

Programação Orientada a Objetos

UML

PROFESSOR:

Demetrios Coutinho

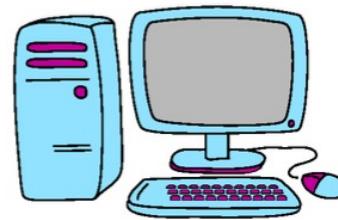
Demetrios.coutinho@ifrn.edu.br

AGENDA

- ~~Histórico da UML~~
- ~~Diagrama de classes~~
- ~~Representação de classes~~
 - Atributos e métodos
 - ~~Tipos de acesso e modificadores~~
- Relacionamentos entre classes
 - ~~Herança, Implementação, Associação, Agregação e Composição~~

Agregação/Composição

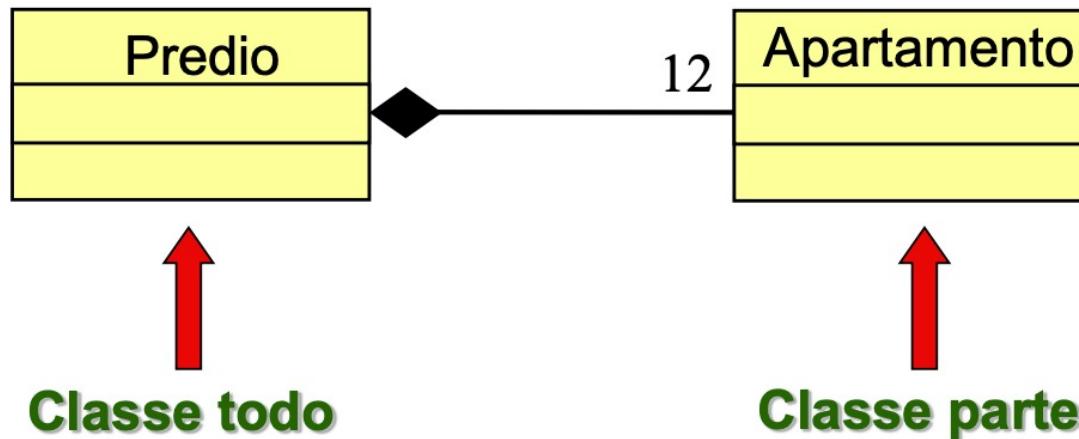
- São também formas de associação, mas representam relacionamentos do tipo “**tem um**”
 - Uma classe é **formada por** ou **contém** objetos de outras classes
 - Exemplos
 - Um carro possui rodas
 - Uma árvore é composta de folhas, tronco, raízes, ...
 - Um computador é composto de CPU, memória, teclado, mouse, monitor, ...



Agregação/Composição

- Classe todo
 - É a classe resultante da agregação/composição
- Classe parte
 - É a classe cujas instâncias formam a agregação/composição
- Exemplo de **composição**: Predio e Apartamento
 - Um prédio tem apartamentos
 - Classe **Predio**: todo ou agregada
 - Classe **Apartamento**: parte

Composição



- Prédio tem como atributo um conjunto (*array*) de apartamentos
- Se o prédio deixar de existir, os apartamentos também deixam de existir
- Segundo a cardinalidade, um prédio precisa ter obrigatoriamente 12 (exatos) apartamentos

Composição

- Na composição, o **todo** é responsável pelo ciclo de vida da **parte**.
- Também se diz que o **todo** é dono da **parte**, e não apenas “possui a **parte**”
- Assim, em composição, a criação da **parte** ocorre no **todo**.

Agregação

- Agregação é uma forma mais fraca de composição
- **Composição:** relacionamento todo-parte em que as partes não podem existir independentes do todo
 - Se o todo é destruído, as partes são destruídas também
 - Uma parte pode ser de um todo por vez
- **Agregação:** relacionamento todo-parte que não satisfaz um ou ambos os critérios
 - A destruição do objeto não implica a destruição do objeto parte
 - Um objeto pode ser parte componente de vários outros objetos

