



Curso UML

Diagramas de Clases

Noção de Classes

- ◆ Uma **classe** é um descritor de um conjunto de objetos que partilham as mesmas propriedades (atributos, operações, relações e semântica).
- ◆ Classes são usadas para capturar o **vocabulário** de um sistema.
- ◆ Classes são abstrações de elementos do domínio do problema, como “Cliente”, “Banco”, “Conta”.





Exemplos de Classes

◆ Exemplos de classes:

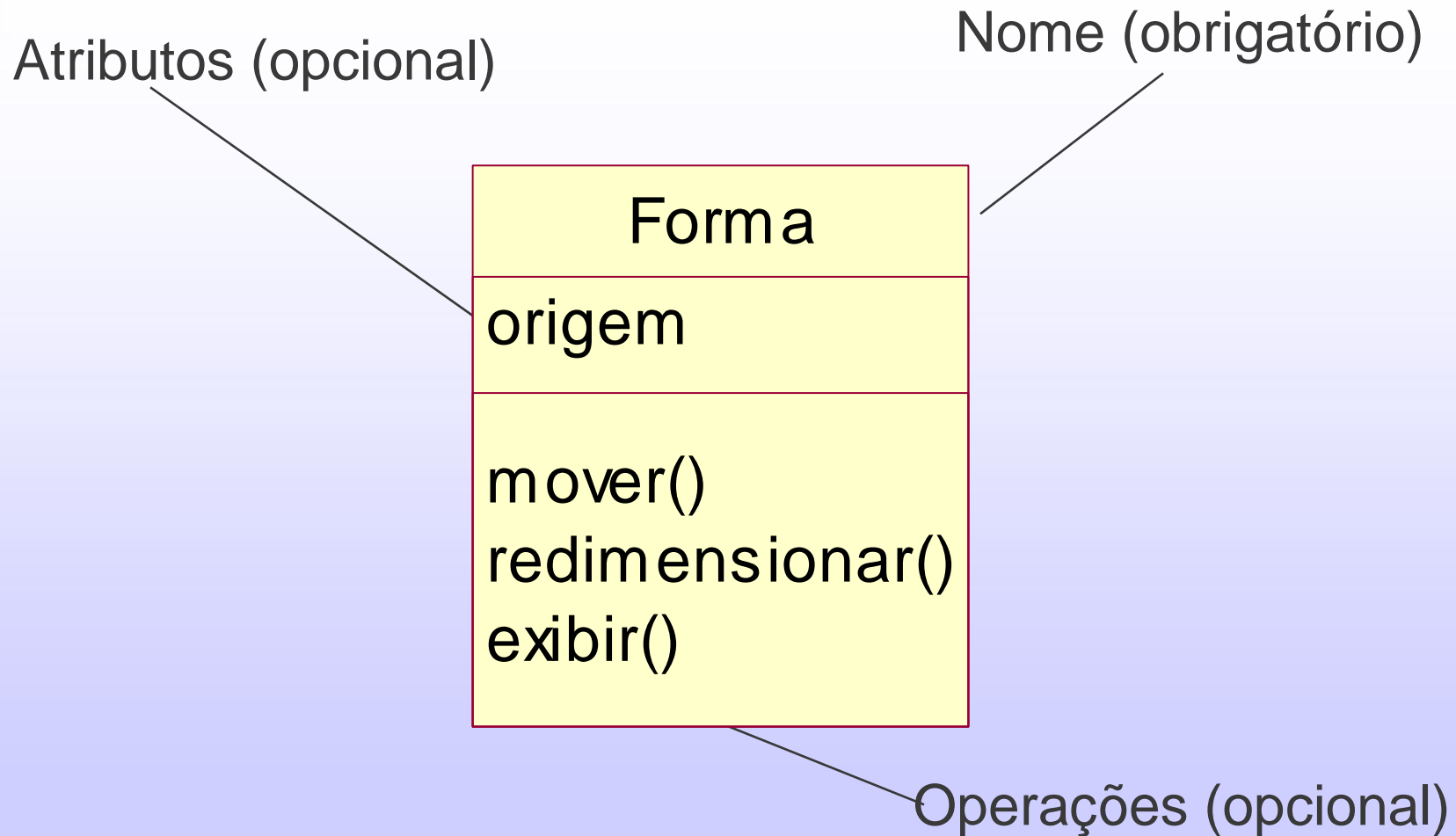
- Coisas concretas: Pessoa, Turma, Carro, Imóvel, Factura, Livro
- Papéis que coisas concretas assumem: Aluno, Professor, Piloto
- Eventos: Curso, Aula, Acidente
- Tipos de dados: Data, Intervalo de Tempo, Número Complexo, Vector



Noção de Classe

- ◆ Para se precisar o significado pretendido para uma classe, deve-se explicar o que é (e não é ...) uma instância da classe:
 - Exemplo: “Um aluno é uma pessoa que está inscrita num curso ministrado numa escola. Uma pessoa que esteve no passado inscrita num curso, mas não está presentemente inscrita em nenhum curso, não é um aluno.”

Classe - Notação Básica



Nome da Classe

- ◆ Toda classe deve ter um nome.
- ◆ Um nome pode ser simples (apenas o nome), ou pode ser precedido pelo nome do pacote em que a classe está contida (nome de caminho).

Conta

Banco

Cliente

Exceções::ClienteNãoCadastrado

Atributos

- ◆ “Um atributo é um propriedade com nome de uma classe que descreve uma gama de valores que as instâncias da propriedade podem tomar”.
- ◆ Em um dado momento, um objeto de uma classe conterá valores para todos os atributos descritos na sua classe.
- ◆ O nome de um atributo tem de ser único dentro da classe.




Atributos



- ◆ Um atributo de uma classe tem um **valor** (possivelmente diferente) em cada instância (objeto) da classe.
- ◆ O valor de um atributo de um objeto pode mudar ao longo do tempo.

Atributos

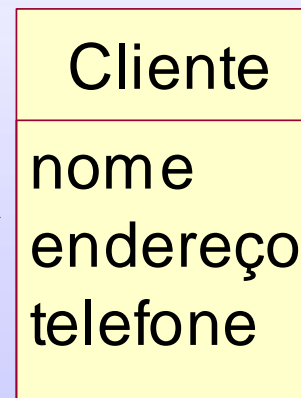
- 
- ◆ O estado de um objeto num dado momento é dado (em parte) pelos valores dos seus atributos (outra parte é dada pelas ligações que tem com outros objetos)
 - ◆ Exemplos:
 - atributos de Pessoa: nome, data de nascimento, peso, etc.
 - atributos de Carro: matrícula, ano, peso, etc.



Atributos - Notação Básica

- ◆ Atributos podem ser identificados apenas com nomes.

Atributos





Atributos - Notação

- ◆ Atributos podem ter seus tipos (ou classes) especificados e terem valores padrão definidos

Pessoa
nome : string altura : real peso : real = 70 nascimento : date



Operações / Métodos

- ◆ “Uma operação é a implementação de um serviço que pode ser solicitado a qualquer objeto da classe”.
- ◆ Um classe pode ter qualquer número de operações, inclusive nenhuma.
- ◆ Operações são o meio de alterar os valores dos atributos.



