

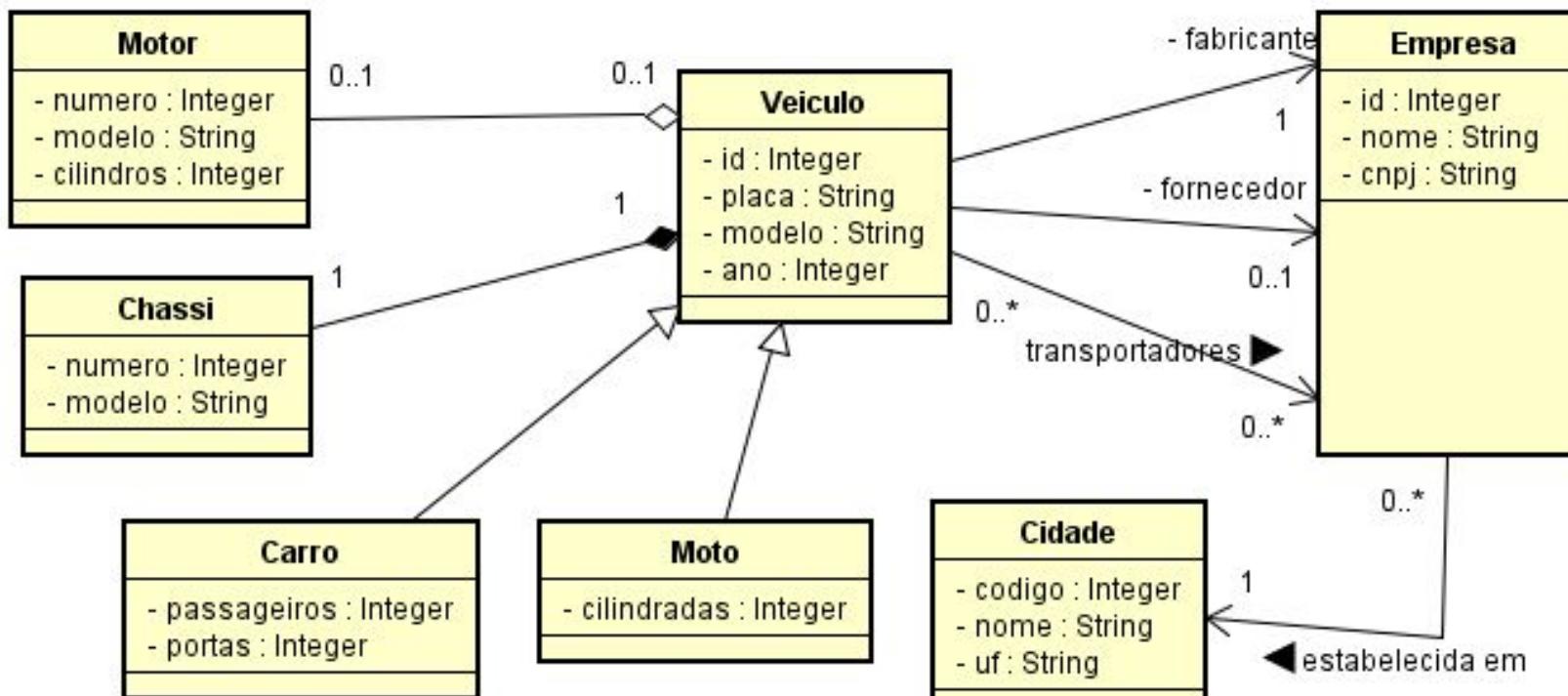
Paradigma Orientado a Objetos: Relacionamentos de entre Classes / Objetos

Prof. Jaqson Dalbosco
jaqson@upf.br

POO - Relacionamentos entre objetos

- Os softwares orientados a objetos geralmente necessitam **controlar diversos tipos diferentes de objetos**, levando a definição de inúmeras classes para sua definição.
- Normalmente estes **objetos precisarão ter relacionamentos** estabelecidos entre eles, de forma semelhante ao que ocorre em um projeto de banco de dados.
- Segue um **exemplo** de como poderia ficar uma definição de classes para **objetos de domínio** para um **estudo de caso simplificado** para um software de um **banco fictício**.