Agregação/Composição

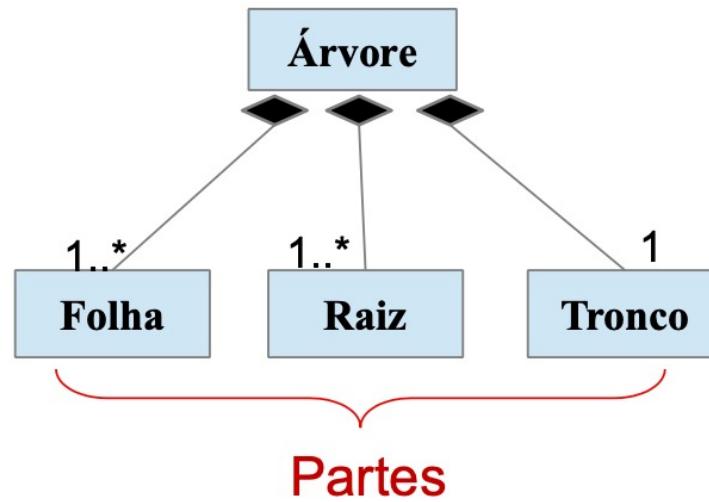
- No diagrama de classes
 - Composição
 - Associação representada com um losango **sólido** do lado **todo**



- Agregação
 - Associação representada com um losango **sem preenchimento** do lado **todo**



Composição

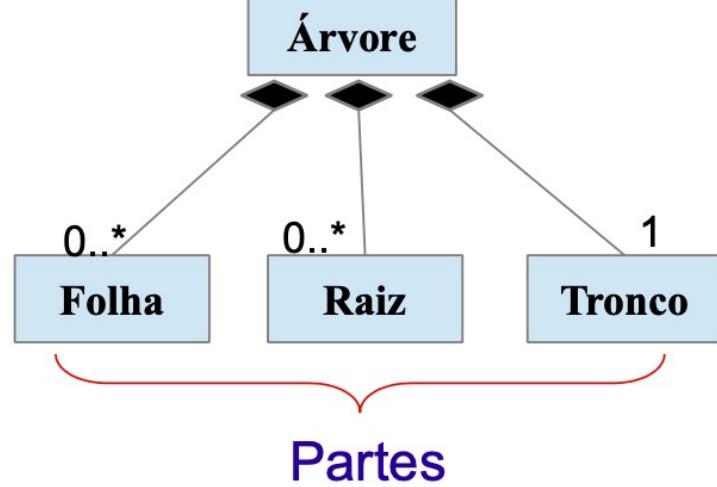
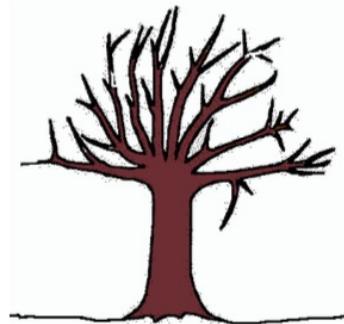


composição



- Não faz sentido que **Folha**, **Raiz** ou **Tronco** existam sem que sejam atributos de uma **Árvore**
 - Neste modelo em particular
- Ainda segundo o diagrama, não pode haver uma **Árvore** sem **Folha**, **Raiz** ou **Tronco** (cardinalidade)

Composição

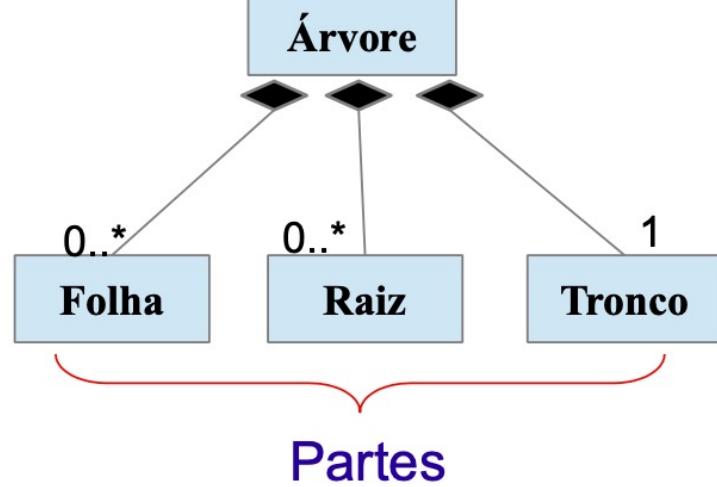
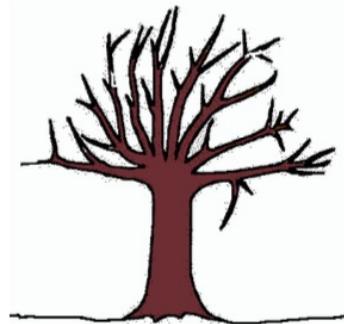


