em um diagrama de classes

- características:
  - nome, endereço, telefone, área de conhecimento, registro no MEC, nota, carga horária, titulação, frequência
- comportamentos:
  - matricular, contratar, demitir, pagar, coordenar, inscrever_disciplina, alocar_disciplina, cadastrar_nota
- classes:
  - professor – professor titular – professor adjunto – secretária – diretor – aluno de graduação – aluno de pós - disciplina - faculdade – curso.

Referências

- Craig Larman, 2007, **“Utilizando UML e Padrões”**, 3ª ed.
- SOMMERVILLE, Ian, **Engenharia de Software**, 8ª Edição, São Paulo, Editora Pearson Prentice Hall, 2007.
- PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 7ª Edição. Porto Alegre: AMGH, 2011. 780 p.
- BOOCH, G **UML: Guia do Usuário**. Rio de Janeiro: Campus, 2005.

Dúvidas

José Osvano da Silva
osvano@gmail.com