Operações / Métodos

- ◆ Como para os atributos, você pode especificar uma operação apenas com seu nome.
- ◆ Pode-se também especificar a **assinatura** da operação: seus parâmetros, o tipo desses parâmetros e o tipo de retorno.

Operações - Notação



Retângulo
mover() aumentar() diminuir()

Retângulo
mover(x : float, y : float) aumentar(float : altura, float largura) diminuir(float : altura, float largura)



Visibilidade

- ◆ Pode-se usar marcações de acesso para especificar o tipo de acesso permitido aos atributos e operações.
- + público: todos os classificadores podem usar
- # protegido: qualquer descendente do classificador poderá usar
- privado: somente o próprio classificador poderá usar



Visibilidade

Toolbar
currentSelection: Tool # toolCount: Integer
+ getTool(i: Integer): Tool + addTool(t: Tool) + removeTool(i: Integer) - compact()

Relacionamentos

◆ Ligam as classes/objetos entre si criando relações lógicas entre estas entidades. Os relacionamentos podem ser dos seguintes tipos:

- Associações;
- Agregação;
- Composição;
- Dependências;
- Generalizações e;
- Realização.





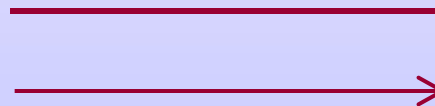
Associações



Associações

- ◆ São relacionamentos estruturais entre instâncias.
- ◆ Descreve um conjunto de ligações, em que as ligações são conexões entre objetos.

Associação
Sem/Com navegação

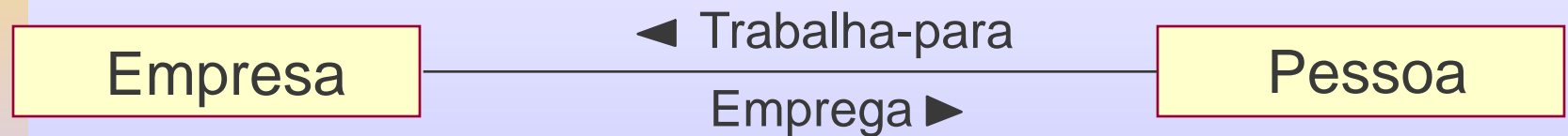
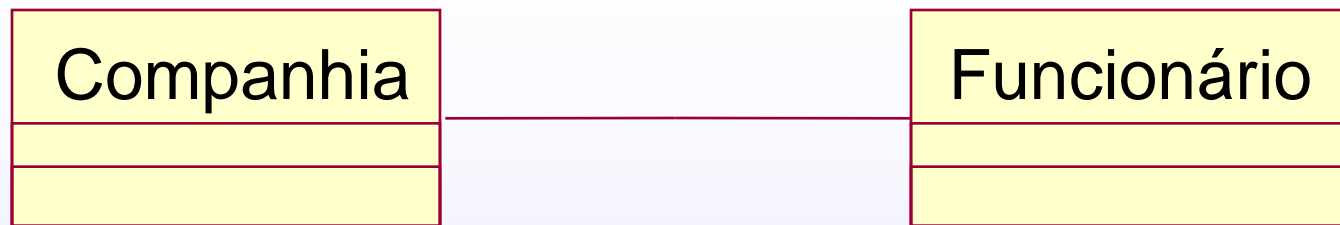




Nome da Associação

- ◆ A indicação do nome é opcional.
- ◆ O nome é indicado no meio da linha que une as classes participantes.
- ◆ Pode-se indicar o sentido em que se lê o nome da associação.

Nome da Associação

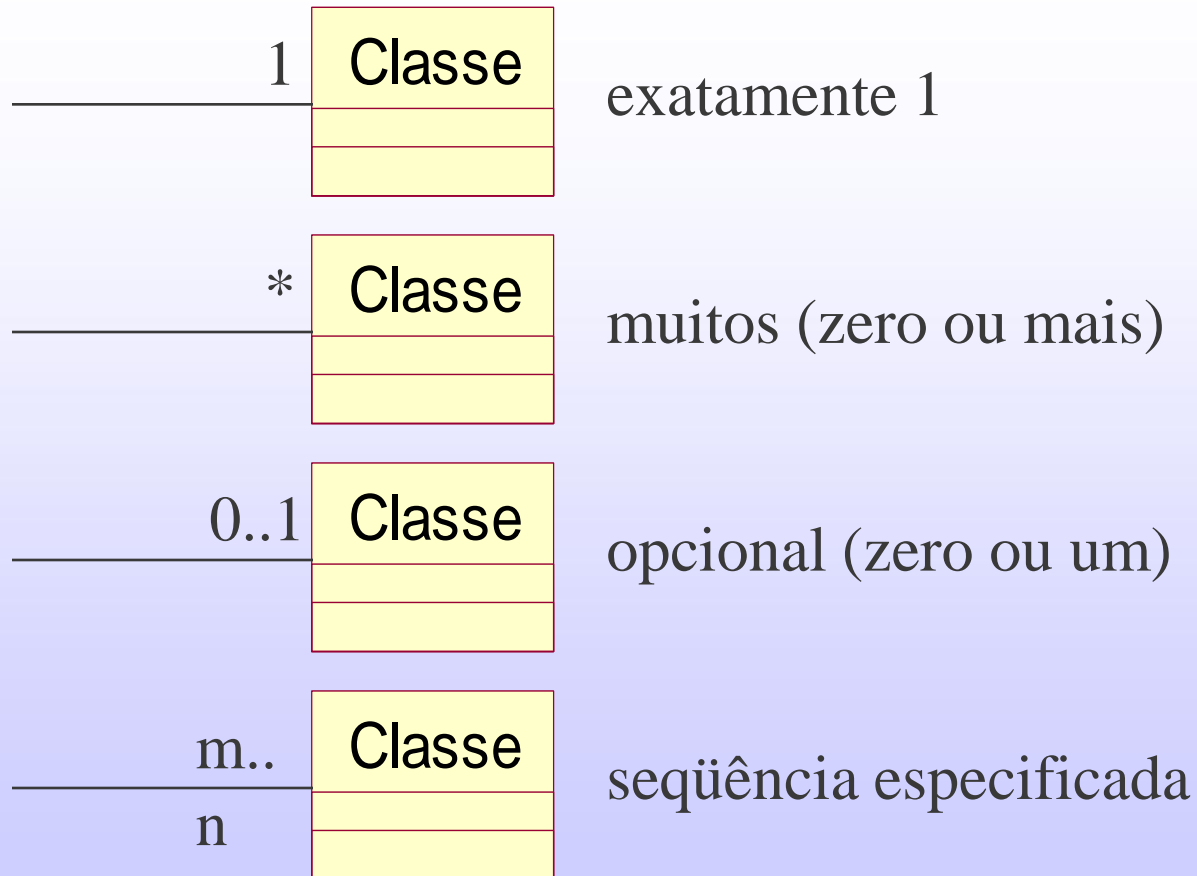




Multiplicidade

- ◆ Em muitas situações é importante determinar a quantidade de objetos que podem ser conectados pela instância de uma associação.
- ◆ É a cardinalidade de uma associação.