composição



- Agora pode haver uma Árvore sem Folha e sem Raiz, mas não sem Tronco

Composição



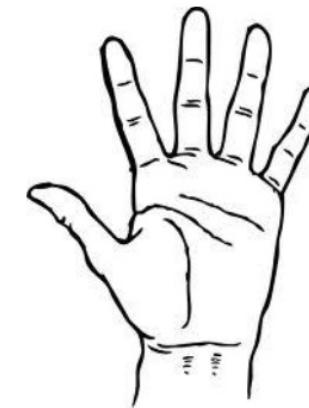
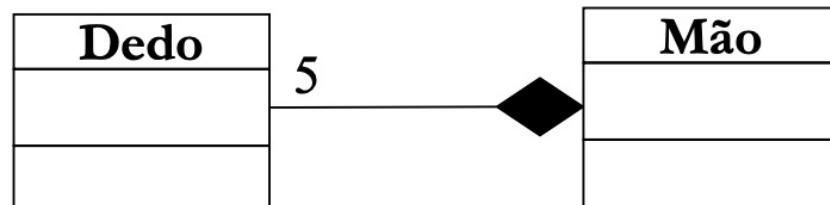
composição



- Agora pode haver uma Árvore sem Folha e sem Raiz, mas não sem Tronco

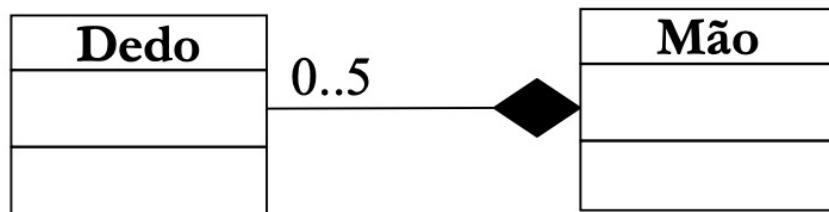
Composição

- Não faz sentido que um **Dedo** exista se não for parte de uma **Mão**
- Segundo a cardinalidade, não pode haver uma mão sem dedos
 - Todo mão tem exatos 5 Dedos



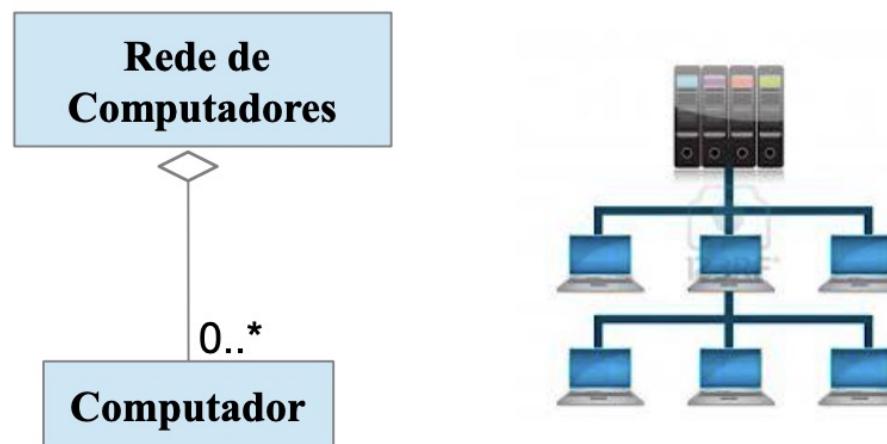
Composição

- Não faz sentido que um **Dedo** exista se não for parte de uma **Mão**
- Na definição de agora, uma mão pode não ter dedos
 - Cardinalidade mínima é 0 e máxima é 5



