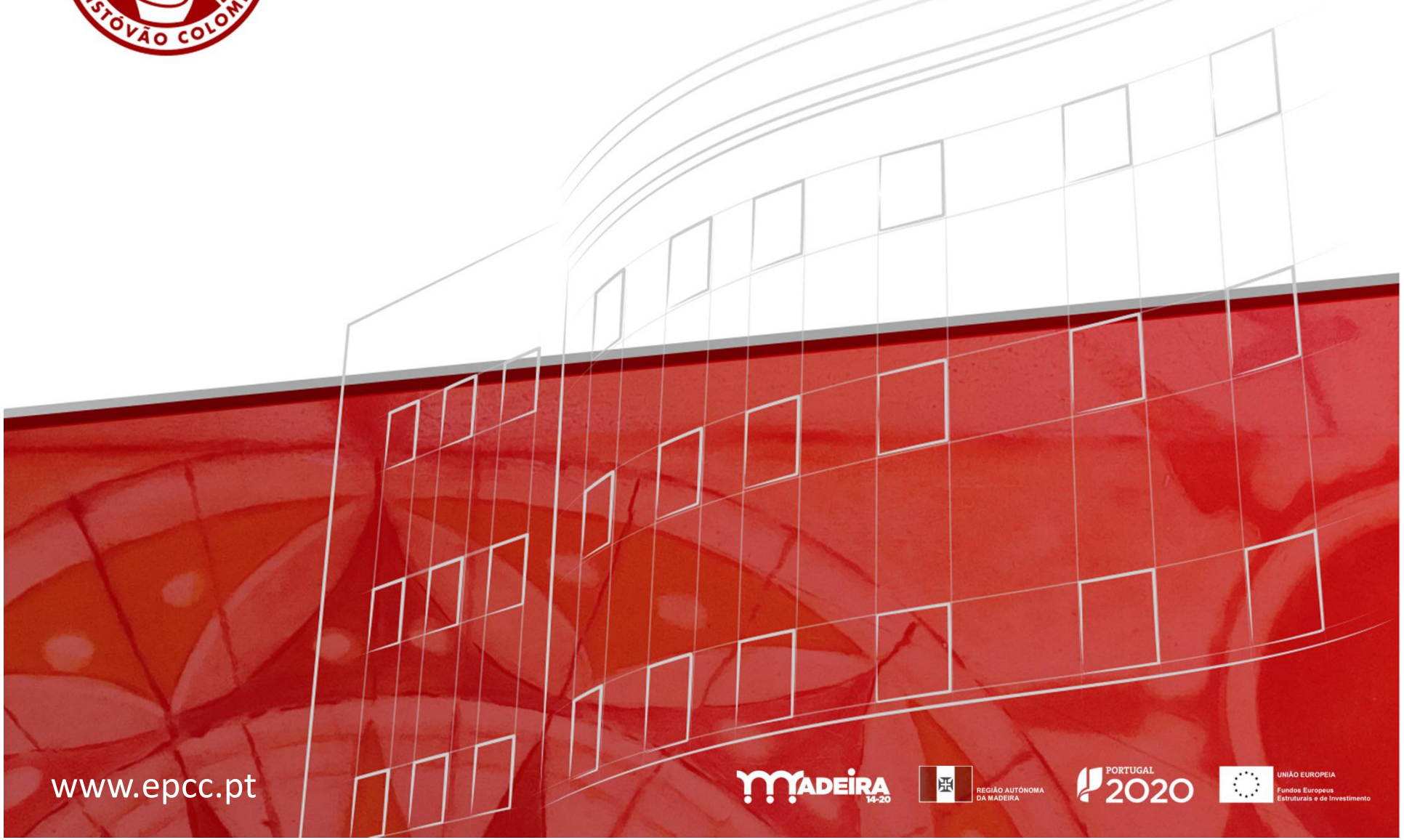
ESCOLA PROFISSIONAL
CRISTÓVÃO COLOMBO



www.epcc.pt

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

PORTUGAL 2020

UNIÃO EUROPEIA
Fundos Europeus Estruturais e de Investimento

Análise de Sistemas

Diagrama de Classes

Autor da apresentação

09/09/2014

Objetivo

- Um diagrama de classes serve para modelar o vocabulário de um sistema, do ponto de vista do utilizador/problema ou do implementador/solução
  - Ponto de vista do utilizador/problema –na fase de captura e análise de requisitos, em paralelo com a identificação dos casos de utilização
  - Vocabulário do implementador/solução –na fase de projecto (design)
- Construído e refinado ao longo das várias fases do desenvolvimento do software, por analistas, projectistas (designers) e implementadores
- Também serve para:
  - Especificar colaborações (no âmbito de um caso de utilização ou mecanismo)
  - Especificar esquemas lógicos de bases de dados
  - Especificar vistas (estrutura de dados de formulários, relatórios, etc.)

Diagrama de Classes

Objetivo

- Representa os tipos (classes) de objetos de um sistema
  - Propriedades(atributos) desses tipos
  - Funcionalidades providas por esses tipos
  - Relacionamentos entre tipos
- Pode ser convertido diretamente para uma linguagem orientada a objetos(POO)
  - Ajuda no processo transitório dos requisitos para o código
  - Pode representar visualmente o código do sistema

Objetos

- É algo bem definido, com estado em que é modelado por atributos e por ligações com outros objetos em dado momento.
- Tem comportamento ou acções em resposta a chamadas de operações ou eventos.
- Tem identidade:
  - Espacial, objetos com o mesmo estado são distinguíveis.

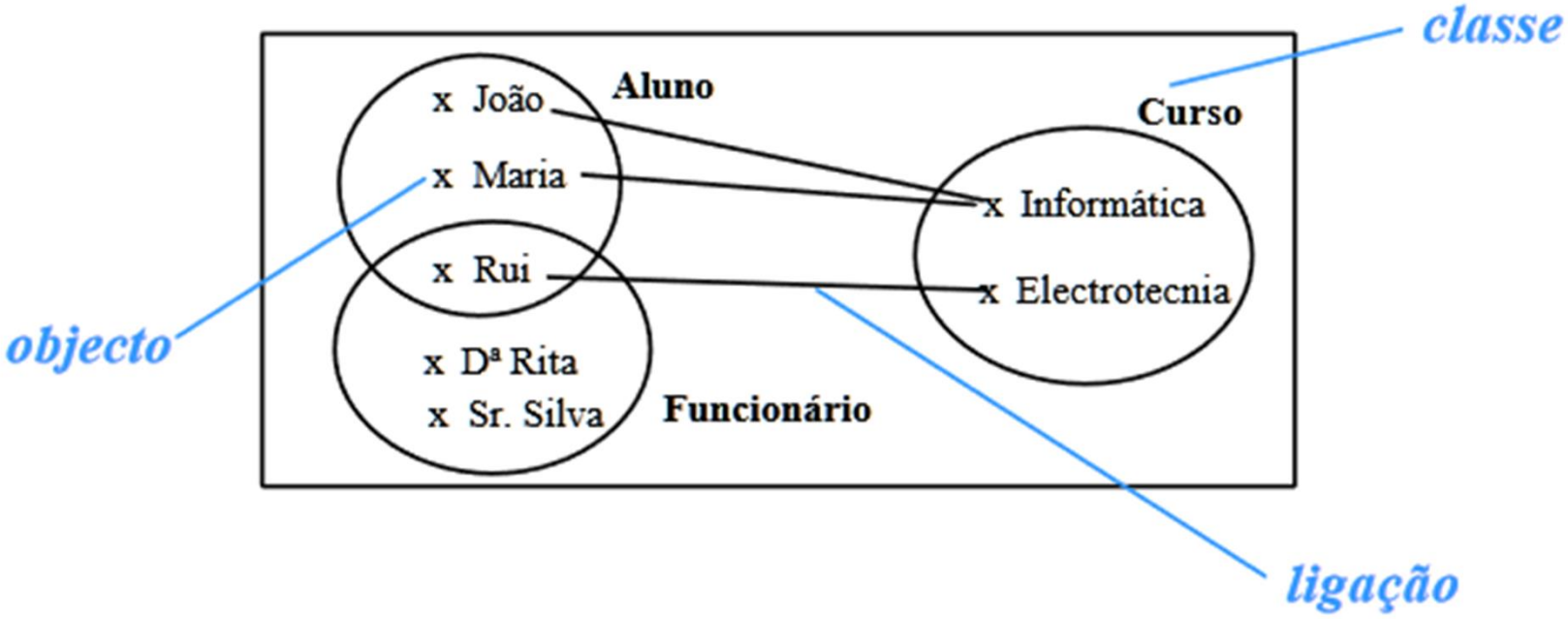
    Exemplo: distinguimos duas cartas em A5, mesmo que tenham os mesmos valores dos atributos.
  - Temporal, percebemos que se trata do mesmo objeto mesmo que o estado mude.