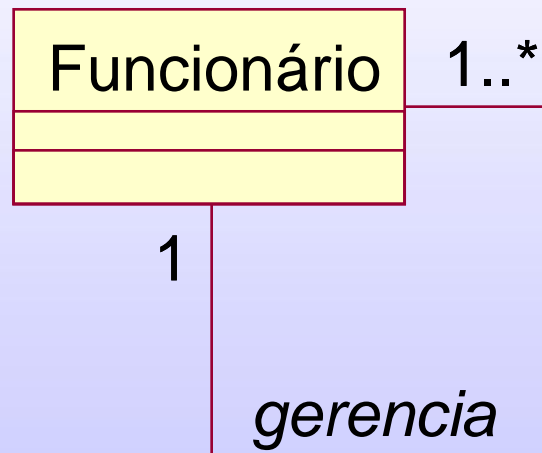
Multiplicidade



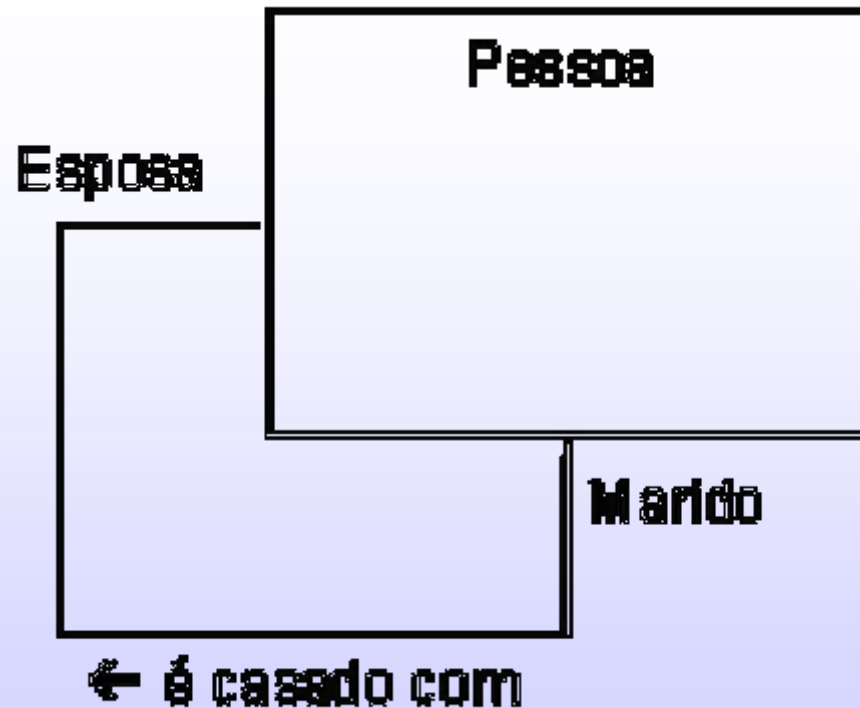


Associações Unárias ou Auto-associações ou Recursivas

- ◆ Quando há um relacionamento de uma classe para consigo própria.



Associações Unárias ou Auto-associações ou Recursivas





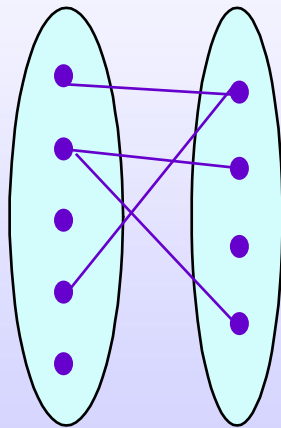
Associações Binárias

- ◆ Quando há duas classes envolvidas na associação de forma direta de uma para a outra.



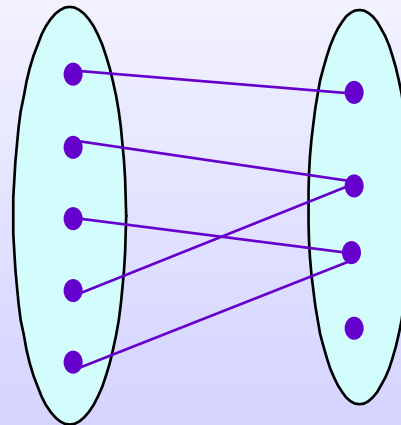
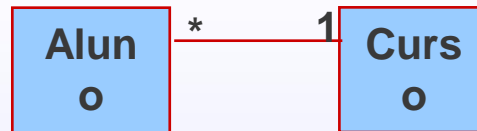
Multiplicidade de Associações Binárias

Muitos-para-Muitos

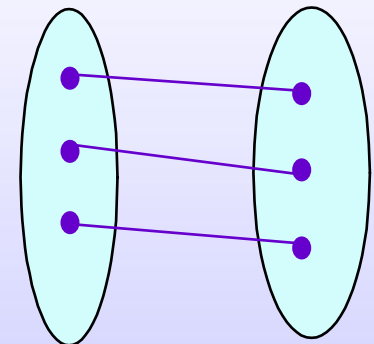
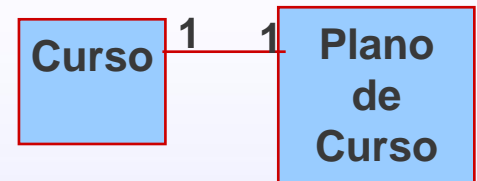


(sem restrições)

Muitos-para-1



1-para-1

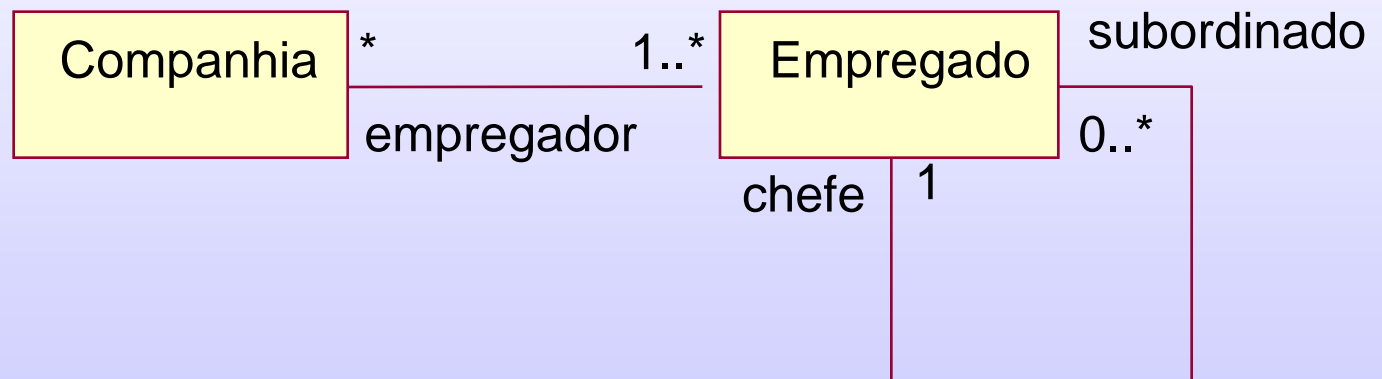




Papéis da Associação

- ◆ Papéis: um dos lados da associação.
- ◆ Nomes de papéis são necessários para associação entre dois objetos da mesma classe.