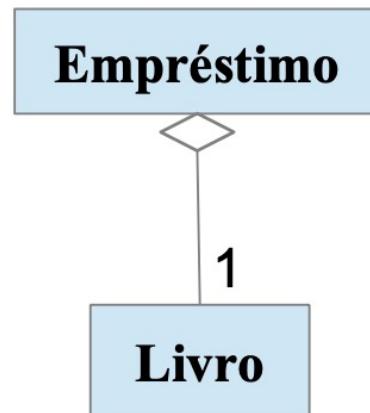
Aggregação

- Uma **Rede** pode ter nenhum ou muitos **Computadores**
- Um computador existe independentemente de uma rede
- Um computador pode estar ligado a mais de uma rede ao mesmo tempo



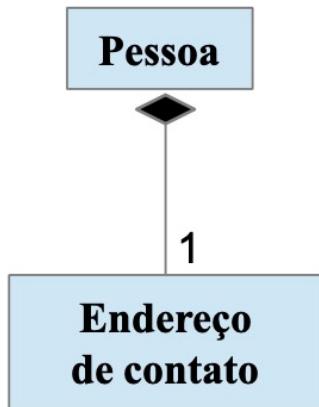
Agregação

- Um **Livro** existe independente de um **Empréstimo**
- Porém, um **Empréstimo** precisa ter pelo menos um **Livro**

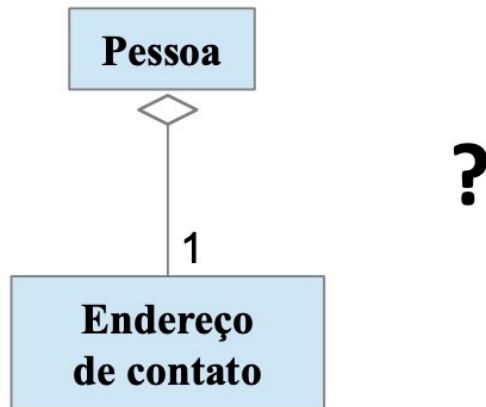


Composição/Agregação

- Quando usar composição ou agregação?

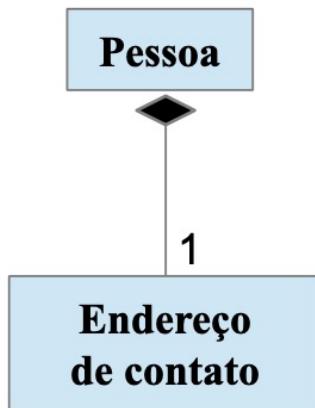


OU

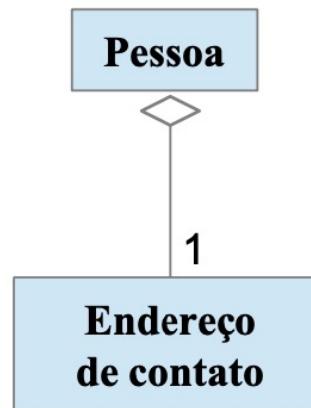


Composição/Agregação

- Quando usar composição ou agregação?
 - Depende dos requisitos do projeto
 - Deve-se interpretar o problema e justificar a escolha