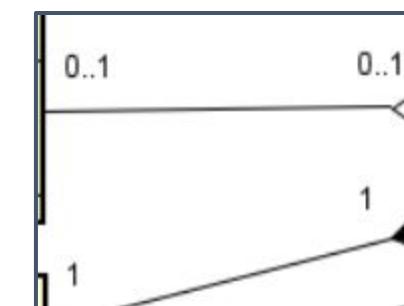
Exemplo fictício de especificação para classes de um estudo de caso



navegabilidade (direção da seta)



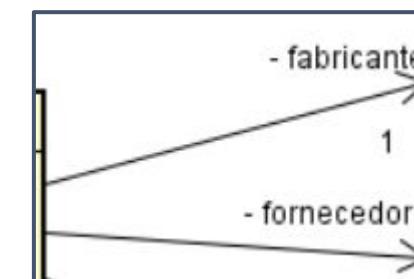
multiplicidade (cardinalidade - 0..1, 1..*, etc)



nome da relação



papel (de um objeto na relação)



Relacionamentos

- ◆ Relacionamentos entre classes / objetos
 - Os **objetos não co-existem isoladamente** em uma aplicação
 - Geralmente **possuem algum tipo de relacionamento**, o que deve ser **especificado nas classes que os definem.**
 - Os relacionamentos podem ser
 - **Herança**
 - Generalização/Especialização
 - **Agregação**
 - **Agregação por Composição**
 - **Associação**