    Exemplo:se pintarmos um folha de papel A4 de amarelo, continua a ser a mesma folha de papel.

Classes

- No desenvolvimento de software OO, não nos interessam tanto os objectos individuais mas sim as classes de objetos.
- Uma classe é um descritor de um conjunto de objectos que partilham as mesmas propriedades (semântica, atributos, operações e relações).
- Um objecto de uma classe é uma instância da classe.
- O conjunto de todos os objectos num determinado contexto.

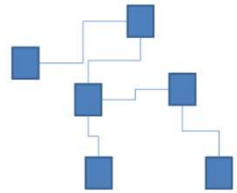
Classes


classe
Aluno
Curso
x João
x Maria
x Rui
x Dª Rita
x Sr. Silva
Funcionário
x Informática
x Electrotecnia
objecto
ligação

Classes

- Podem representar:
  - 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

Representação dos elementos em UML



- As classes são representadas por rectângulos
- Os relacionamentos por linhas.

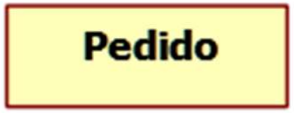
Para se precisar o significado pretendido para uma classe, deve-se explicar o que é uma instância da classe:

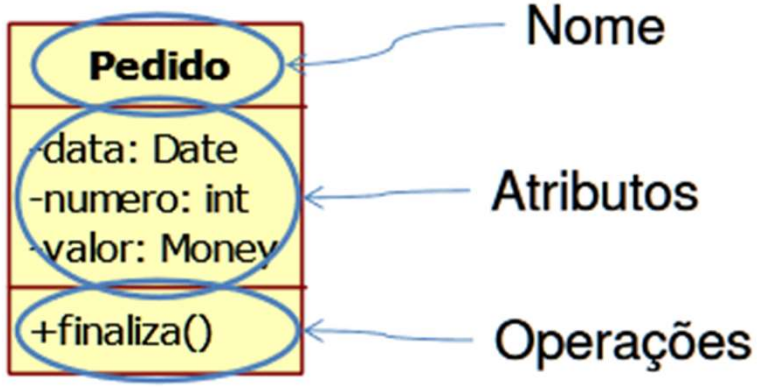
  - Exemplo: “Um aluno é uma pessoa que está inscrita num curso ministrado numa escola. Uma pessoa que esteve no passado inscrita num curso, mas não está presentemente inscrita em nenhum curso, não é um aluno.”

- Em geral, o nome da classe não é suficiente para se compreender o significado da classe

Classes

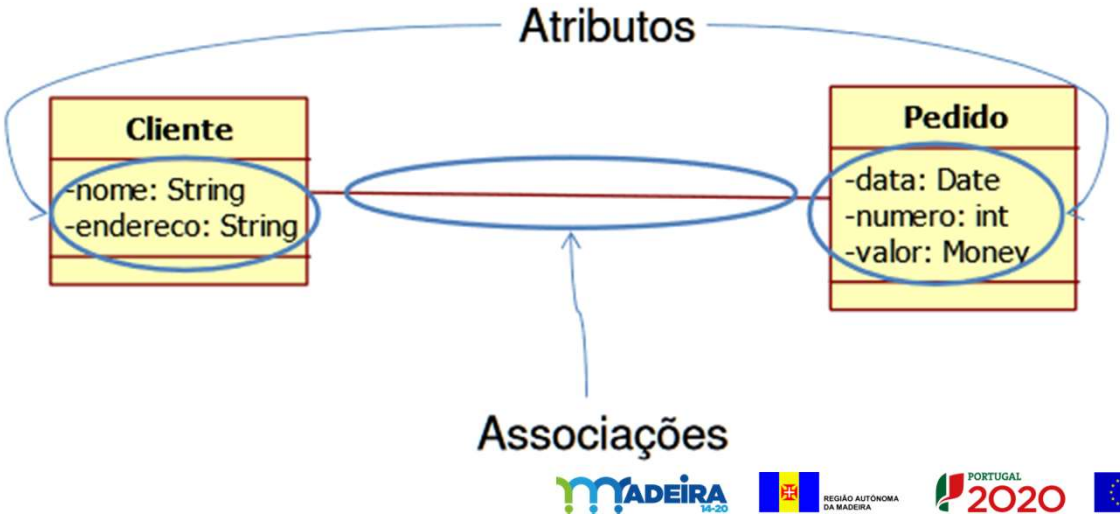
- Na representação das classes:
  - Nome(obrigatório)
  - Atributos
  - Operações(metodos)





Propriedades

- Classes são descritas pelas propriedades
  - Primitivas: representadas como atributos
  - Compostas: representadas como associações para outras classes
- As propriedades tornam-se sempre campos da classe.



Atributos

- Visibilidade(encapsulamento)
- Nome
- Tipo
- Multiplicidade
- Estado padrão

Exemplo:

endereco: String[1]= “Não definido”;

Atributos: Visibilidade(encapsulamento)

Princípio do encapsulamento

Esconder todos os detalhes de implementação que não interessam aos clientes (utilizadores) da classe.

Atributos: Visibilidade(encapsulamento)

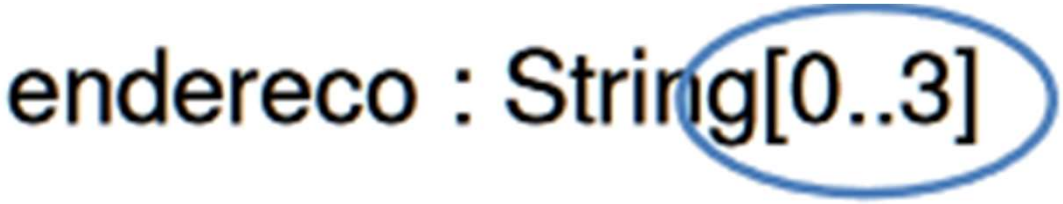
- Privado (-) - endereco : String
  - Apenas a própria classe pode manipular o atributo
- Pacote (~) ~endereco: String
  - Qualquer classe do mesmo pacote pode manipular o atributo
- Protegido (#) #endereco: String
  - Qualquer subclasse pode manipular o atributo
- Publico (+) +endereco: String
  - Qualquer classe do sistema pode manipular o atributo

Atributos: Nomes e tipos

- O nome do atributo corresponde ao nome que identificara uma propriedade da classe
  - É aceitável utilizar nomes com espaço e acentos na fase de análise
- O tipo do atributo corresponde ao tipo que representará a propriedade
  - Tipos primitivos da linguagem
  - Classes de apoio da linguagem (String, Date.)

Atributos: Multiplicidade da classe

- Número de instâncias que podem existir
  - Por omissão, é 0..*
- Representa o número de elementos de uma propriedade.
- Multiplicidade de atributos:
  - Número de valores que o atributo pode ter do tipo especificado.



