Relacionamentos entre Classes

BCC 221 - Programação Orientada a Objectos(POO)

Guillermo Cámara-Chávez

Departamento de Computação - UFOP

Paradigma de Programação Orientada a Objetos I

Objetivo

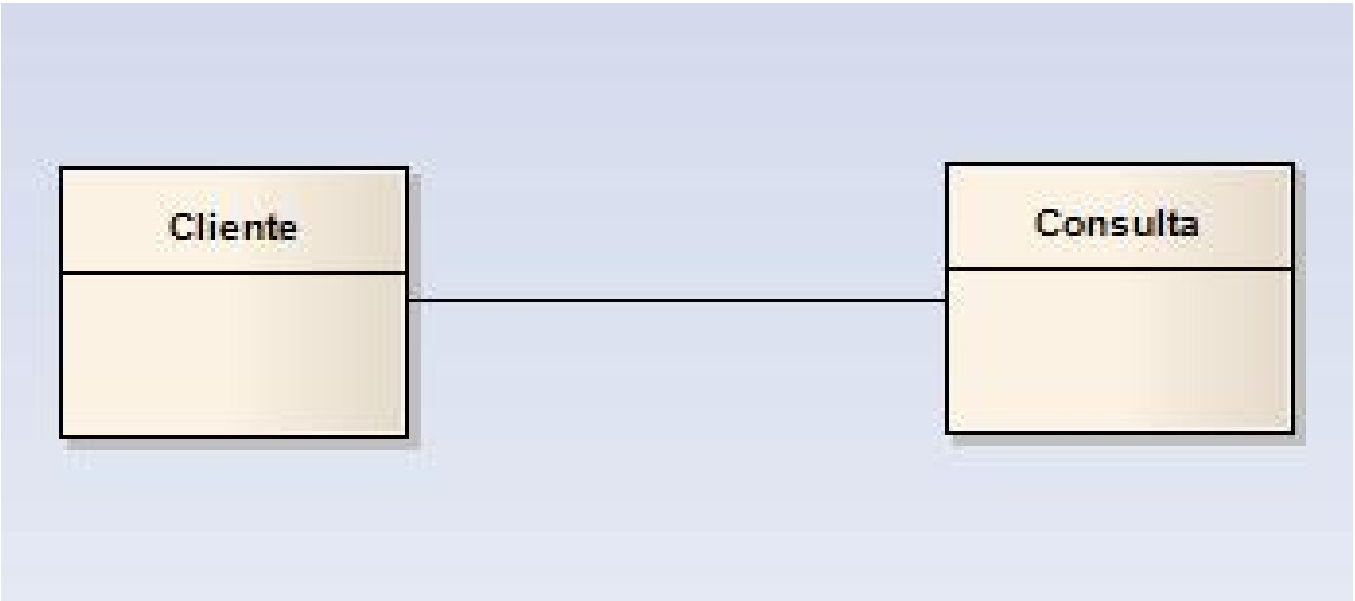
A programação orientada a objetos (POO) é um modelo desenvolvido para **aproximar o mundo real do mundo virtual**, simulando o real no computador.

Relacionamentos entre classes I

- Classes **possuem relacionamentos** entre elas (**comunicação**)
  - Compartilham informações
  - Colaboram umas com as outras
- Principais tipos de relacionamentos
  - Associação
  - Agregação / Composição
  - Herança
  - Dependência

Relacionamentos entre classes II

- Uma **Associação** é o mecanismo pelo qual um objeto utiliza os recursos de outro
- A função da mesma é só mostrar o relacionamento ou dependência de uma classe com a outra