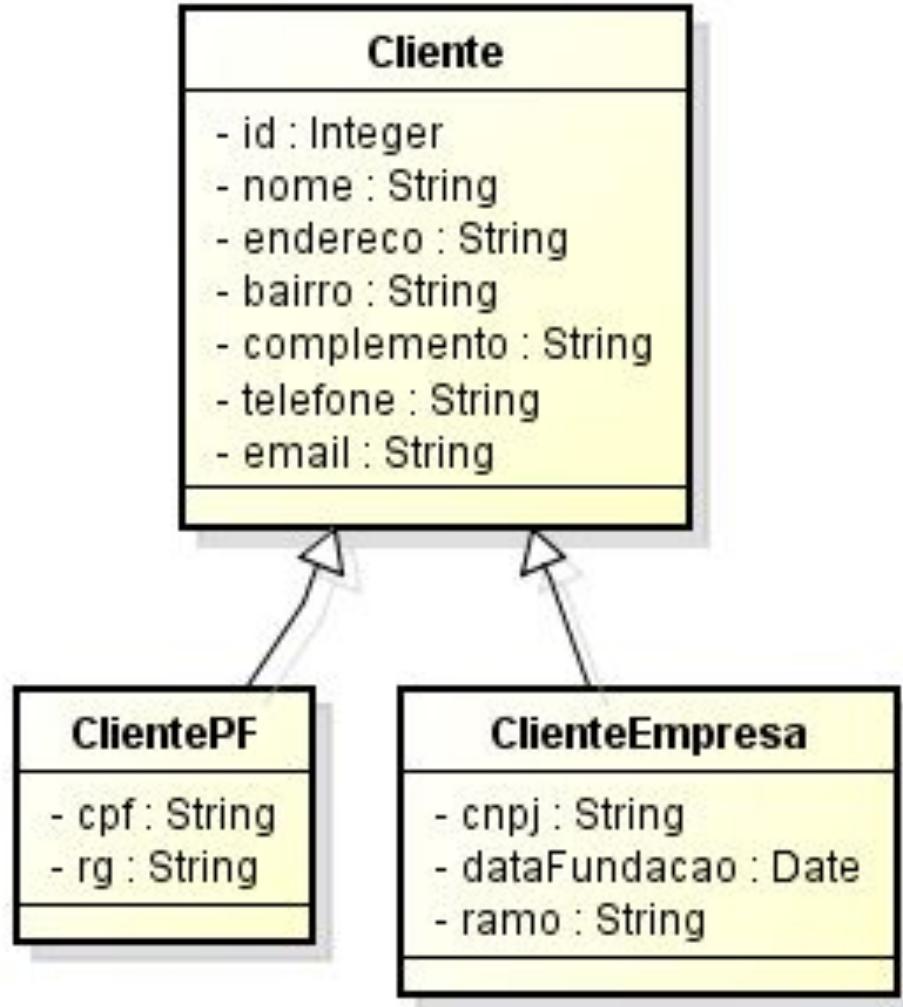
Herança

- Define que **uma classe é uma especialização de outra classe mais genérica**
- A **classe genérica (superclasse)** possui características comuns às outras **classes mais específicas / especializadas (subclasse)**.
- Exemplo
 - As classes Pessoa, Estudante, Funcionário e Professor
 - **Estudante, Funcionário e Professor** são um tipo de **Pessoa**
 - **Pessoa** (generalização)
 - **Estudante, Funcionário e Professor** (especialização, particularização)
- Dica
 - Podemos dizer que a **subclasse específica** “**É um**” ou “**É um tipo de**” uma **superclasse** mais genérica.

Herança (papéis das classes)

- **Superclasse**
 - Representa a generalização
- **Subclasse**
 - Representa a especialização/particularização
- **Exemplo**
 - Relacionamento de herança entre as classes Pessoa e Estudante, Funcionário e Professor
 - Pessoa: **superclasse** de Estudante, Funcionário e Professor
 - Estudante, Funcionário e Professor: **subclasses** de Pessoa