Papéis da Associação





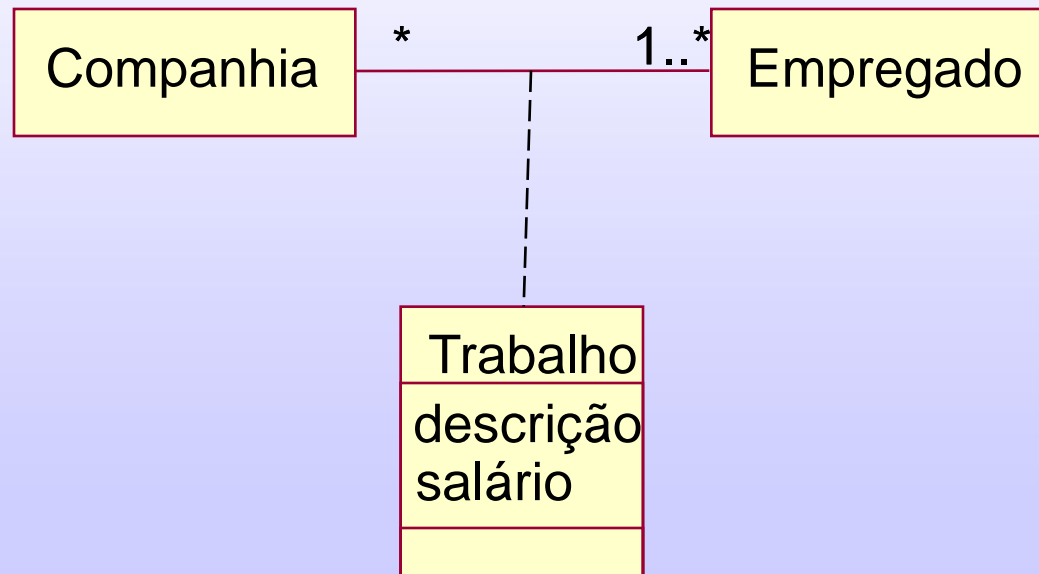
Associação - Navegabilidade

- ◆ Em geral a navegação entre as classes de uma associação é bi-direcional.
- ◆ Porém é possível limitá-la a apenas uma direção (unidirecional)



Associação com Atributos

- ◆ Modela as propriedades associadas com uma associação.
- ◆ As propriedades devem ser representadas por uma classe.

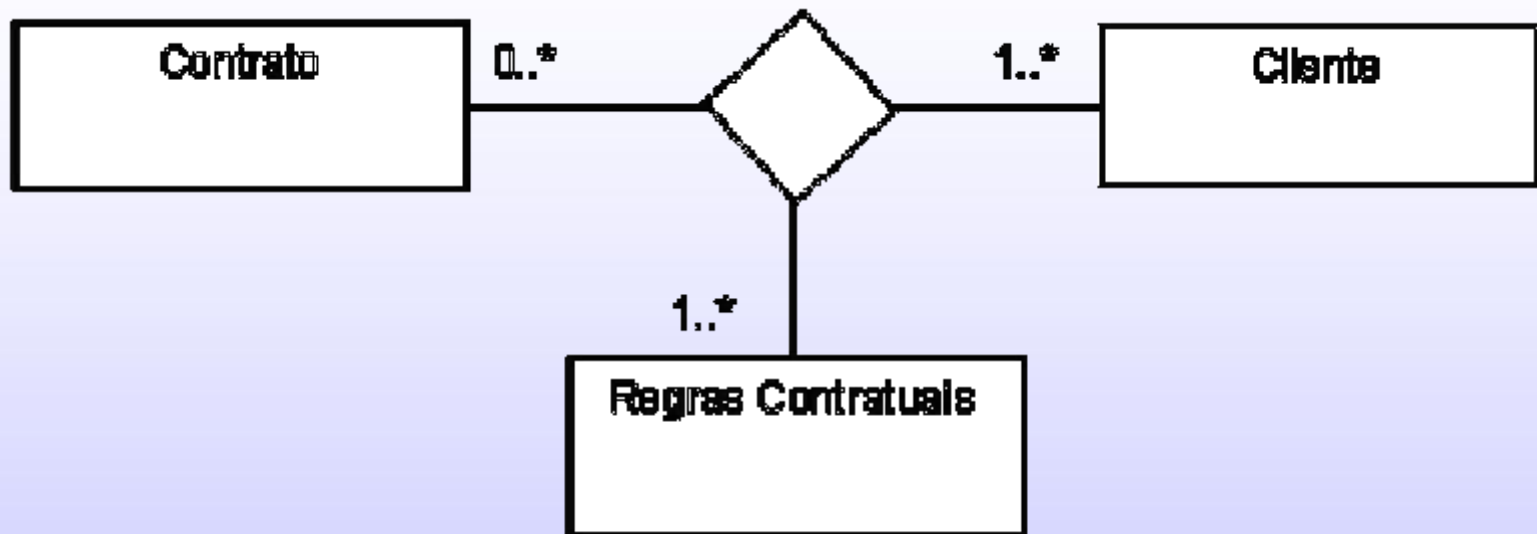




Associação Ternária

- ◆ Mais de duas classes podem ser associadas entre si, a associação ternária associa três classes.
- ◆ Ela é mostrada como um grade losango (diamante).