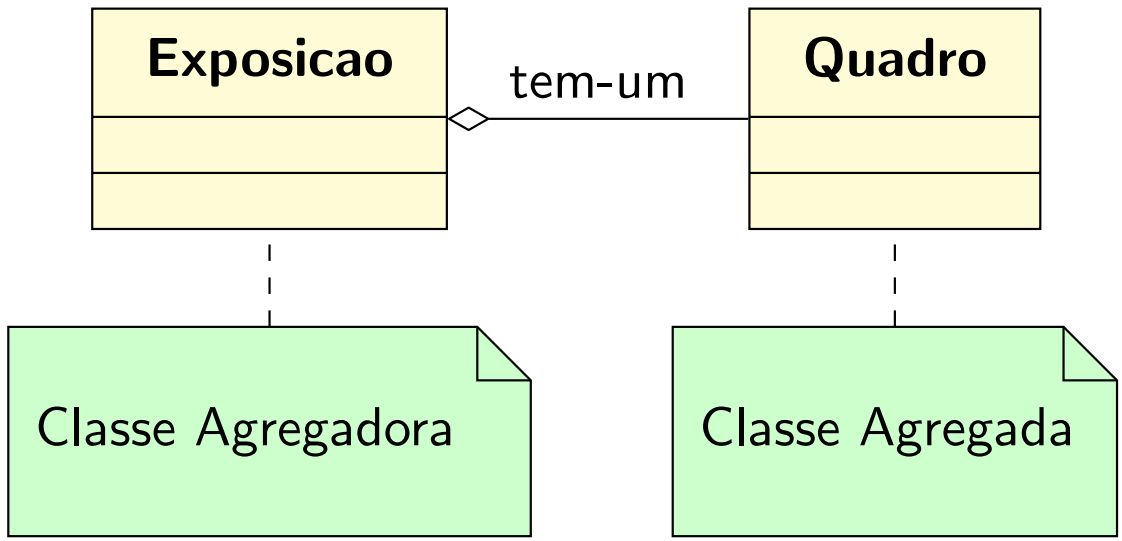


Relacionamentos entre classes III

- **Agregação** e **Composição** são tipos especiais de **Associação**;
- A associação de agregação é usada para indicar que um objeto colabora com outro.
- Pode ser uma associação simples
  - "**Usa um / tem um**" : Uma pessoa **usa um** computador
- Ou uma acoplamento
  - "**Parte de**" : O teclado é **parte de** um computador

Relacionamentos entre classes IV

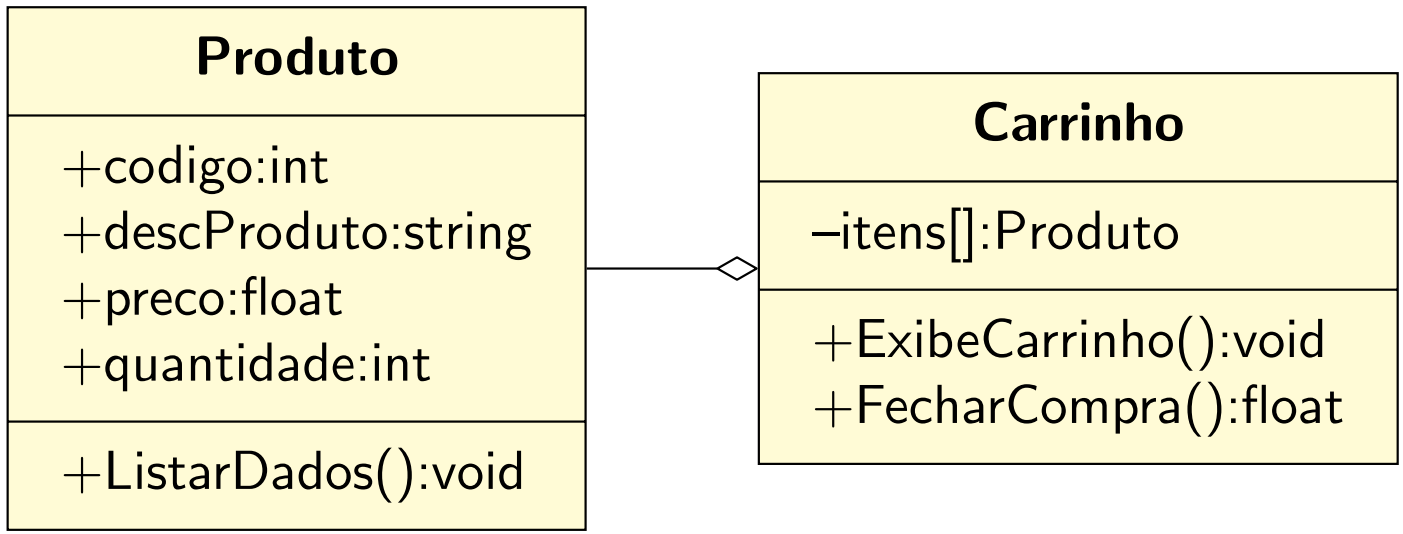
- Um objeto agregado **tem existência independente** do objeto agregador
- O objeto agregado **pode existir** após **eliminação** do objeto agregador