Atributos: Multiplicidade da classe

- Número de instâncias que podem existir
  - Por omissão, é 0..*
- Representa o número de elementos de uma propriedade.
- Notação:

  1 - exactamente um

  0..1 - zero ou um (zero a 1)

  * - zero ou mais

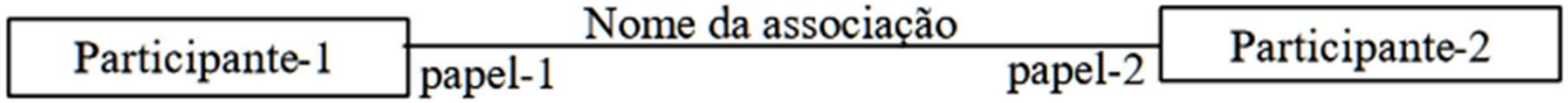
  0..* - zero ou mais

  1..* - um ou mais

  1, 3..5 - um ou três a 5

Associação

- Uma associação é a relação entre objetos das classe participantes
- Não gera novos objetos
- Só as classes que estão relacionadas são as classes cujos objetos podem se comunicar
- Identifica o papel das classes na associação

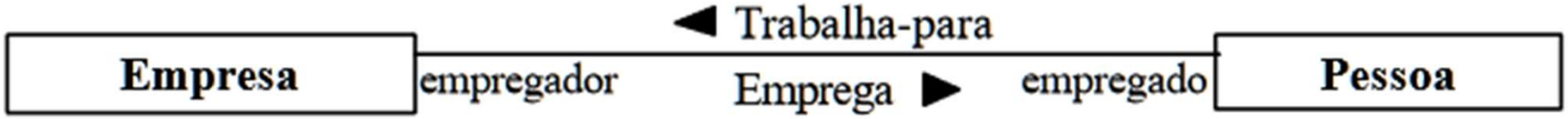


Associação - Atributos

- Nome
  - Nome da associação
- Papéis
  - Papéis das classes que estão relacionadas pela associação
  - O papel da classe A é o nome do atributo que a classe B possui que guarda o objetivo da classe A
- Multiplicidades
  - Quantidades de objetos associados a um papel
- Navegabilidade
  - Indica a direção da relação entre as classes

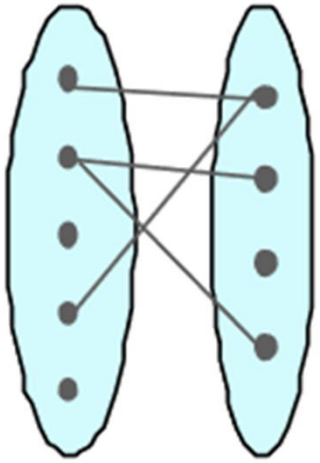
Associação - Nomes

- 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



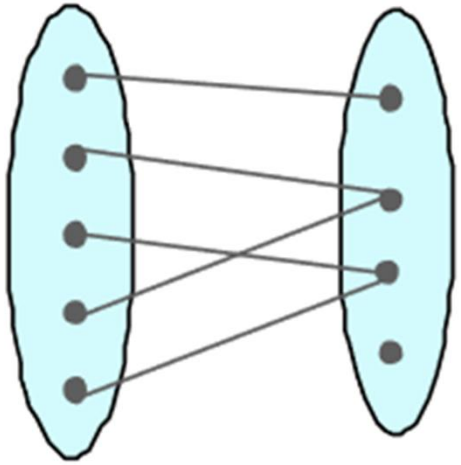
Associação – Multiplicidade Binária

Muitos-para-Muitos

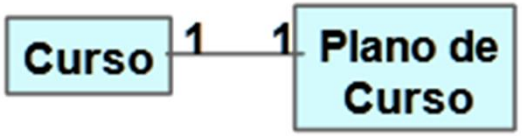


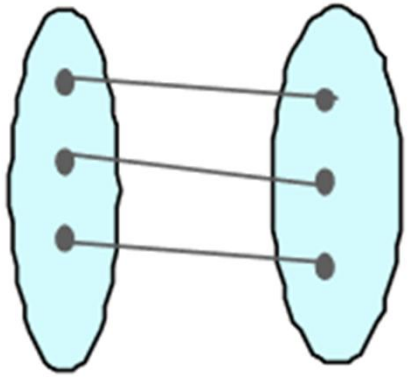
(sem restrições)

Muitos-para-1

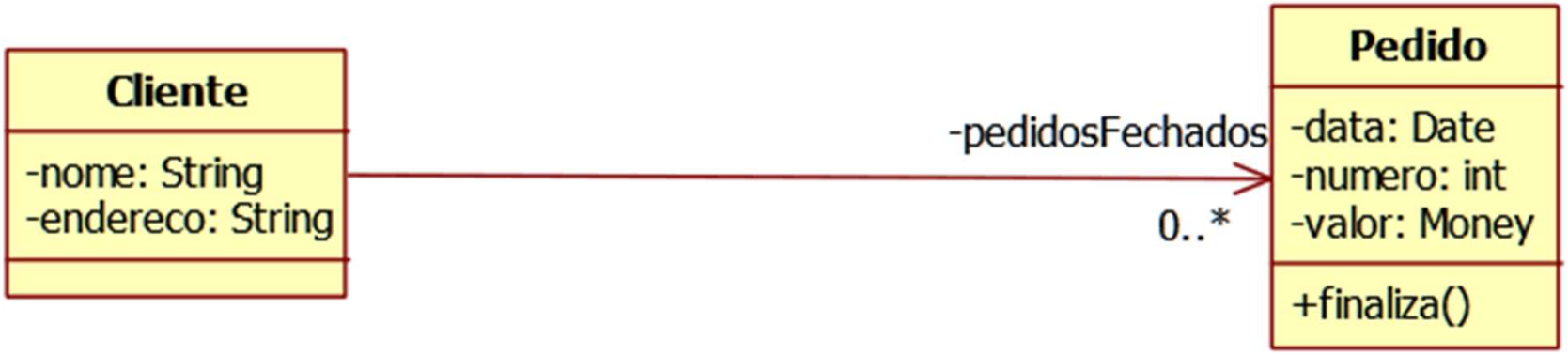


1-para-1





Associação - Exemplo



Associação reflexiva

- Pode-se associar uma classe com ela própria (em papéis diferentes).

Operações ou métodos

- Operações são descritas em função :
  - Visibilidade(igual aos atributos)
  - Nome (igual aos atributos)
  - Lista de parâmetros
  - Tipo de retorno

+ finaliza(data : Date) : String

Operações ou métodos : Lista de parâmetros

- A lista de parâmetros pode ser composta por zero ou mais parâmetros separados por vírgula
  - Parâmetro: nome : tipo [= valor padrão]
  - Nome
  - Tipo
    - Primitivo
    - Classe
  - Valor padrão (opcional)

Exercício

Uma loja que vende portáteis possui um sistema capaz de controlar a venda e o stock. Cada portátil possui um código de barras, uma marca e o número de exemplares que a loja possui.

Os clientes da loja são registados pelo nome.