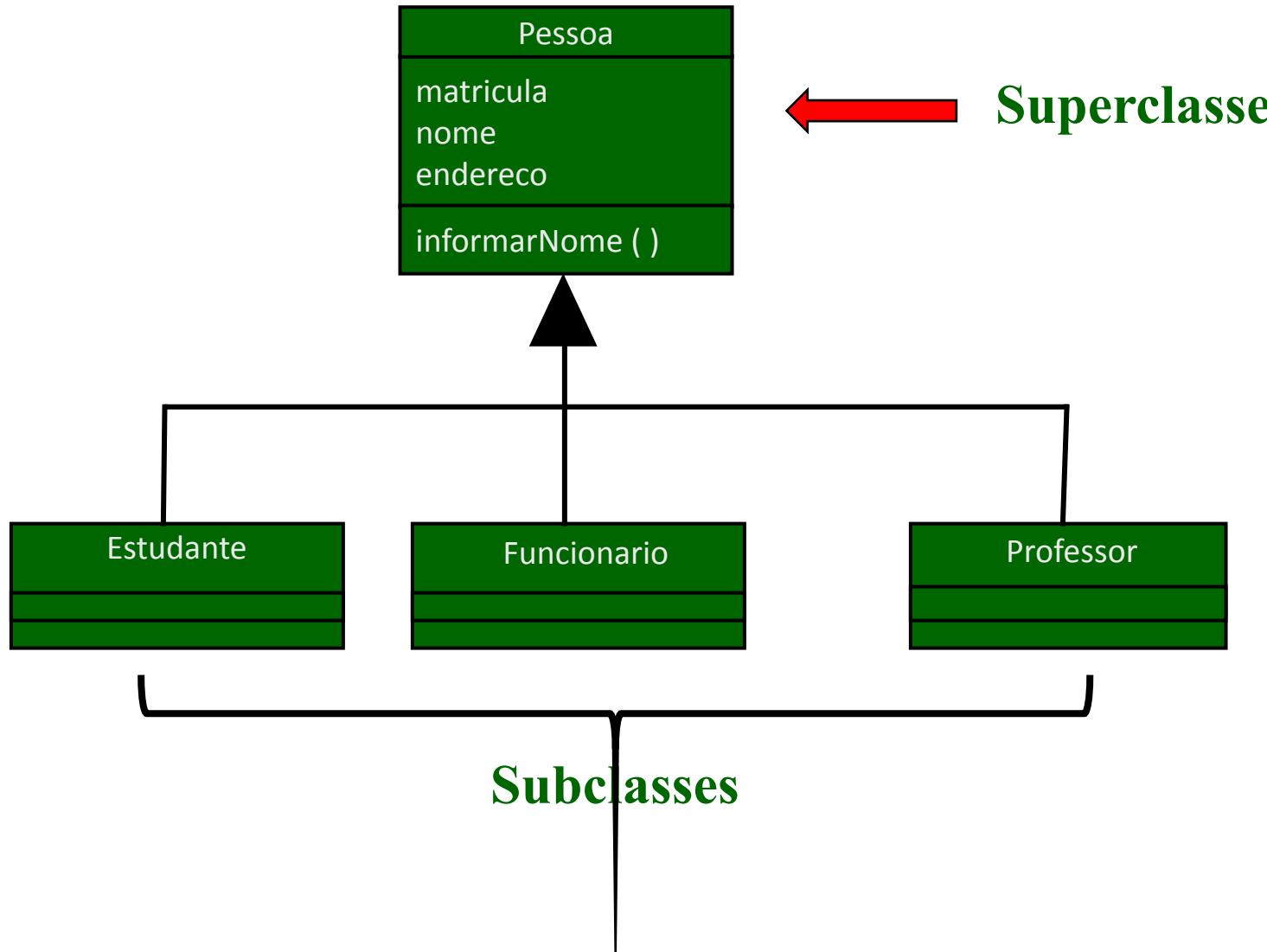
Herança - exemplo



Para a **implementação** de herança no Java utiliza-se a palavra **“extends”** na **definição das subclasses** de forma a declarar a sua extensão de uma superclasse.

Na herança não há navegabilidade e multiplicidade. Sempre uma subclasse herda características de uma superclasse.

Herança (exemplo)



Herança (significado)

- **Atributos e métodos da superclasse são herdados pela subclasse**
 - Pessoa possui o atributo nome
 - Em consequência, estudante, funcionário e professor também possuem o atributo nome