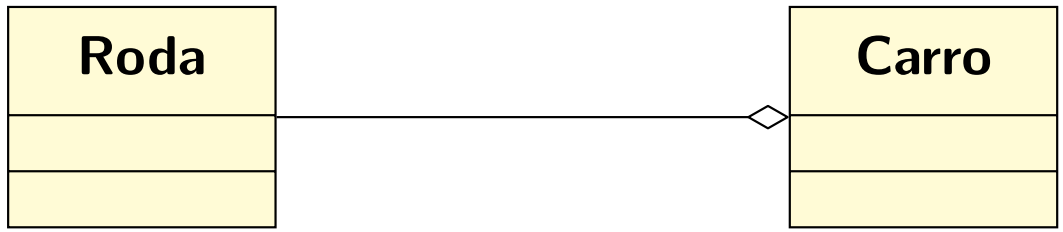
Relacionamentos entre classes V

- A classe Agregada faz parte da estrutura da classe Agregadora
- O objeto agregado **é parte do** objeto agregador (guardado em **variável de instância**

Relacionamentos entre classes VI



Relacionamentos entre classes VII

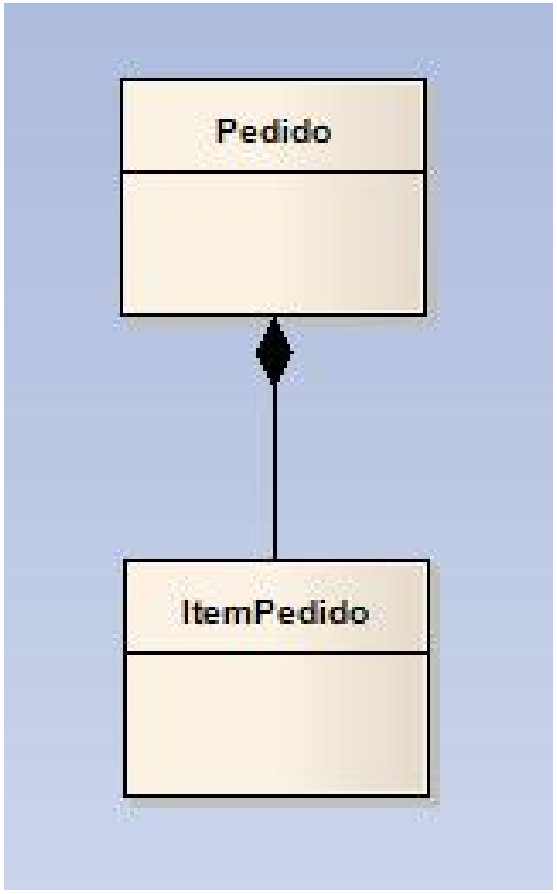


- A roda pode ser criada semanas antes e pode permanecer em um armazém até ser colocada no carro
- A instância da classe Roda tem uma vida claramente independente da instância da classe Carro.

Relacionamentos entre classes VIII

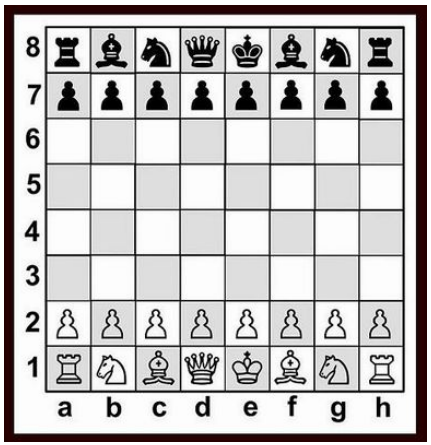
- A associação de **composição** é um tipo de associação de agregação com restrições na ligação entre parte e agregado
- A existência da parte depende da existência do agregado
  - O agregado (todo) não existe (ou não faz sentido sem as partes
  - Ou, as partes não existem sem o todo

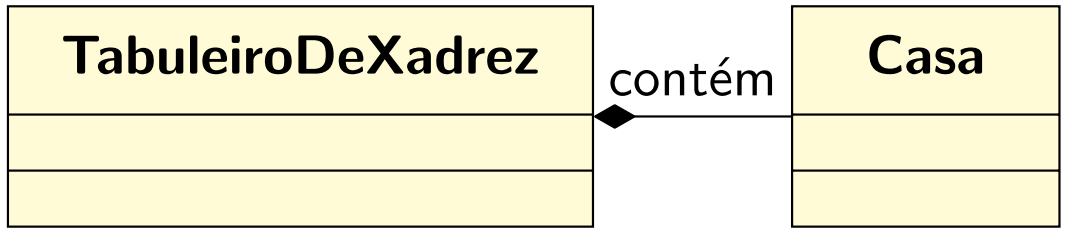
Relacionamentos entre classes IX



Um item de pedido não faz sentido existir sem um pedido, ou seja, dentro de um item de pedido sempre vou ter um pedido associado.