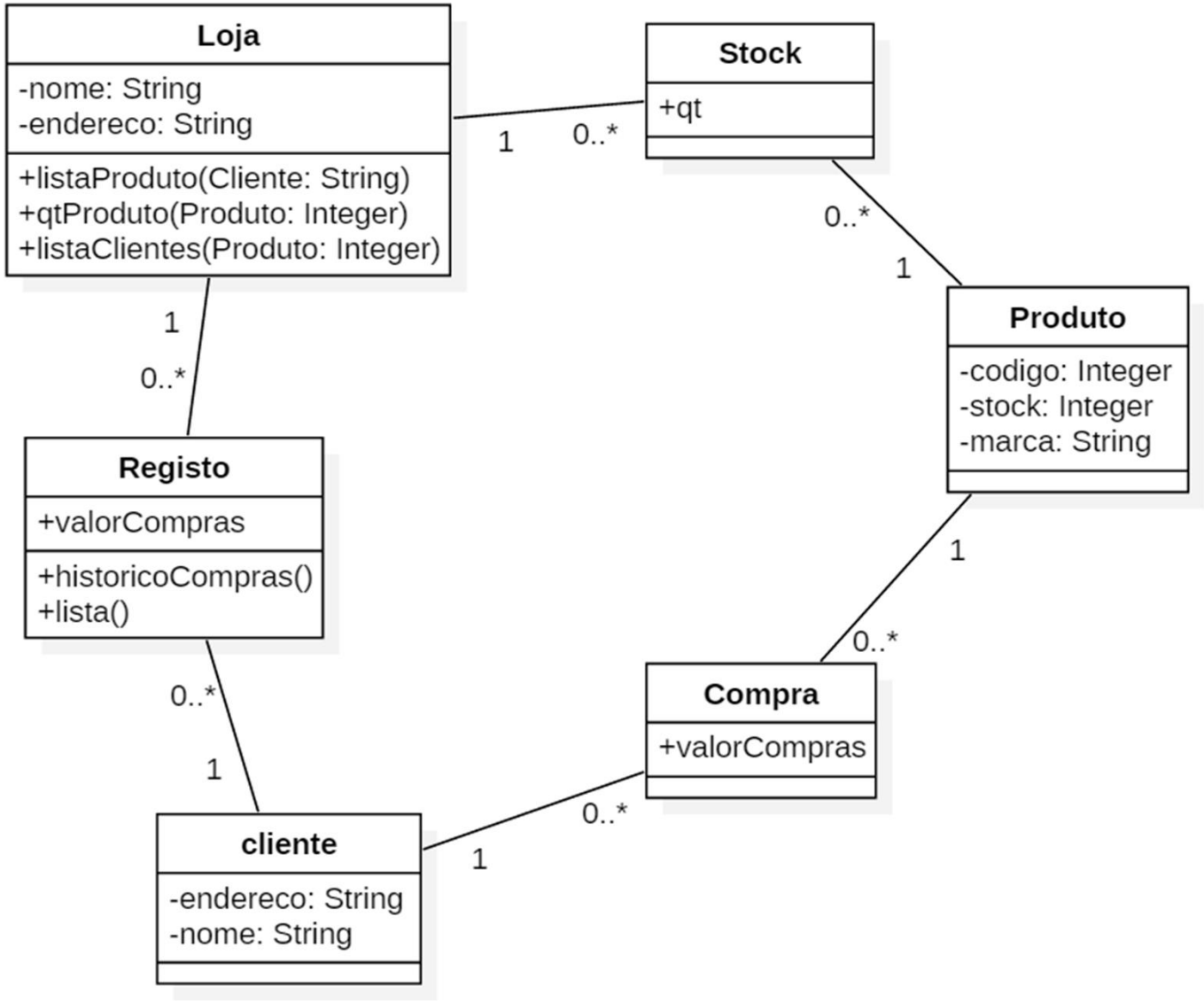
Faça um diagrama de classe que modele um sistema capaz de respondendo as perguntas :

–Quais foram os portáteis compradas por um cliente?

–Quais são os cliente que já compraram um determinada portátil?

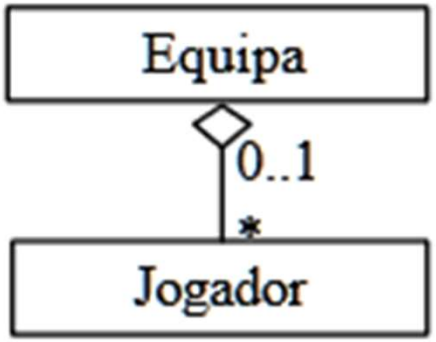
–Quantos exemplares possuem de um determinado portátil?

Exercício


Loja
-nome: String
-endereco: String
+listaProduto(Cliente: String)
+qtProduto(Produto: Integer)
+listaClientes(Produto: Integer)
Stock
+qt
Produto
-codigo: Integer
-stock: Integer
-marca: String
Registo
+valorCompras
+historicoCompras()
+lista()
Compra
+valorCompras
cliente
-endereco: String
-nome: String
1
0..*
1
0..*
0..*
1
1
0..*
0..*
1
1
0..*

Agregação

- Associação com significado contém(é constituído por)/faz parte de.
- O todo existe sem as partes e vice-versa.

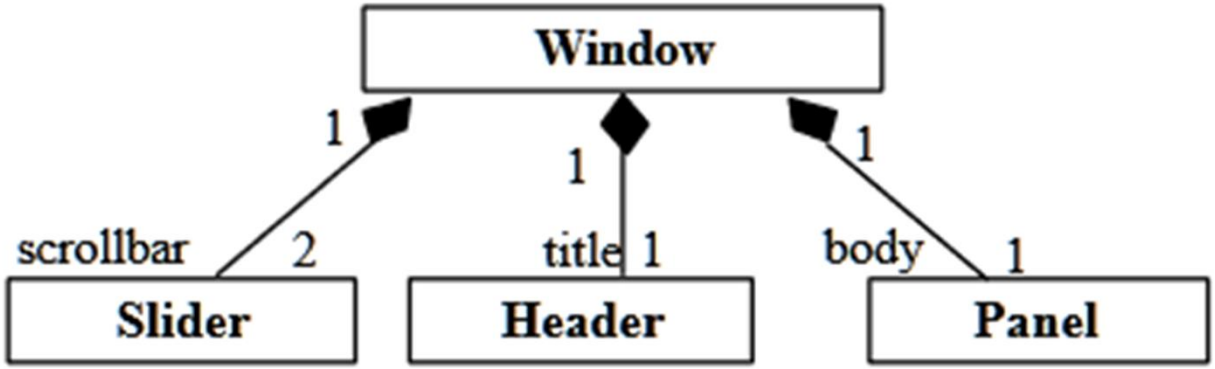


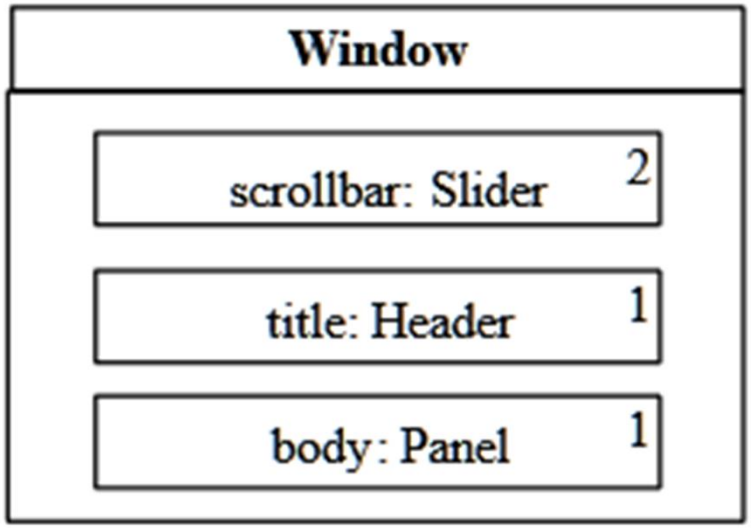
- Uma equipa **contém** *0 ou mais* jogadores
- Um jogador **faz parte de** *uma* equipa (num dado momento), mas também pode estar desempregado

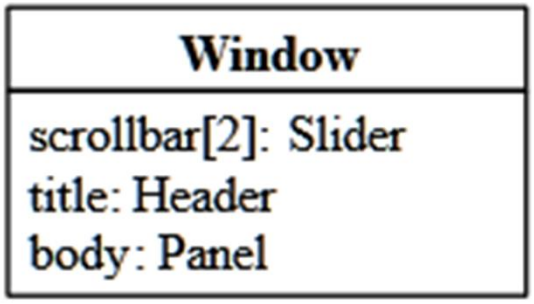
Composição

- Forma mais forte de agregação.
- Associação com significado de “é composto de”.
- As partes não existem sem o todo.
- Existe um forte grau de pertença das partes ao todo.
- O topo e as partes têm ciclo de vida coincidente.