- Todos os atributos e métodos da superclasse são herdados pela subclasse **podendo haver encapsulamento** nas estruturas
- Métodos **construtores não são herdados.**
- A **superclasse implementa atributos e métodos genéricos** que servem para todas as **subclasses e estas, implementam atributos e métodos específicos** ao seu contexto
- Este tipo de relacionamento é **um dos pontos chaves do paradigma** de orientação a objetos, pois contempla a propriedade de **reuso de software**

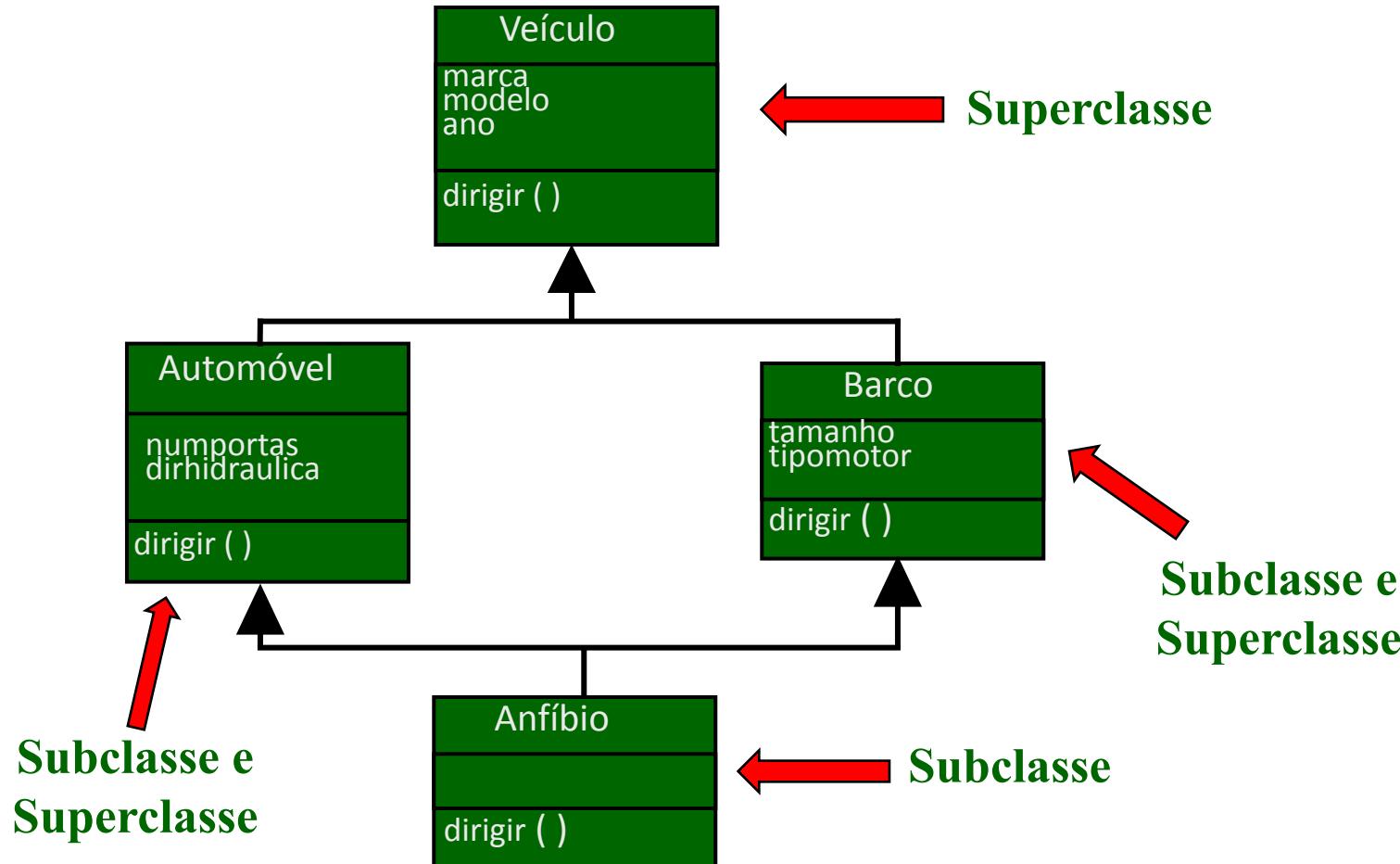
Herança (características)