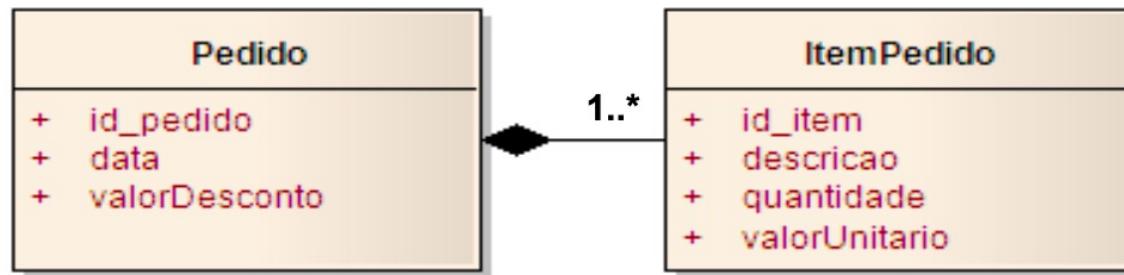


OU



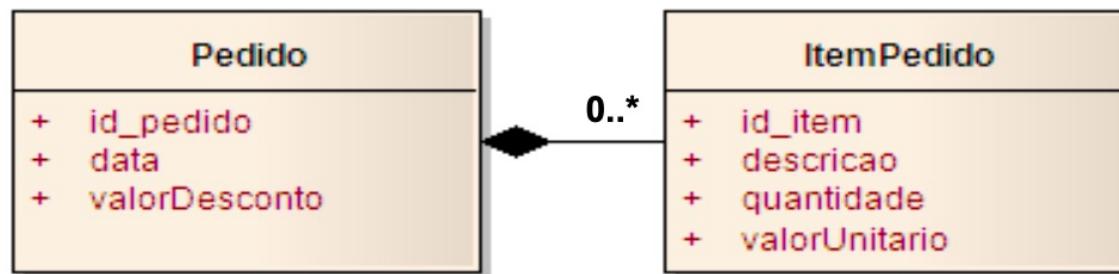
Composição

- Um Pedido é composto por um ou vários ItemPedido
 - Pela cardinalidade, um Pedido precisa ter ao menos um ItemPedido
 - Composição indica que ItemPedido só existe com Pedido



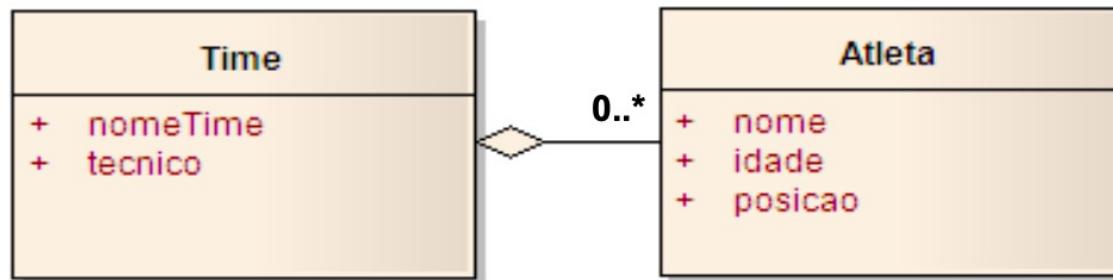
Composição

- Um Pedido é composto por um ou vários ItemPedido
 - Agora, um Pedido pode não ter ItemPedido
 - Contudo, a composição continua indicando que ItemPedido só existe com Pedido



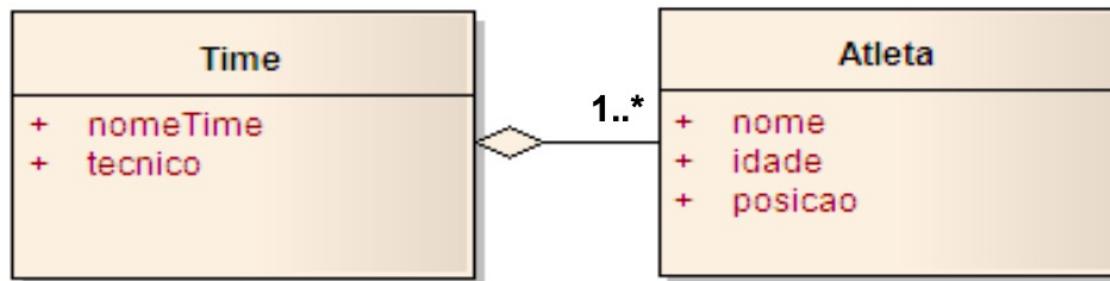
Aggregação

- Um Time é formado por Atletas
 - Pela cardinalidade, um Time pode existir mesmo que não haja Atleta neste time
 - Agregação indica que Atleta existe independente da existência de um Time