Composição : Notação alternativa

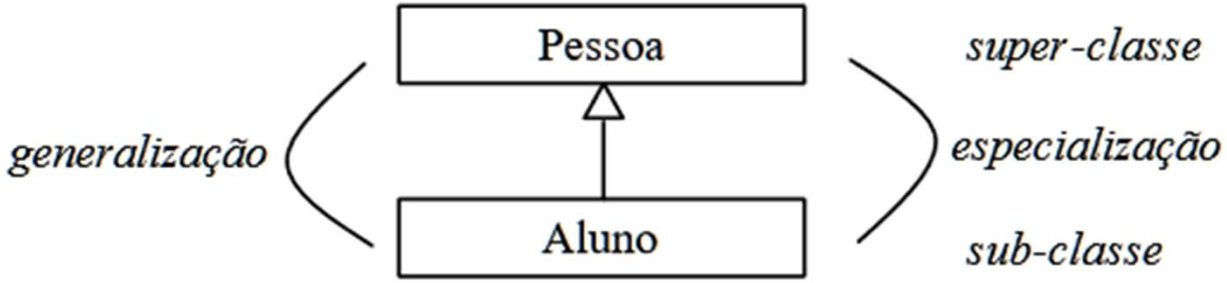






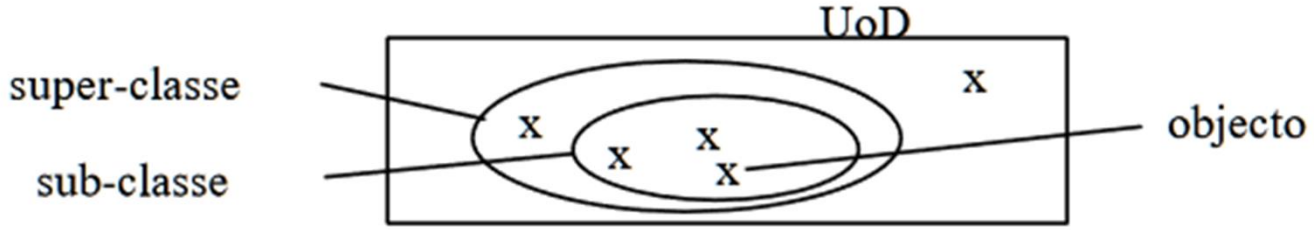
Generalização

Generalização



- **Relação semântica "is a" ("é um" / "é uma") :** um aluno é uma pessoa

- **Relação de inclusão nas extensões das classes:**



extensão (sub-classe) ⊂ extensão (super-classe)

- **Relação de herança nas propriedades:** A sub-classe herda as propriedades (atributos, operações e relações) da super-classe, podendo acrescentar outras

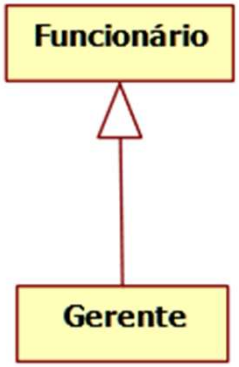
Generalização

Se Gerente "é um" Funcionário

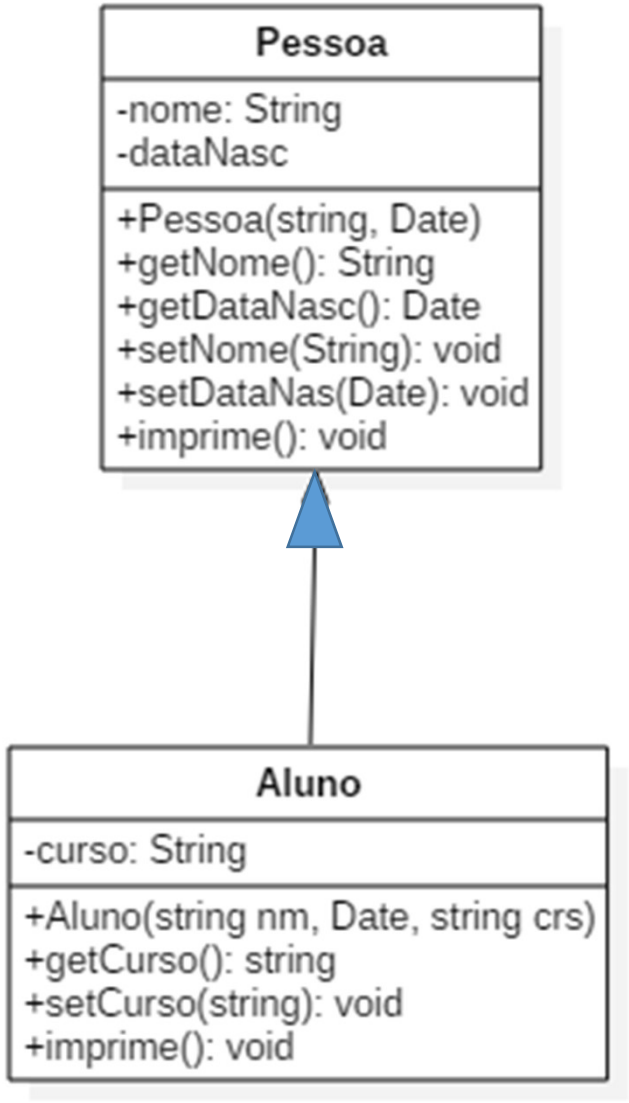
–Todas as operações e propriedades (não privadas) de Funcionário vão estar disponíveis em Gerente

–Toda instância de Gerente pode ser utilizada onde se instancia Funcionário

– Gera o efeito de herança e polimorfismo.