- **É um relacionamento entre classes e não entre objetos, ou seja**
 - Cada instância de um objeto da subclasse gera uma estrutura composta por atributos e métodos da superclasse acrescidos dos atributos e métodos da subclasse.
 - Quando compila o programa este relacionamento já fica implementado.

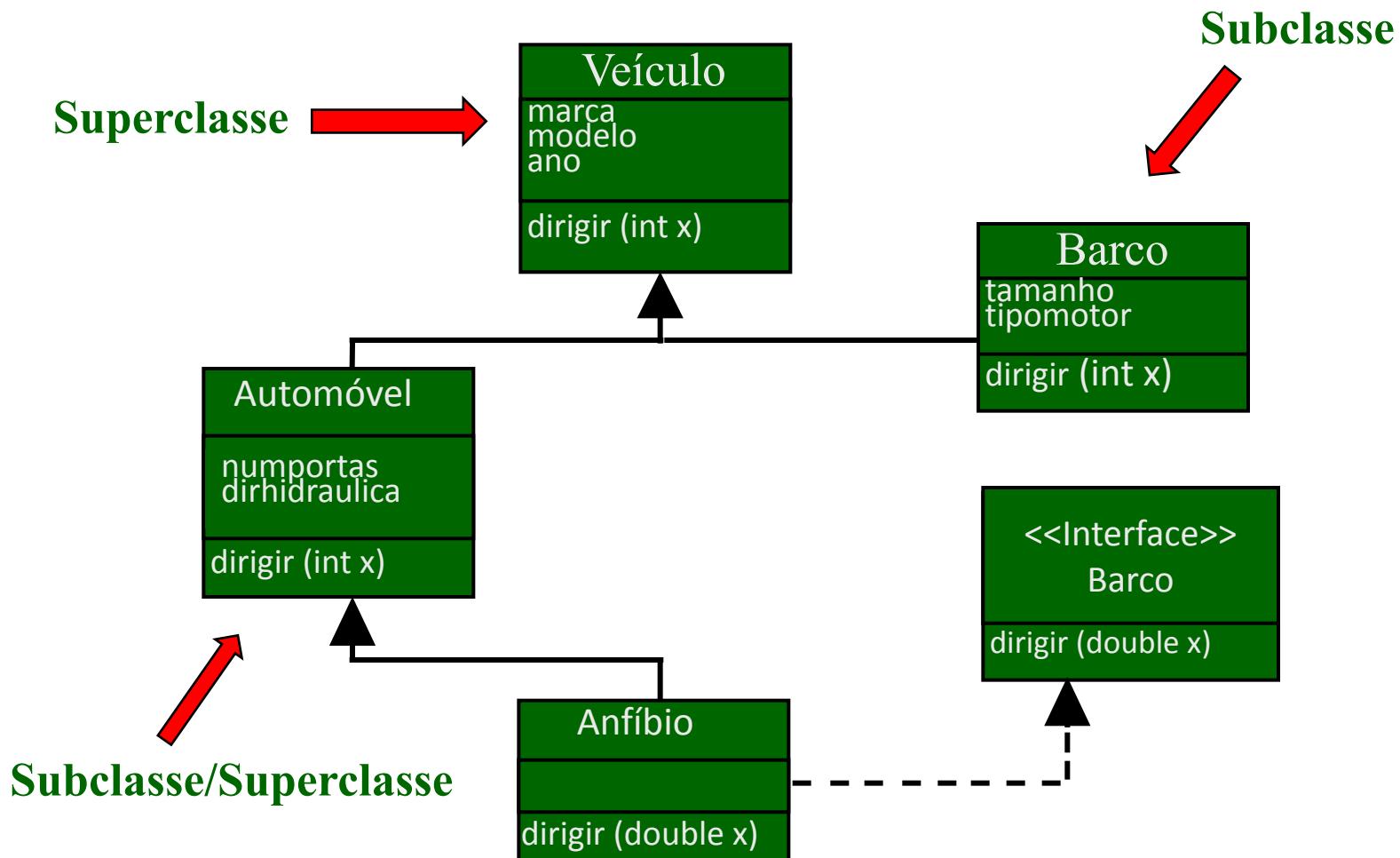
Herança Múltipla

- Quando uma subclasse herda atributos ou métodos de duas ou mais superclasses
- É um tipo de relacionamento que pode apresentar problemas se não for bem utilizado
 - colisão de nomes provindos de atributos das superclasses
 - colisão de métodos provindos das superclasses
- Algumas linguagens implementam outras não
 - Uso de Interfaces como uma forma de herança múltipla a nível mais abstrato

Herança Múltipla (exemplo)

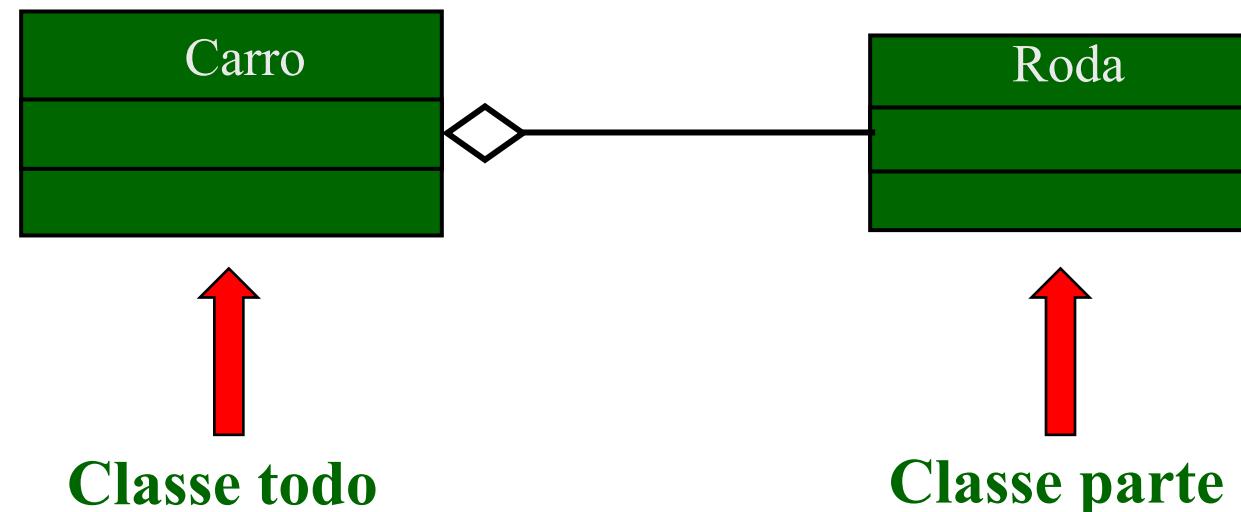


Herança Múltipla (exemplo com Interface)



Agregação

- Relação que pode ocorrer entre duas classes
- **Caracteriza uma relação “todo-parte”**
- Exemplo
 - Relação entre as classes Carro e Roda
 - Um Carro possui rodas