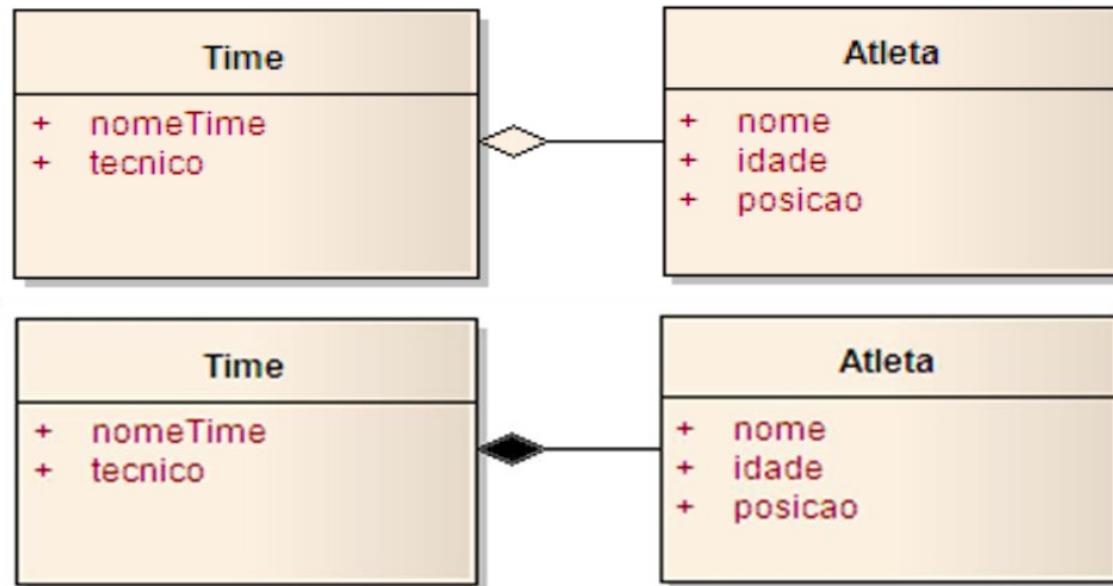
Aggregação

- Um Time é formado por Atletas
 - Agora, um Time precisa conter pelo menos um Atleta
 - Contudo, a agregação continua indicando que Atleta existe independente da existência de um Time



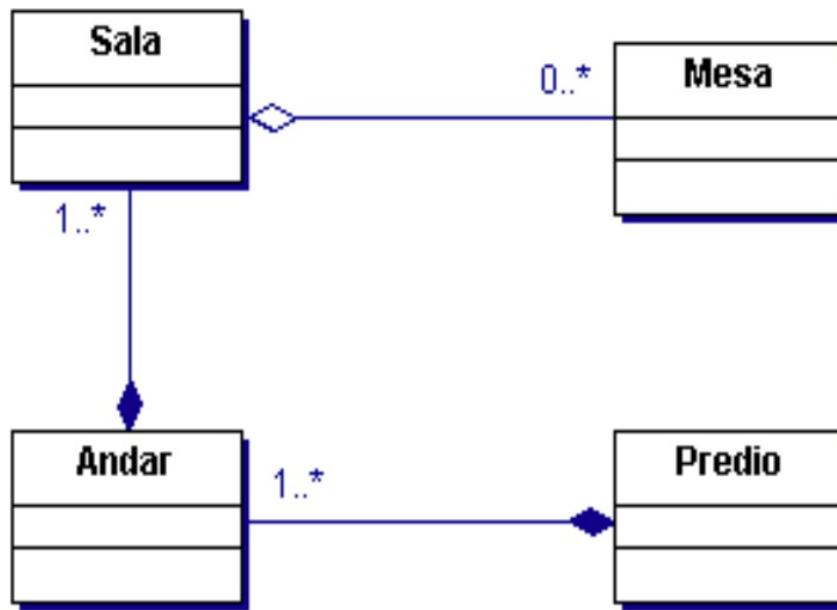
Agregação/Composição

- A semântica (significado) é interpretada pelo projetista
 - Modela o sistema de acordo com sua compreensão e conveniência
 - O mesmo problema pode ser interpretado como composição ou agregação