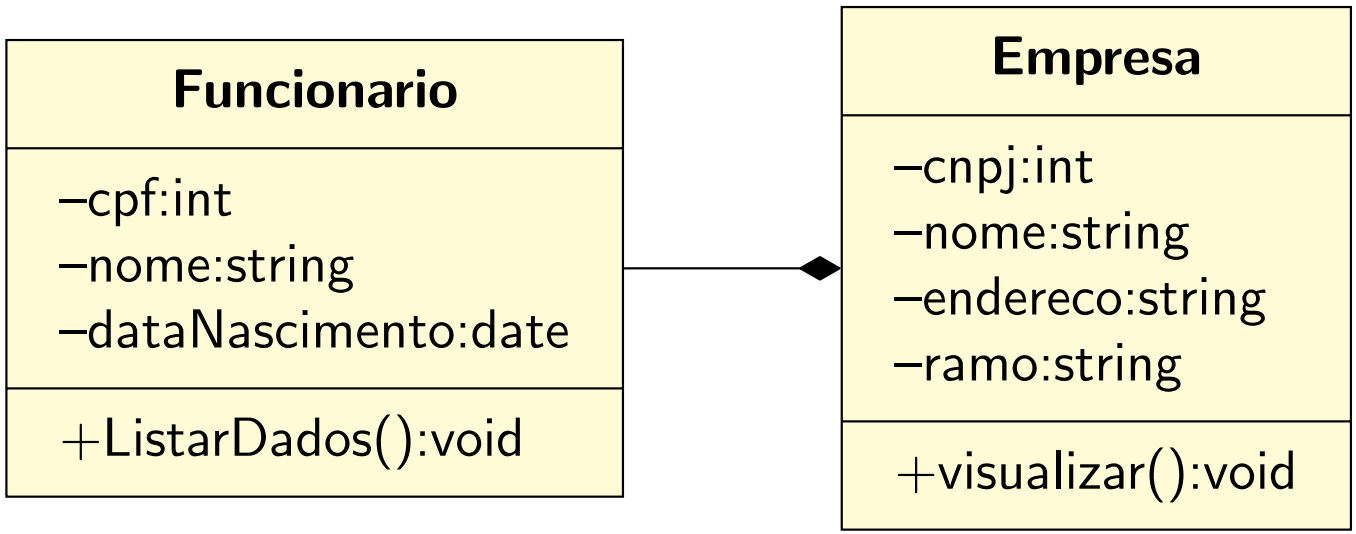
Relacionamentos entre classes X





Um tabuleiro de Xadrez é formado, composto por 32 casas pretas e 32 casas brancas