Agregação (papéis das classes)

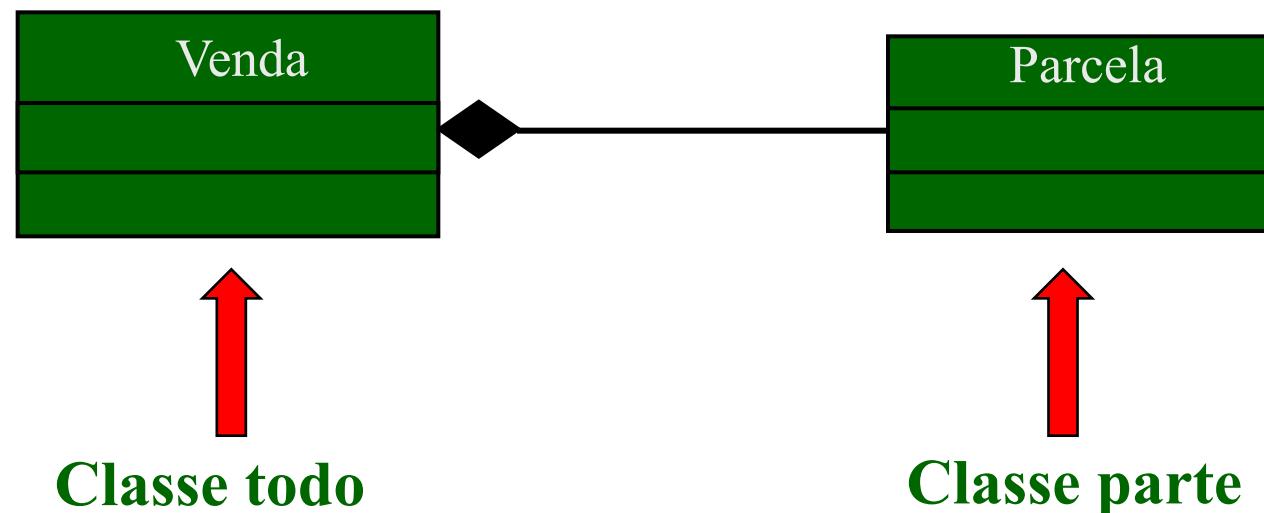
- Classe **todo ou agregada**
 - É a classe resultante da agregação
- Classe **parte**
 - É a classe cujas instâncias formam a agregação
- Exemplo entre as classes
 - Carro e Roda
 - Classe Carro: todo ou agregada
 - Classe Roda: parte

Agregação (características)

- Quando o **todo existe, independentemente da parte** (e vice-versa)
- **Não existe uma “ligação forte” entre as duas classes/objetos**
- Objetos da classe **todo são independentes da classe parte**
- O objeto **“parte” pode ser agregado em diferentes objetos “todo”**
- Agregação (como encontrar)
 - Dica
 - Pense se um objeto “Contém” ou “É parte do” outro
 - Perguntas
 - é uma relação todo-parte?
 - o objeto parte vive sem o todo?

Agregação por Composição

- ◆ Ocorre quando tem-se uma **situação semelhante à da agregação** entre dois objetos
- ◆ Diferencia-se porque os **objetos precisam estar sempre unidos**, de forma **dependente**, ou seja, **existe uma “ligação forte”** entre os dois objetos
 - A “parte” pode compor apenas 1 “todo”



Agregação por Composição (características)

- O “todo” é responsável pela vida de suas partes (criação e destruição)
- A “parte” não tem vida independente do “todo”
- Os objetos da classe parte são dependentes, em termos de vida, da classe todo
- Os objetos da classe parte não podem continuar vivendo quando o todo é destruído
- **Agregação por Composição (como encontrar)**
 - Dica
 - Pense se um objeto “Contém” ou “É parte” do outro
 - Perguntas
 - é uma relação todo-parte?
 - o objeto parte vive sem o todo?
 - Não tem sentido os objetos parte continuarem existindo sem o objeto todo

Agregação x Composição (diferenças)