Associação Ternária





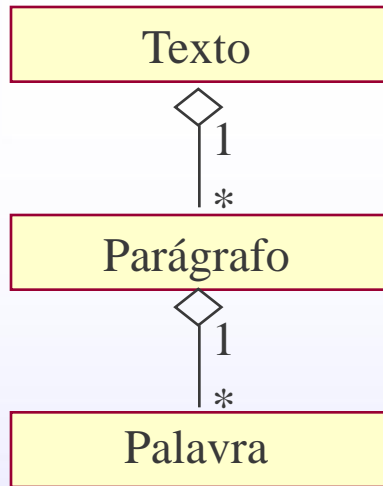
Agregação



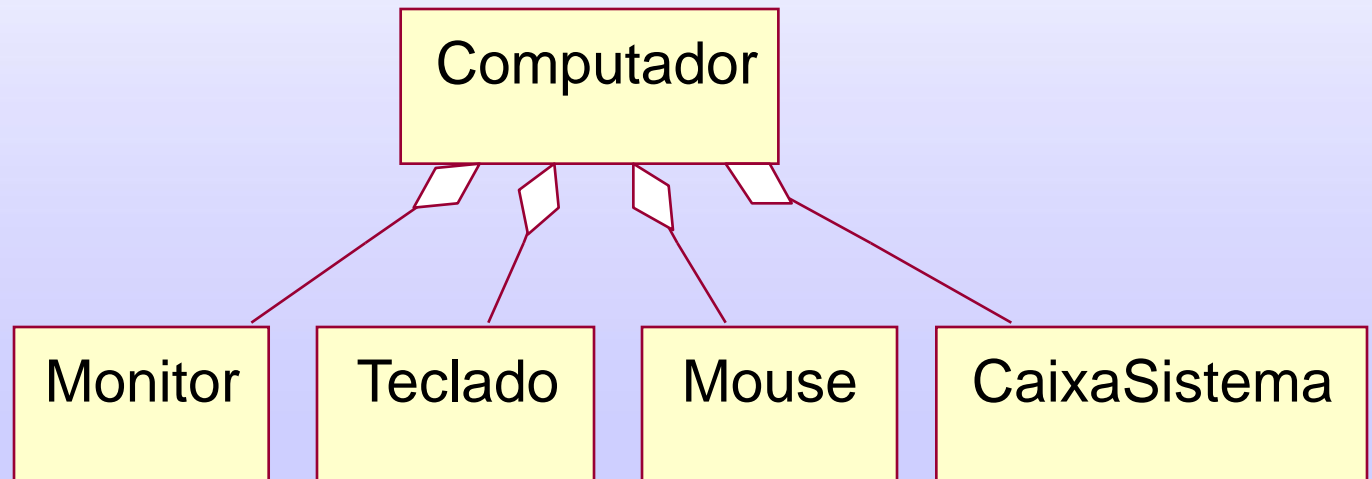
Agregação

- ◆ um tipo especial de associação entre o todo e suas partes, no qual o todo é composto de partes
- ◆ Não impõe que a vida das “Partes” esteja relacionado com a vida do “Todo”.
- ◆ As palavras chaves usadas para identificar uma agregação são: "consiste em", "contém", "é parte de".

Agregação - Exemplos



- Um texto **contém** *0 ou mais* parágrafos
- Um parágrafo **faz parte de** *um* texto
- Um parágrafo **contém** *0 ou mais* palavras
- Uma palavra **faz parte de** *um* parágrafo





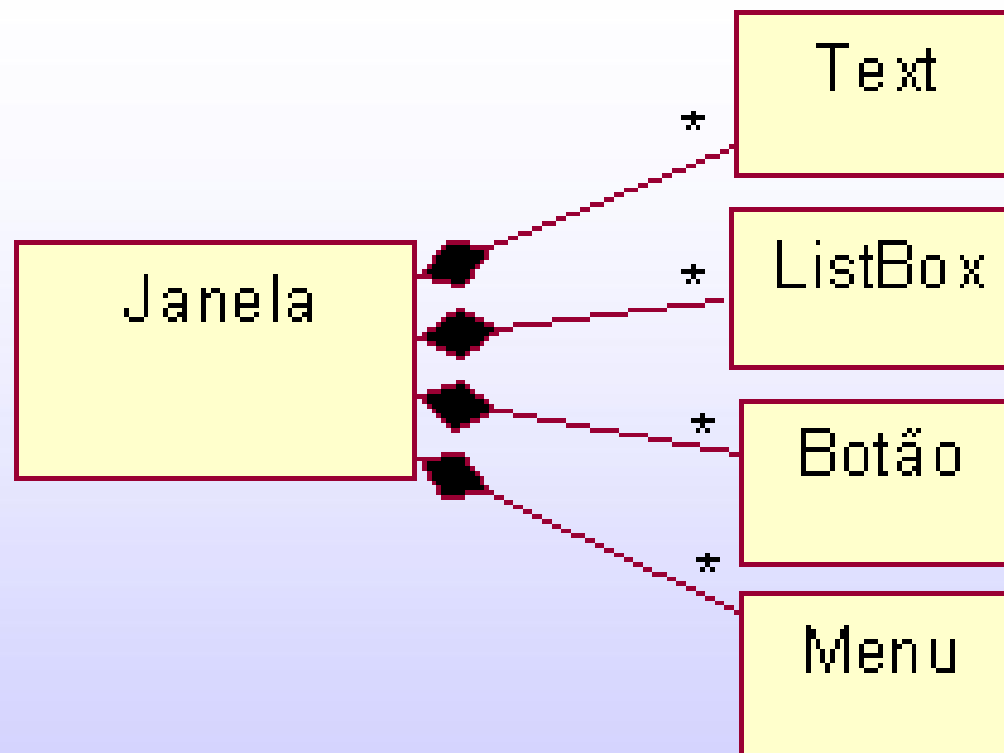
Composição



Composição

- ◆ Uma forma mais forte de agregação.
- ◆ Há uma coincidência da vida das partes.
- ◆ Uma vez criada a parte ela irá viver e morrer com ele.
- ◆ O “Todo” é responsável pelo gerenciamento da criação e destruição das partes.

Composição





Dependências



Dependências

- ◆ Dependências são relações de **uso**.
- ◆ Uma dependência indica que mudanças em um elemento (o “servidor”) podem afetar outro elemento (o “cliente”).
- ◆ Uma dependência entre classes indica que os objetos de uma classe **usam** serviços dos objetos de outra classe.

Dependências





Generalizações (Herança)



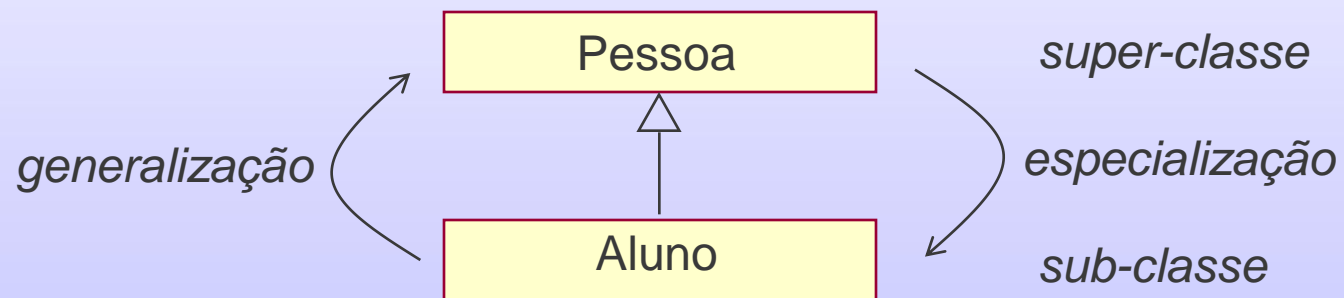
Generalizações

- ◆ Relação semântica “is a” (“é um” / “é uma”).
- ◆ Relacionamento entre um elemento mais geral (chamado de **superclasse** ou pai) e um mais específico (chamado de **subclasse** ou filho).