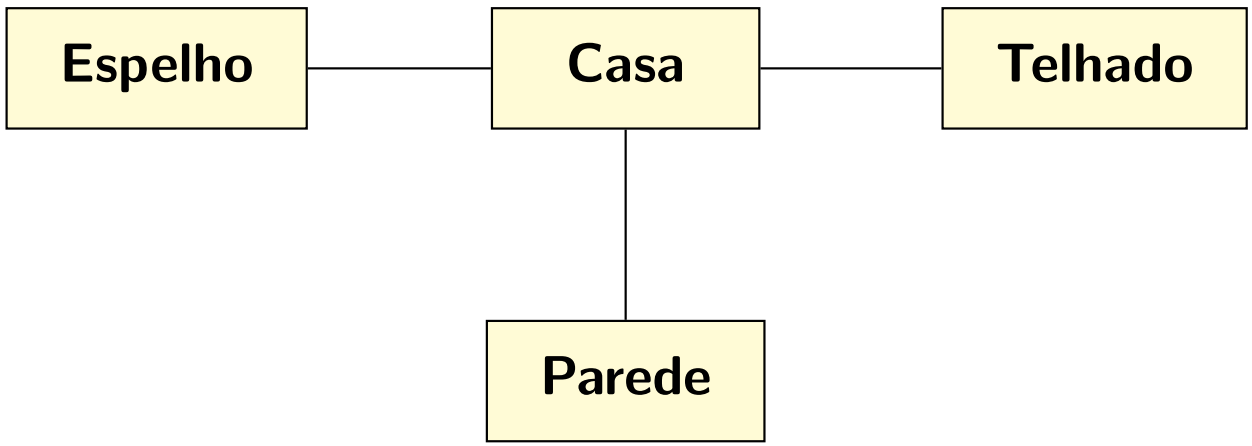
Relacionamentos entre classes XI



- Observe que não faz sentido ter funcionários, se não existir uma empresa onde eles possam trabalhar.
- Se a empresa deixar de existir, automaticamente ela deixará de ter funcionários.

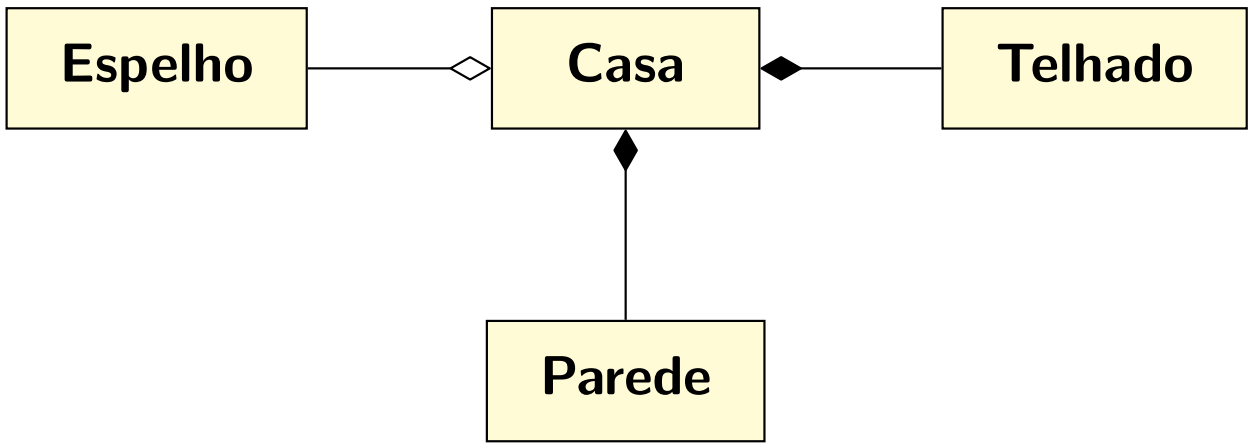
Relacionamentos entre classes XII

Quais seriam os tipos de associação?