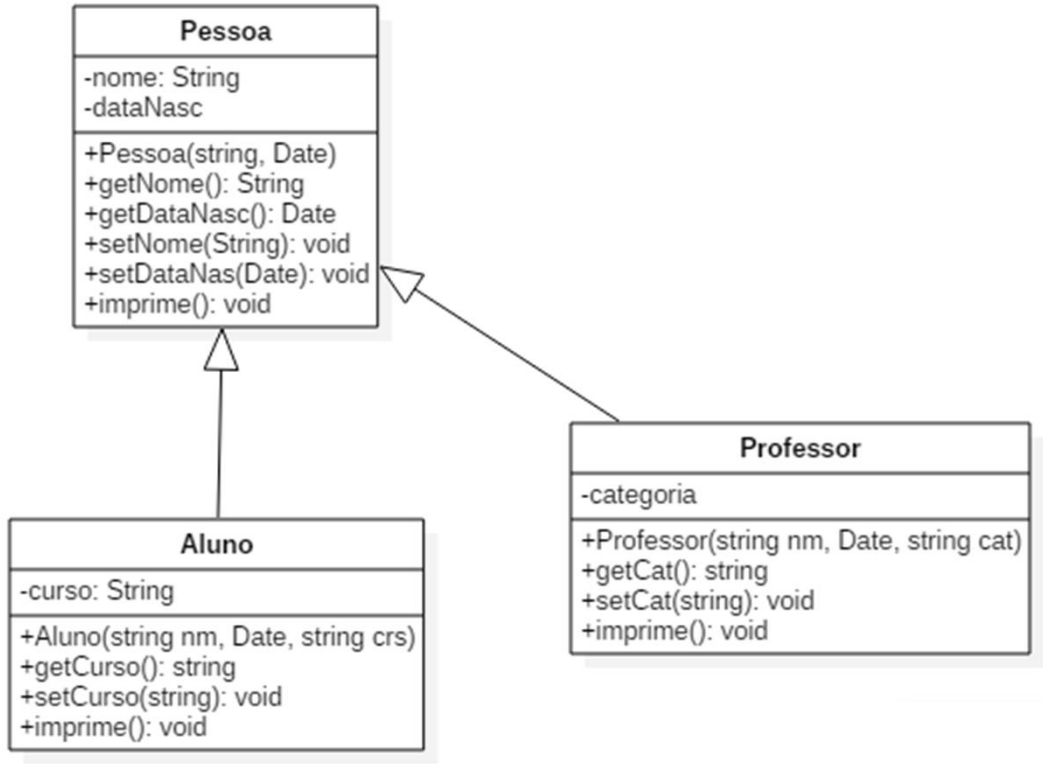
Generalização



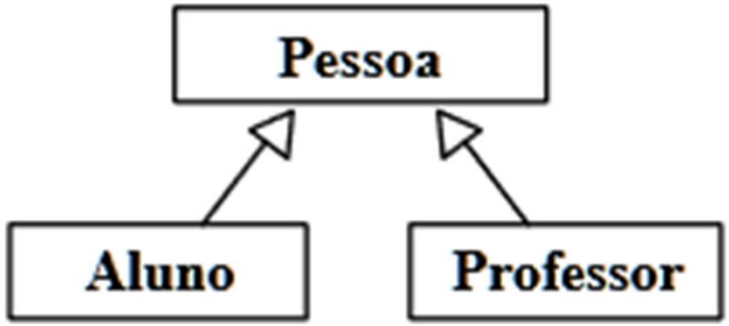
Hierarquias de classes

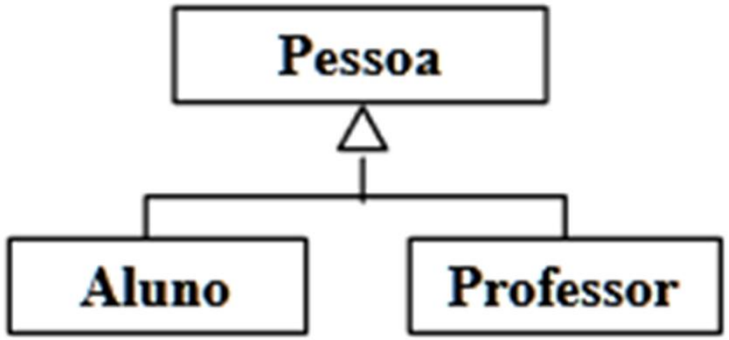
Em geral, pode-se ter uma hierarquia de classes relacionadas por herança / generalização.

- em cada classe da hierarquia colocam-se as propriedades que são comuns a todas as suas subclasses
- evita-se redundância, promove-se reutilização