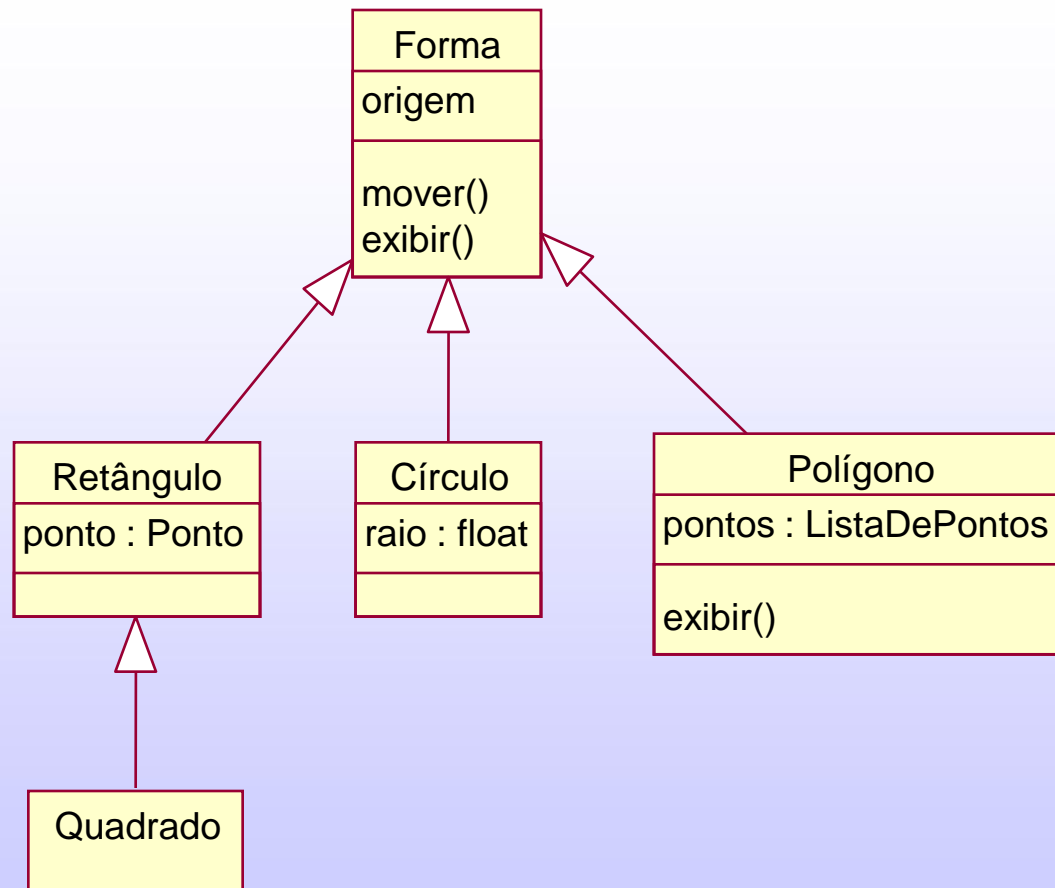


Generalizações

- ◆ A sub-classe herda as propriedades (atributos, operações e relações) da super-classe, podendo acrescentar outras.

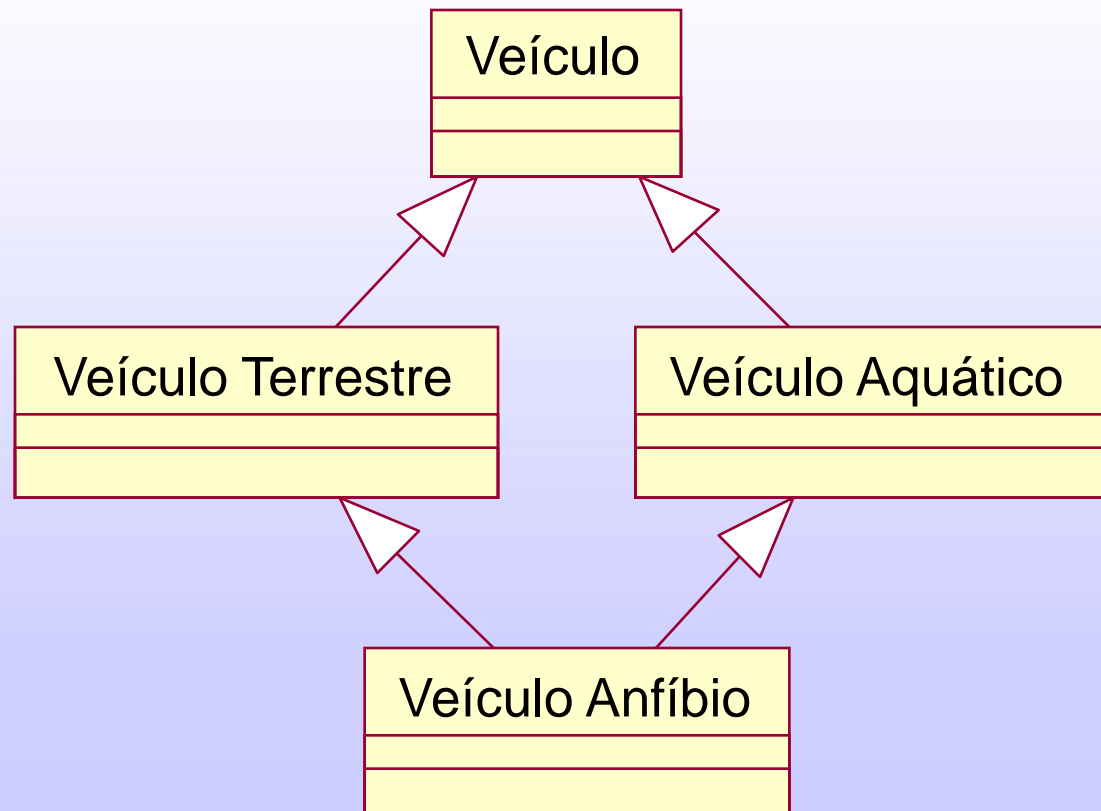


Generalizações - Exemplo



Herança Múltipla

- ◆ Ocorrem múltiplas superclasses para uma mesma subclasse.