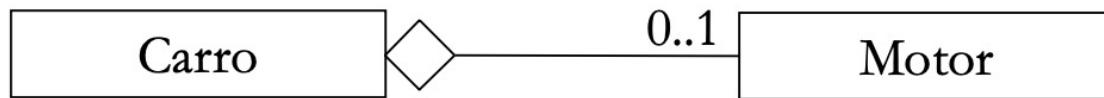
Agregação/Composição

- Outros exemplos



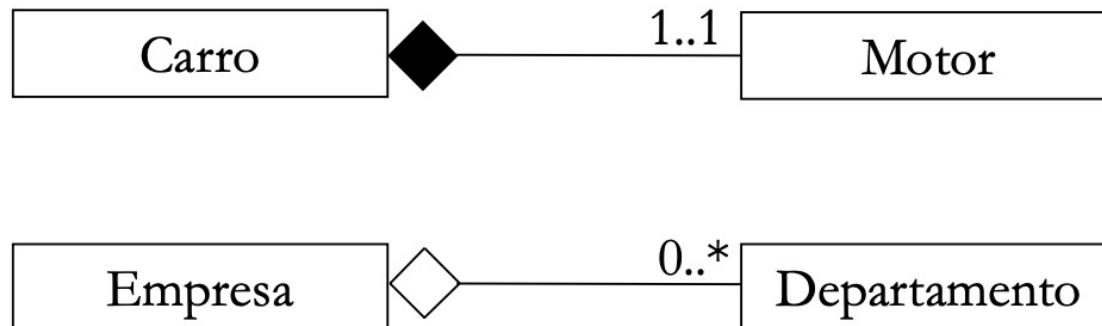
Aggregação/Composição

- Outros exemplos



Agregação/Composição

- Outros exemplos
 - Se invertermos, fica correto?



Aggregação/Composição

- Outros exemplos
 - Se invertermos, fica correto?
 - Sim! O projeto é seu!
 - Desde que satisfaça as características do problema



Agregação/Composição

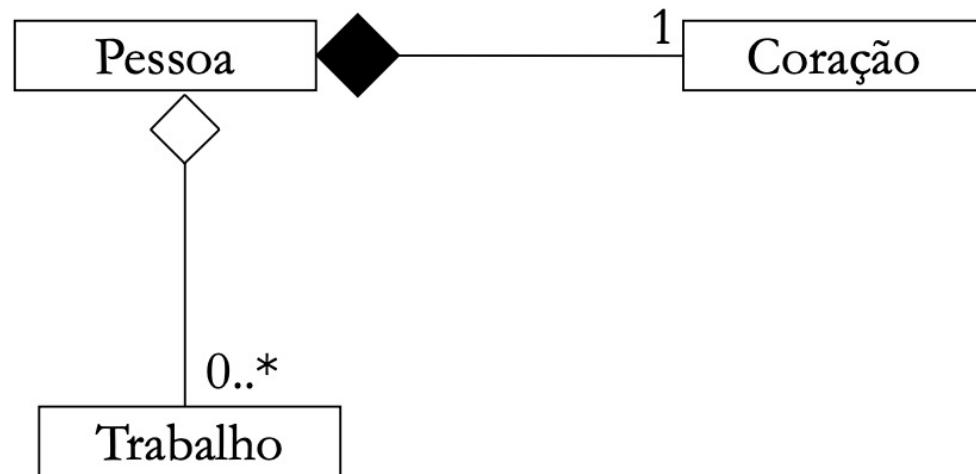
- Outros exemplos
 - Objeto **Pessoa** possui um atributo chamado **Coração**
 - Coração também possui atributos:
 - Conjunto do tipo **Arteria**
 - Conjunto do tipo **Cardiomiocito**
 - FC
 - ...
 - Um **Coração** só faz sentido se estiver vinculado a uma **Pessoa**
 - Atributo de **Pessoa**
 - Então, a relação Pessoa – Coração é uma **composição**

Agregação/Composição

- Outros exemplos
 - Objeto **Pessoa** também pode ter um atributo chamado **Trabalho**
 - **Trabalho** pode ter seus próprios atributos: **Local**, **CNPJ**,
...
 - Em geral, uma **Pessoa** tem um **Trabalho** mas não precisa ter um para existir
 - Além disso, se você pedir demissão, o **Trabalho** que antes era seu não deixará de existir.
 - Assim, a relação Pessoa – Trabalho é uma **agregação**