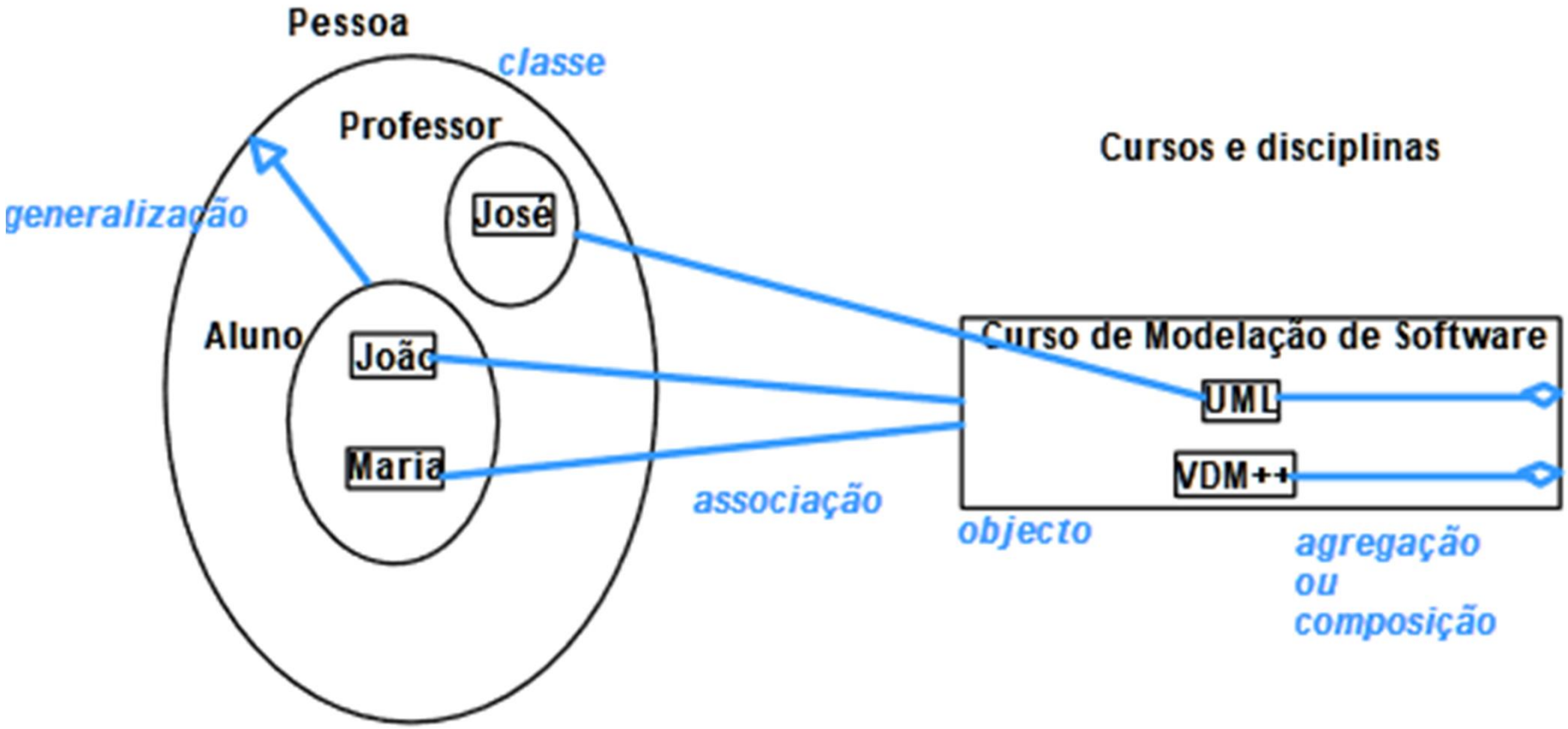


Notações alternativas para a hierarquia de classes





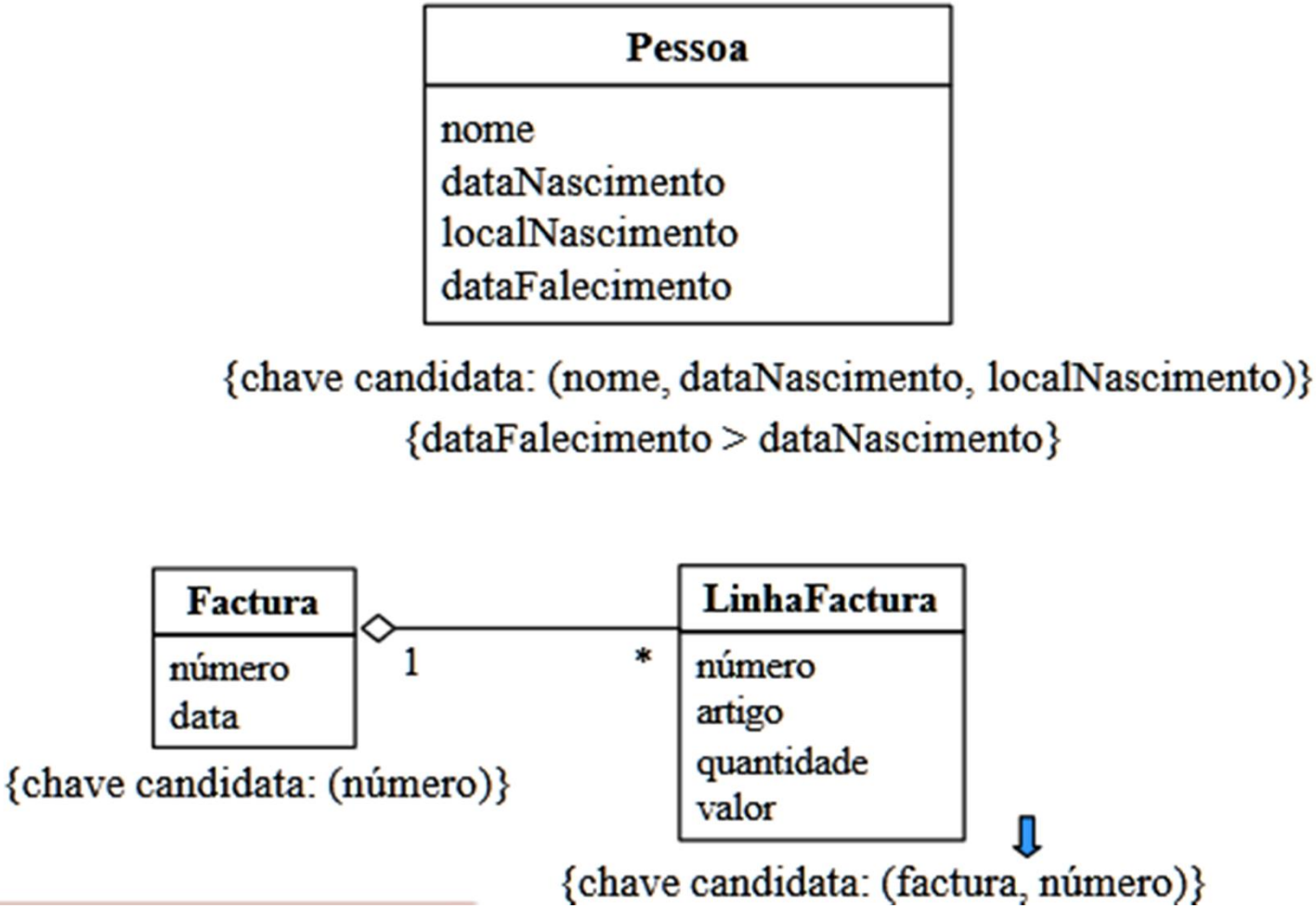
Agregação vs Generalização



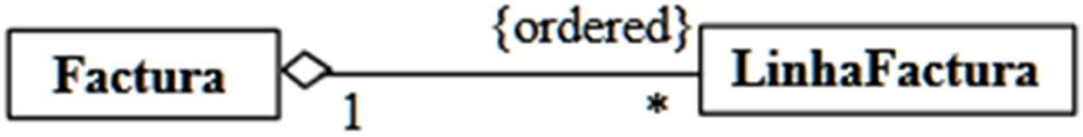
Restrições

- Uma restrição especifica uma condição que tem de se verificar no estado do sistema (objectos e ligações)
- Uma restrição é indicada por uma expressão ou texto entre chavetas ou por uma nota posicionada junto aos elementos a que diz respeito

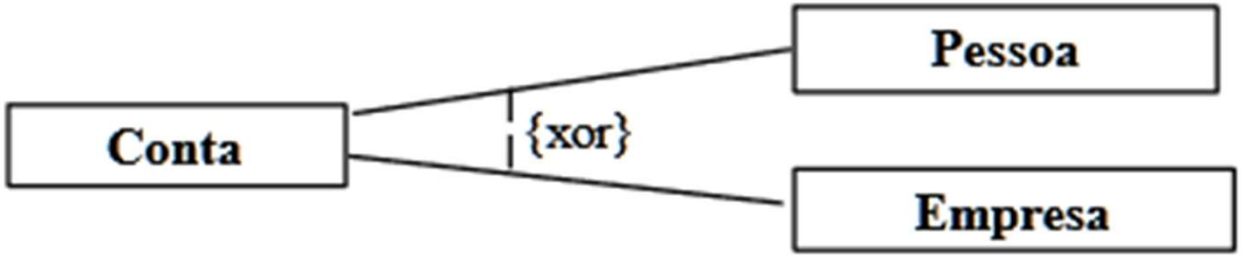
Restrições – Exemplo restrições em classes



Restrições – Exemplo restrições em associações



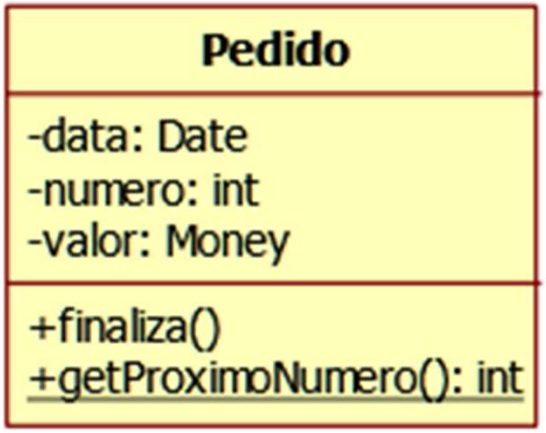
uma factura é constituída por um conjunto ordenado de 0 ou mais linhas



associações mutuamente exclusivas

Propriedades e operações estáticas

- Propriedade que não é instanciada no objeto
- Operações de escopo estático
- Acesso direto pela classe
  - Ex: Pedido.getProximoNumero()
- Não é necessário instanciar o objeto para aceder à propriedade.
- Sublinhada no diagrama.