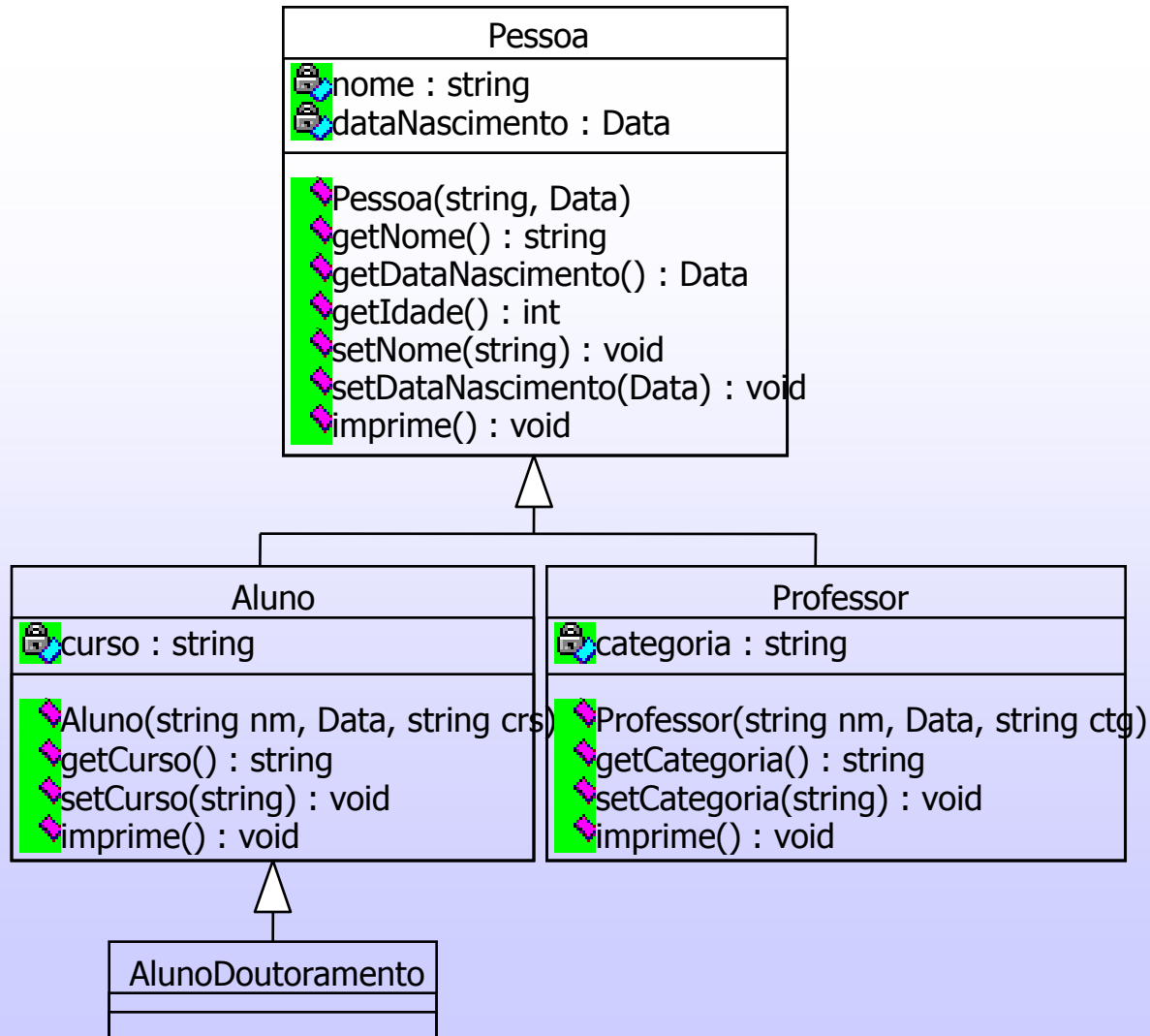




Herança

- ◆ Em cada classe da hierarquia colocam-se as propriedades que são comuns a todas as suas subclasses.
- ◆ Evita-se redundância, promove-se reutilização!

Herança - Outro Exemplo





Realização



Realização

- ◆ Relacionamento entre uma interface e o elemento que a implementa.



Interface

- ◆ É uma coleção de operações que especificam serviços de uma classe ou componente.
- ◆ Diferentemente das classes, as interfaces não especificam nenhuma estrutura.
- ◆ Interfaces **não podem conter atributos.**



Diagrama de Classes



Diagrama de Classes

- ◆ Os diagramas de classes são os principais diagramas estruturais da UML
- ◆ Diagramas de classe mostram classes, interfaces e seus relacionamentos
- ◆ As classes especificam a estrutura e o comportamento dos objetos, que são instâncias de classes