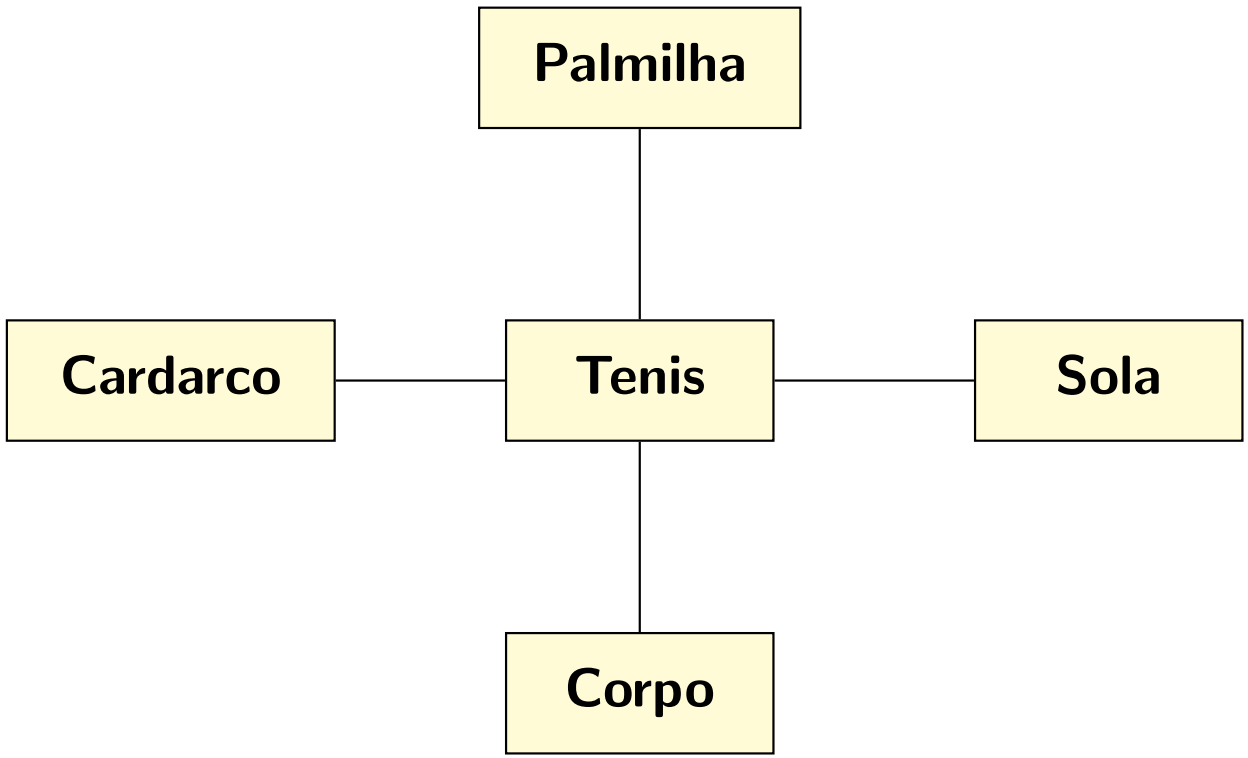
Relacionamentos entre classes XIII

Quais seriam os tipos de associação?



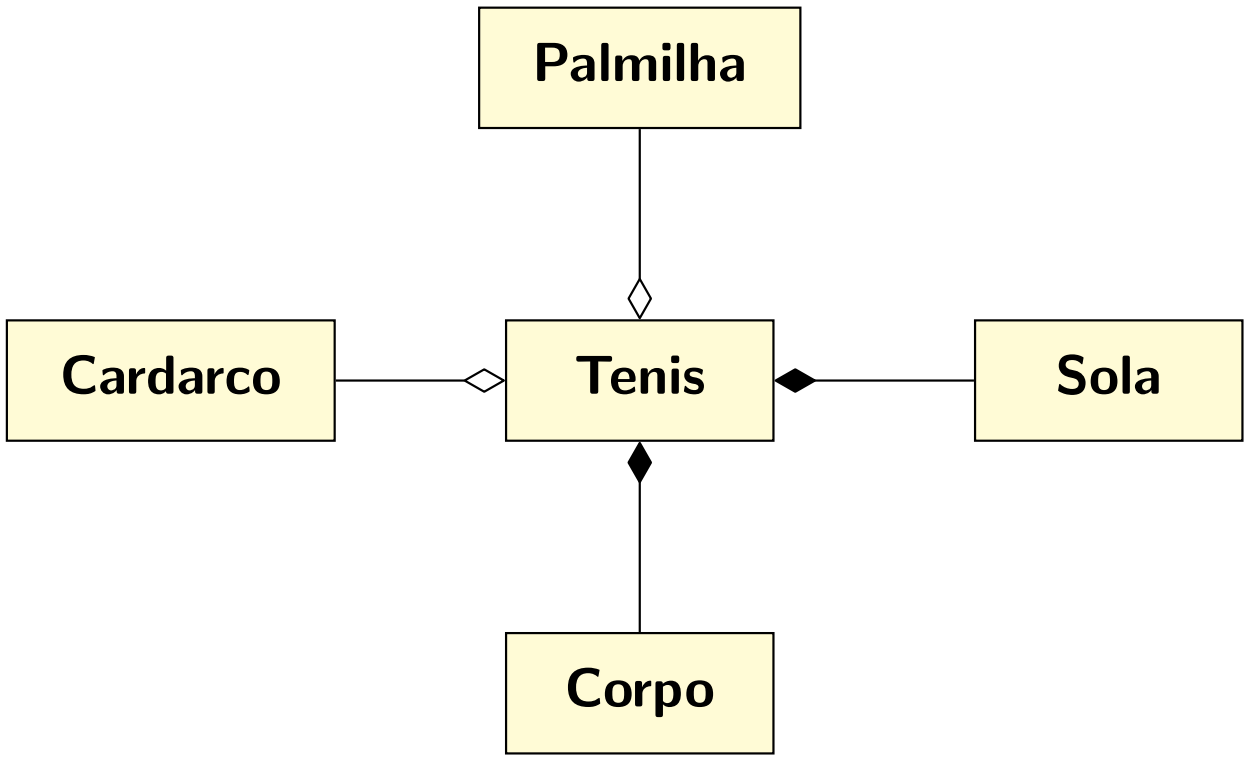
Relacionamentos entre classes XIV

Quais seriam os tipos de associação?