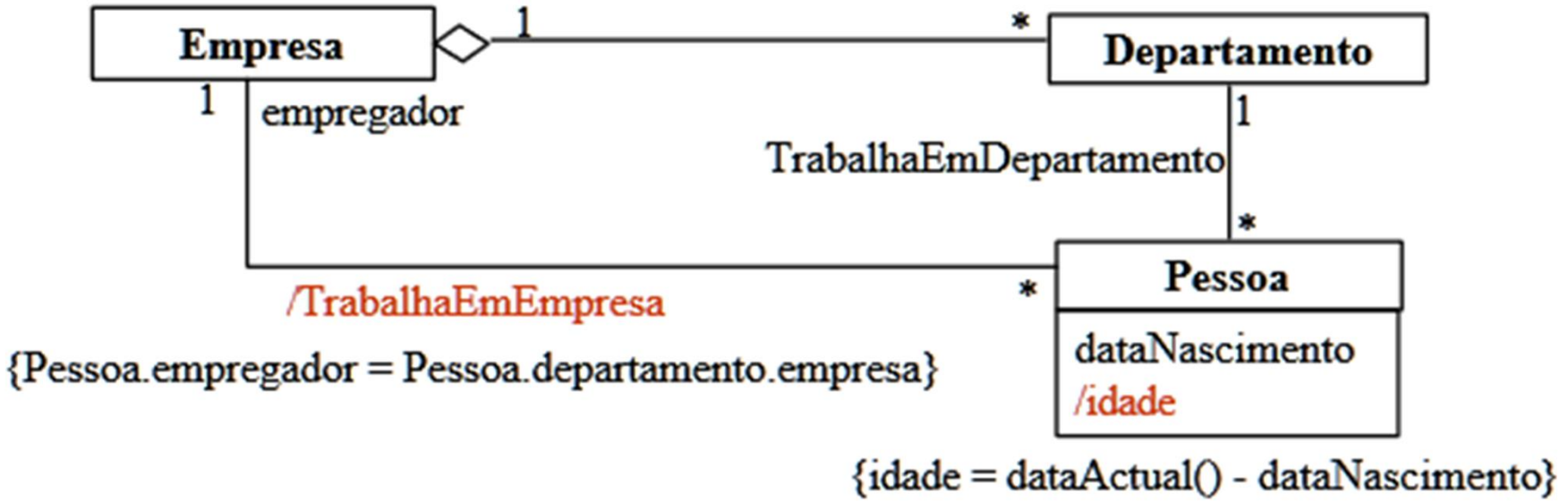


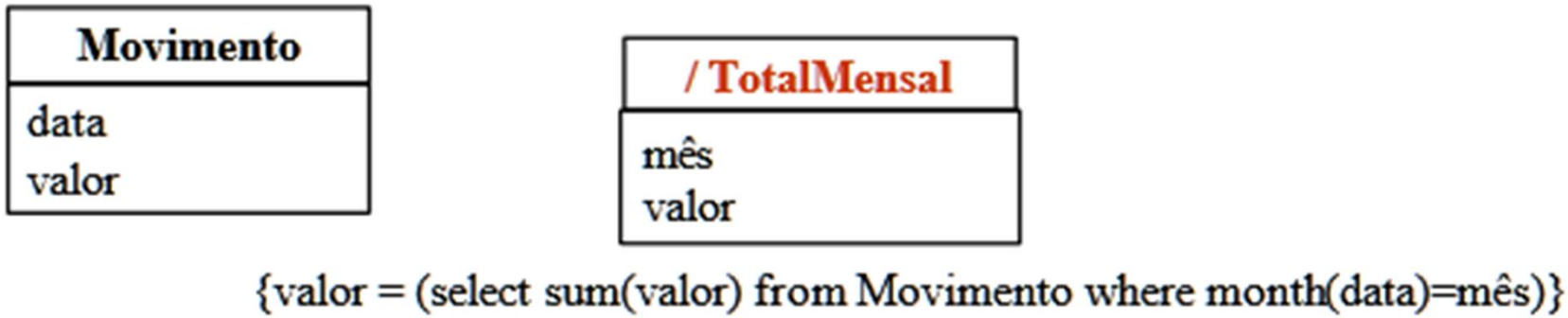
Propriedades derivadas

- São propriedades que na verdade não existem como atributos ou associações
- Podem ser inferidas por outras propriedades da classe
- É interessante explicitar através de nota ou restrição a fórmula de derivação
- São marcadas com o símbolo "/"

duração = fim - início

Propriedades derivadas

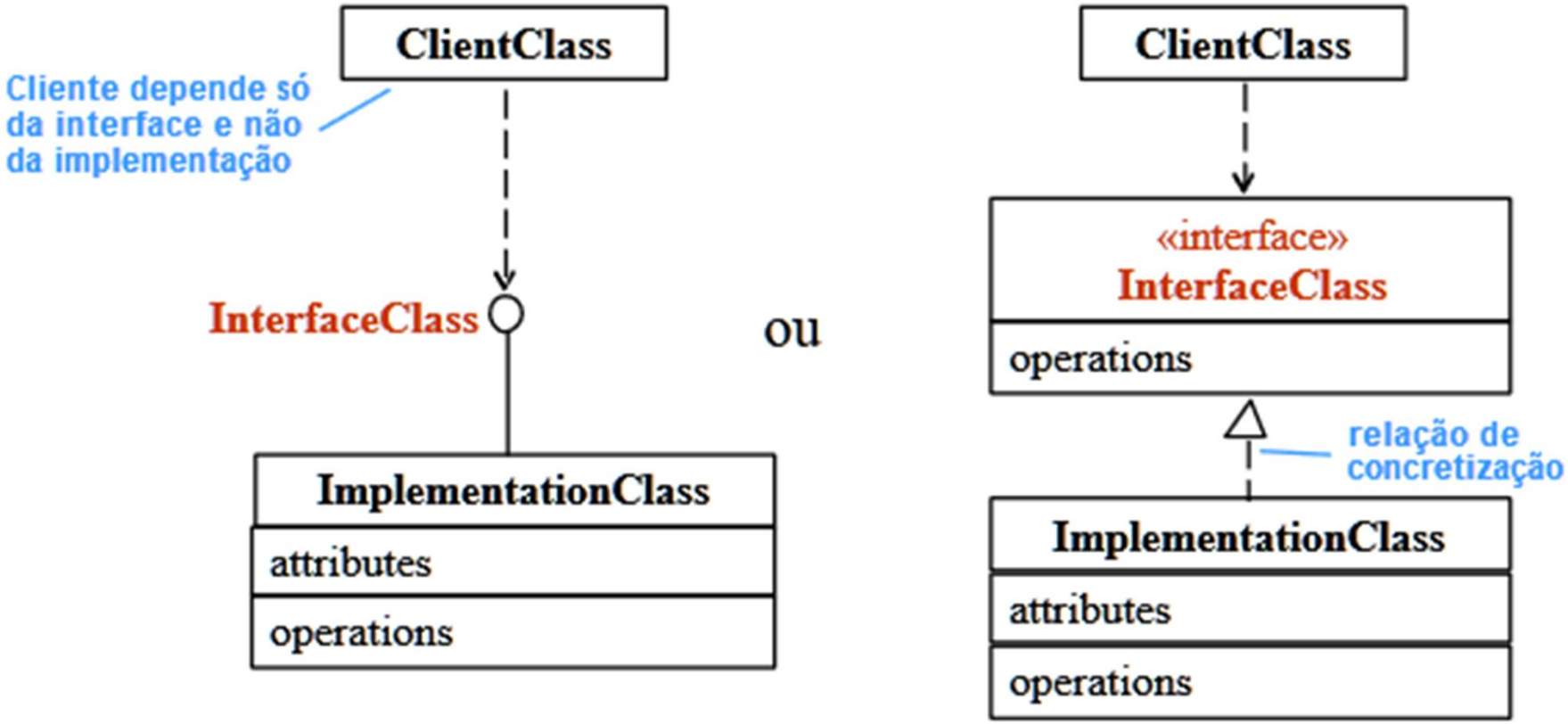




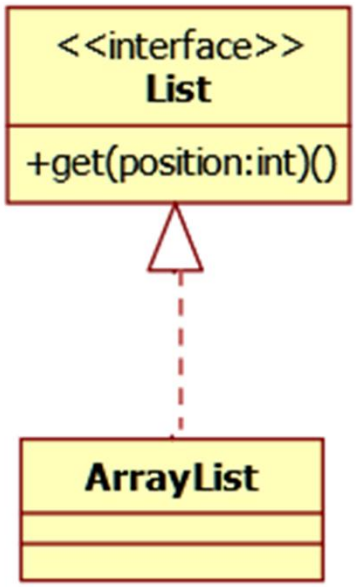
Interfaces

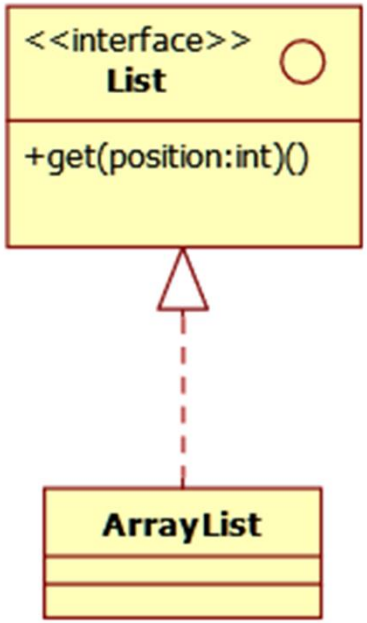
- Uma interface especifica um conjunto de operações (com sintaxe e semântica) externamente visíveis de uma classe.
- Uma classe sem nenhuma implementação
  - Todas operações são abstratas
- Faz uso da palavra-chave <<interface>>
  - Pode ser representado também como um ícone
- O relacionamento de realização indica as classes que implementam a interface
  - Equivalente a generalização

Interfaces: notações alternativas



Interfaces: notações alternativas





Cenários e etapas de selecção dos elementos

- O diagrama deve começar simples
- Todo o diagrama tem: classes, atributos, operações, associações.

Candidatos a classes:

- Entidades externas que produzem ou consomem informações.
- Informações compostas(Ex: Produto)
- Eventos que ocorrem no sistema(Ex. Pedido)
- Papéis de interacção no sistema(Ex. Cliente)
- Contexto do sistema(Ex. Loja)

Cenários e etapas de selecção dos elementos

- **Atributos**
  - Informação primitiva que precisa de guardada(Ex. preço)
- **Associações**
  - A classe A precisa relacionar-se com a classe B (Ex. Cliente - Pedido)
- **Operações**
  - Funcionalidades que devem ser fornecidas por um modelo(classe) por forma a produzir valor ao sistema(Ex. calcularTotal em Pedido)