Agregação/Composição

- Outros exemplos

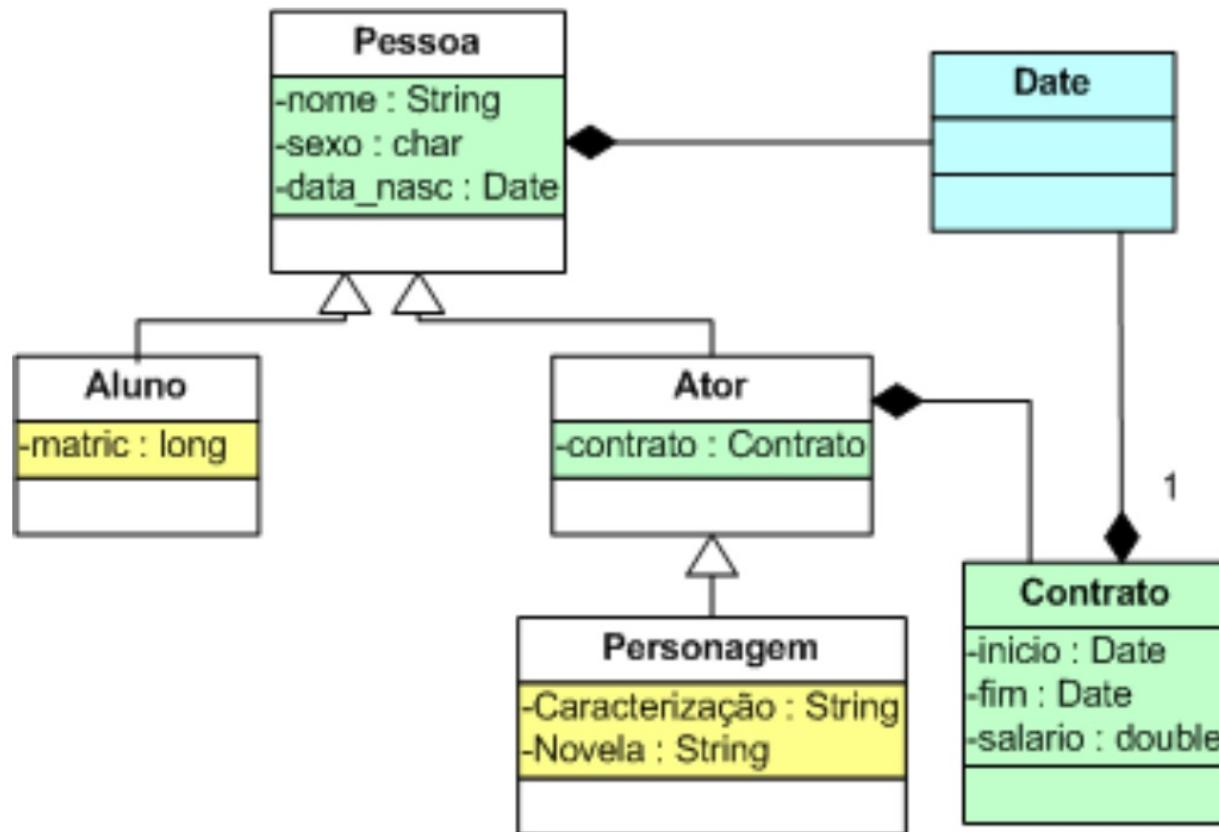


Relacionamentos

- Como saber que tipo de relacionamento deve ser utilizado?
 - Existem atributos ou métodos em comum entre as classes? Ou seja, uma classe “é do tipo” da outra?
 - Sim: Isso é **HERANÇA**
 - Não: Existe relação todo-parte?
 - Não: Isso é uma **ASSOCIAÇÃO SIMPLES**
 - Sim: A parte vive sem o todo?
 - » Sim: Isso é uma **AGREGAÇÃO**
 - » Não: Isso é uma **COMPOSIÇÃO**

Relacionamentos

- Exemplo



Resumo

- Histórico da UML
- Diagrama de classes
- Representação de classes
 - Atributos e métodos
 - Tipos de acesso e modificadores
- Nesta aula foram vistos os principais relacionamentos entre classes
 - Herança, Implementação, Associação, Agregação e Composição

Softwares para UML

- Plugin para o NetBeans
 - Ferramentas -> Plugins -> easyUML
- Outros
 - Rational Rose
 - Yed
 - StarUML
 - Dia
 - Argo UML
 - Microsoft Visio
 - Enterprise Architect

DÚVIDAS?