Relacionamentos entre classes XV

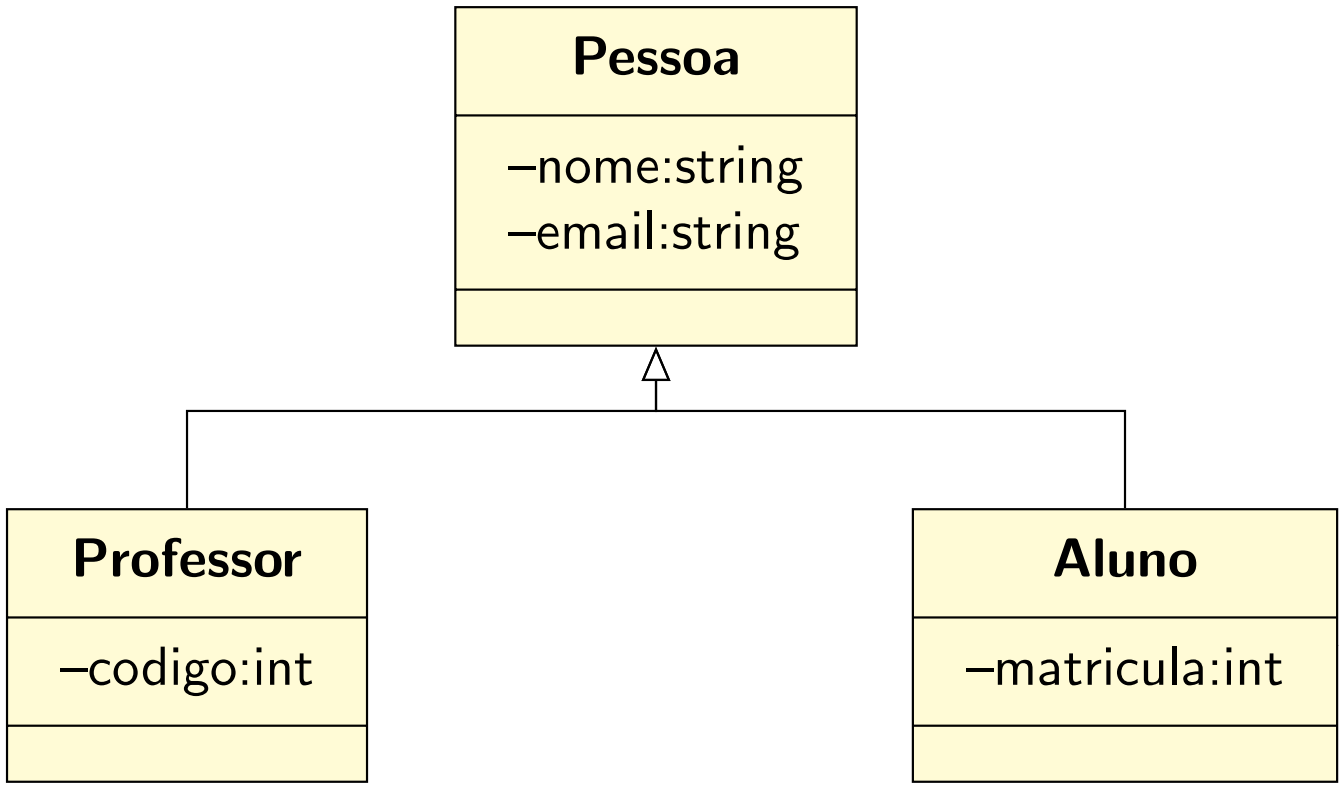
Quais seriam os tipos de associação?