Diagrama de Classes

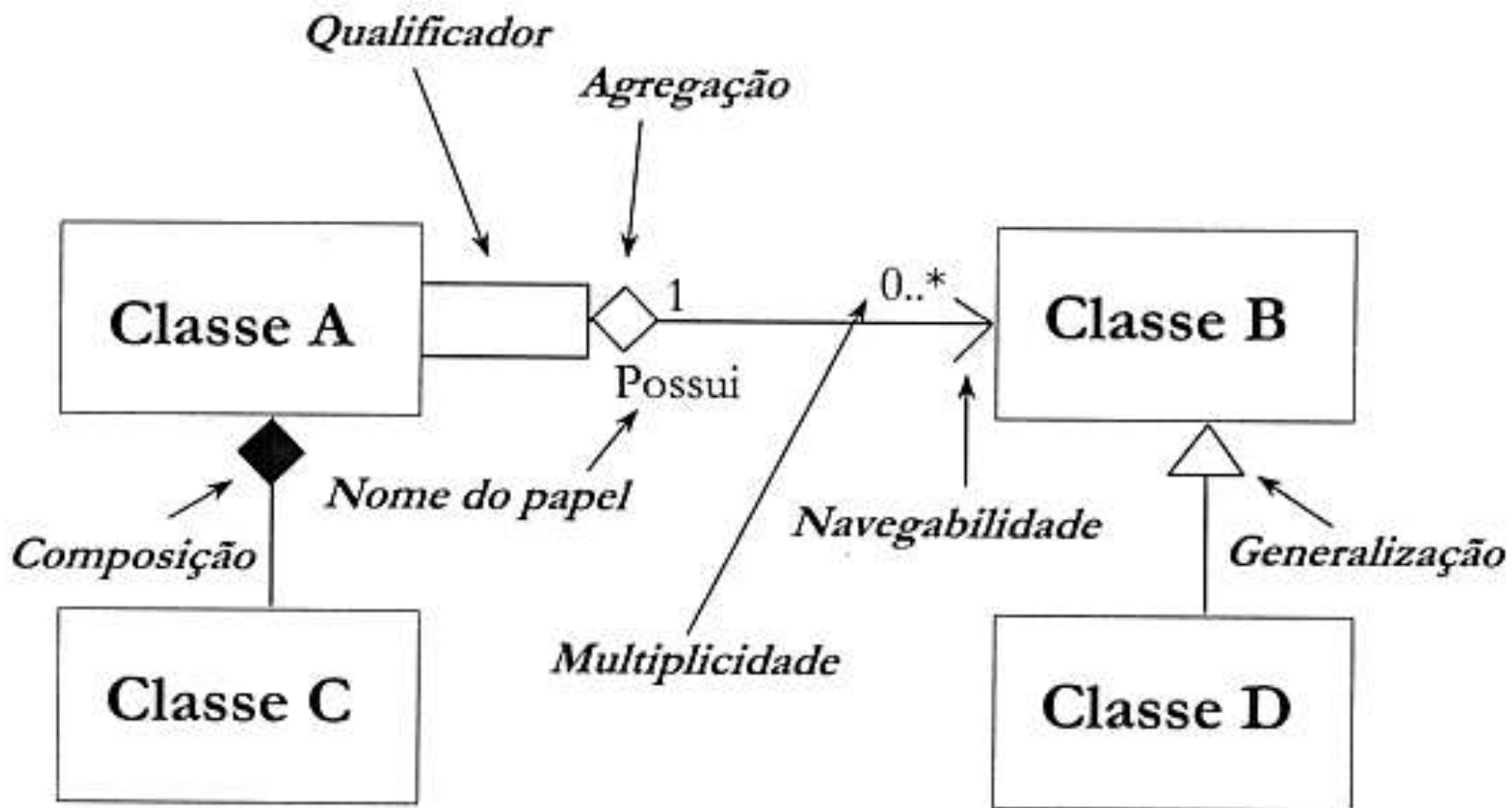
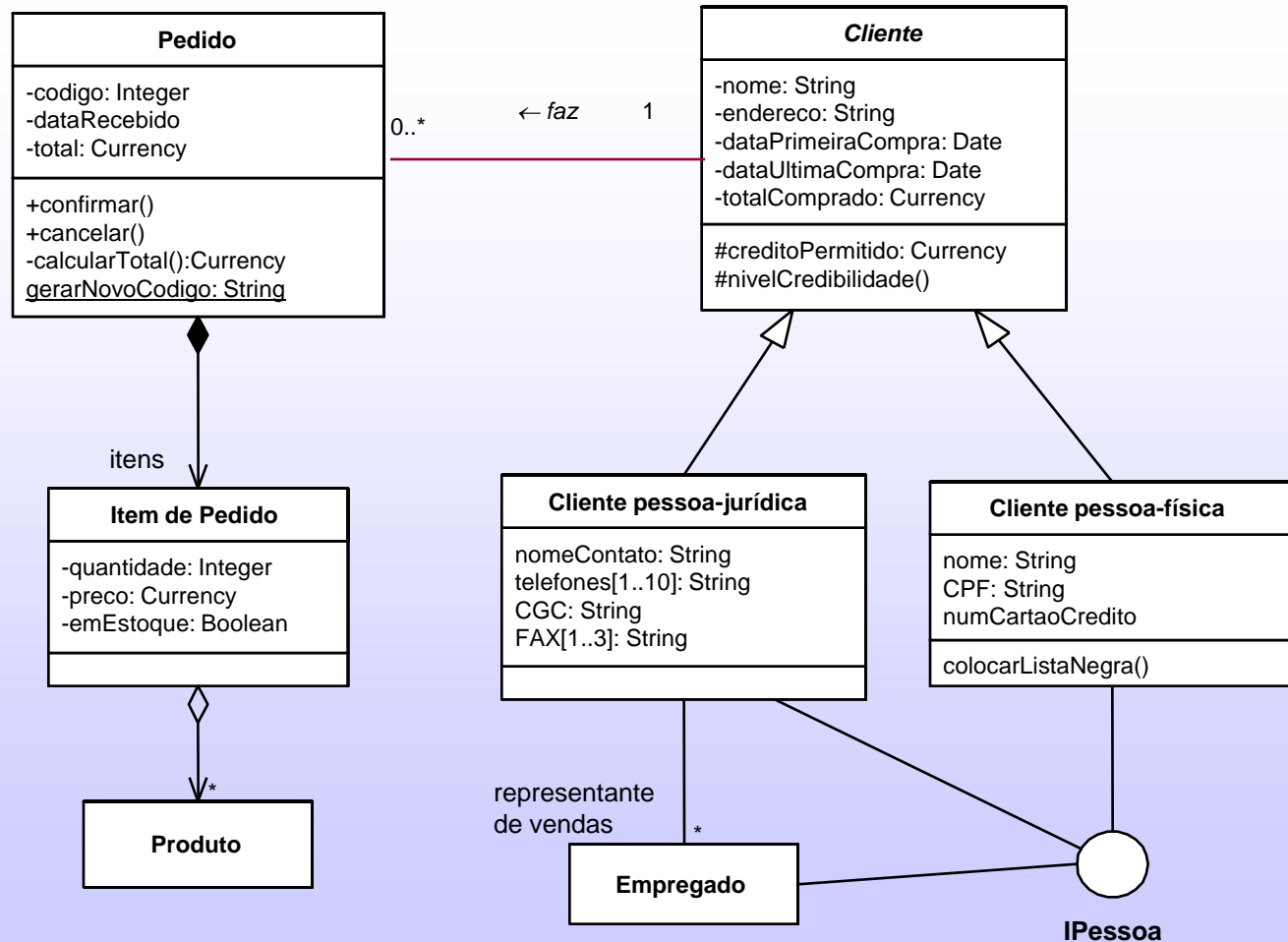


Diagrama de Classes - Exemplo





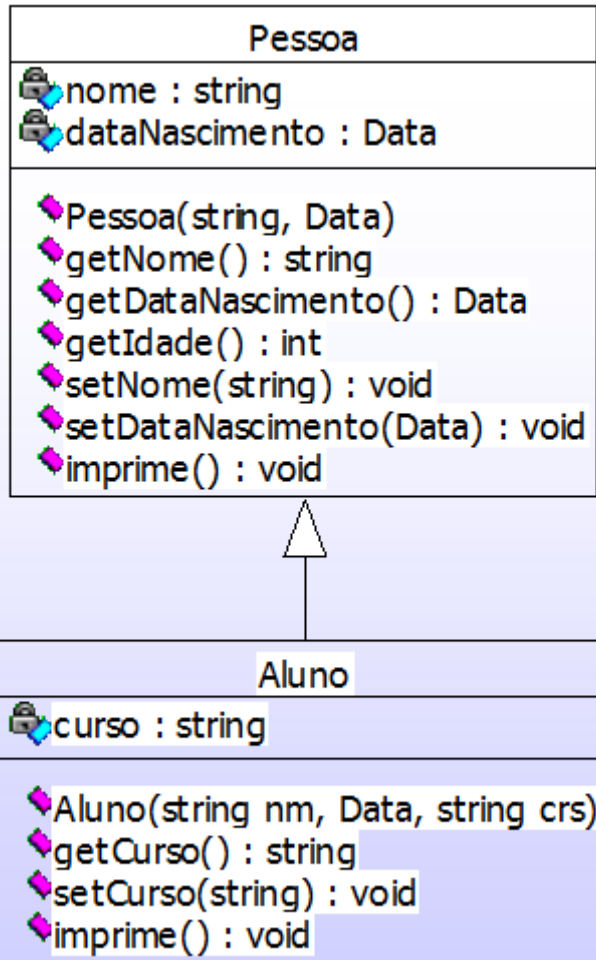
Mapeamento para Linguagens de Programação



Mapeamento para LP

- ◆ A UML é independente de qualquer linguagem de programação.
- ◆ Existem ferramentas que oferecem suporte para o mapeamento das classes criadas por meio da UML para Linguagens OO.

Mapeamento para C++



```
class Pessoa {
private:
    string nome;
    Data dataNascimento;
public:
    Pessoa(string, Data);
    string getNome();
    Data getDataNascimento();
    int getIdade();
    void setNome(string);
    void setDataNascimento(Data);
    void imprime();
};
```

```
class Aluno : public Pessoa {
private:
    string curso;
public:
    Aluno(string nm, Data, string crs);
    string getCurso();
    void setCurso(string);
    void imprime();
};
```



Pacotes



Pacotes

- ◆ Os pacotes são unidades de agrupamentos de classes.
- ◆ Classes são agrupadas em pacotes principalmente por:
 - questão de organização;
 - performance de aplicação;
 - solução de problemas de escopos de variáveis;
 - organização por estereótipos;
 - verticalização de uma entidade para comercializá-la como um componente etc.
- ◆ O único tipo de relacionamento entre pacotes é o de dependência.

Pacotes - Exemplo