Relacionamentos entre classes XVI

- Identificar super-classe (geral) e sub-clases (especializadas)
  - O relacionamento de **herança** define um relacionamiento do tipo "**é um**"
  - Tudo o que a classe geral pode fazer, as classes específicas também podem
- Atributos e métodos definidos na classe-mãe são **herdados** pelas classes filhas

Relacionamentos entre classes XVII



Relacionamentos entre classes XVIII

- Vantagens da herança
  - O gráfico de herança é uma fonte de conhecimento sobre o domínio do sistema
  - É um mecanismo de abstração usado para classificar entidades
  - **Mecanismos de reuso** em vários níveis

Relacionamentos entre classes XIX

- Classes de objetos não são autocontidas
  - Não podem ser compreendidas sem referencia às super-classes
- **O código da superclasse não estará disponível no código da subclasse**
  - É necessário que este bem documentado

Relacionamentos entre classes XX

- Podemos pensar sobre herança como algo semelhante a funções
  - Quando identificamos um trecho de código que se repete várias vezes, criamos uma função com aquele conteúdo
  - Quando identificamos várias características em comum em um grupo de classes, podemos criar uma superclasse
  - Objetivo: evitar redundância

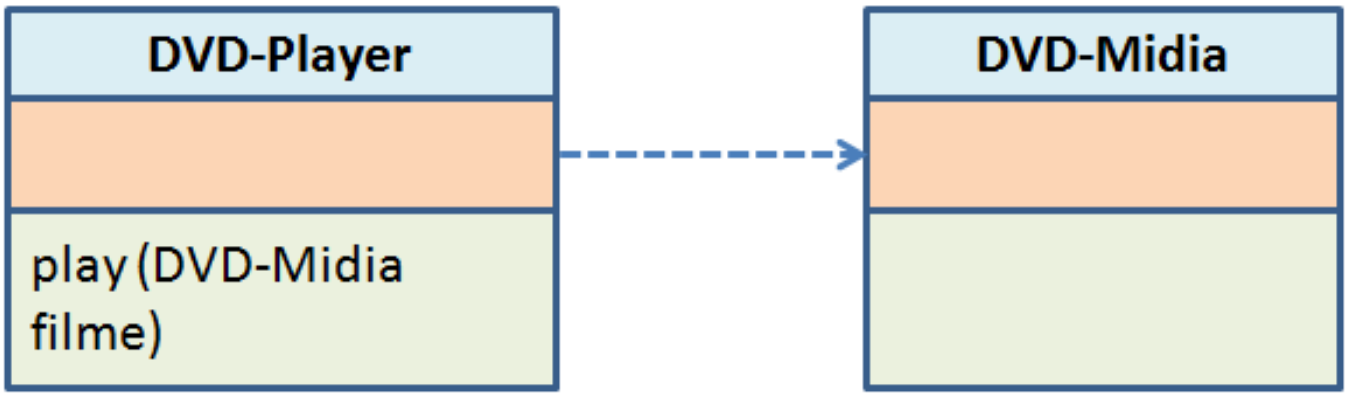
Relacionamentos entre classes XXI

- Uma **Dependência** é uma forma fraca de relacionamento
- Tipo menos comum de relacionamento
- Identifica uma ligação fraca entre objetos de duas classes
  - Uma classe depende de outra porque apenas em um momento específico ela a utiliza

Relacionamentos entre classes XXII

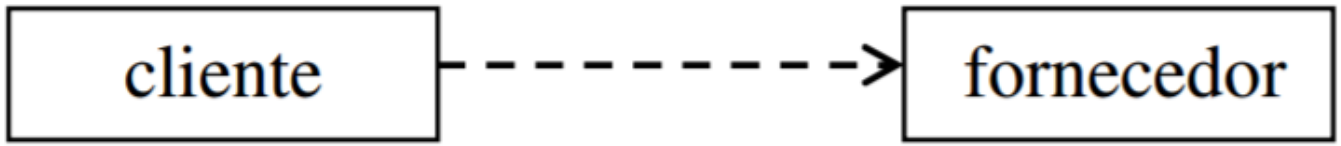
- Classe independente em declarações de (**existência** durante a execução do método, *i.e.*, temporária)
  - Variáveis **locais**
  - Parâmetros de métodos
- Conhecida como a relação "**knows about**" ("sabe sobre" ou "conhece")

Relacionamentos entre classes XXIII



O método *play* da classe DVD-Player recebo como parâmetro uma instância da classe DVD-Midia

Relacionamentos entre classes XXIV



- A classe cliente depende de algum serviço da classe fornecedor
- A mudança de estado do fornecedor afeta o objeto cliente
- A classe cliente não declara nos seus atributos um objeto do tipo fornecedor
- Fornecedor é recebido por parâmetro de método

Cuidado: a modificação da classe independente afeta a classe dependente

FIM