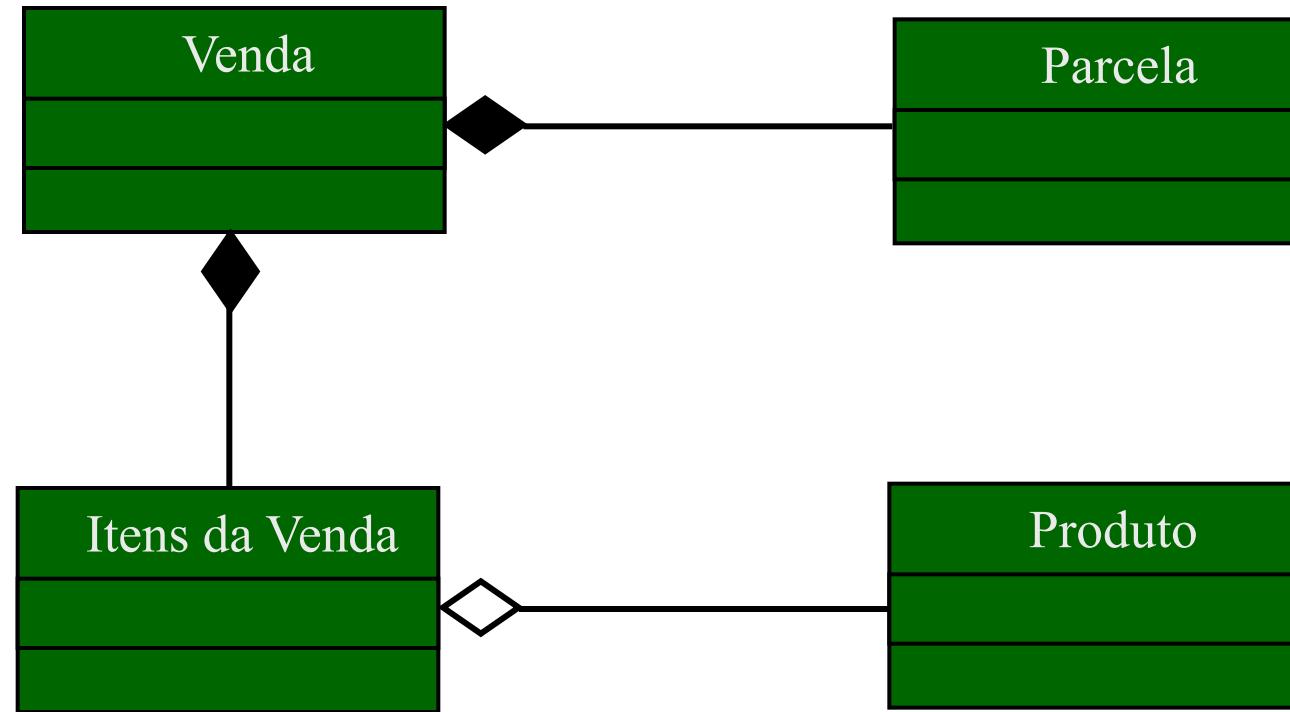
Agregação

- Os objetos “parte” podem existir antes do objeto “todo”
- Os objetos “parte” continuam existindo se o objeto “todo” não mais existir

Composição

- Os objetos “parte” não podem existir antes do objeto “todo”
- Os objetos “parte” não podem existir se o objeto “todo” deixar de existir

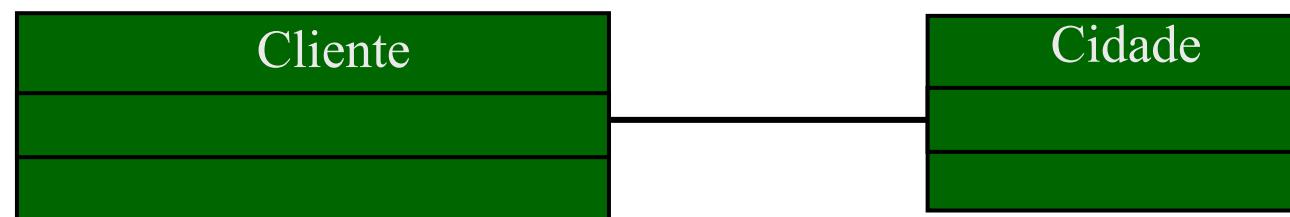
Agregação e Composição (exemplo)



- A venda possui parcelas e itens. Se excluir a venda, são excluídos também seus itens e suas parcelas
- Os itens da venda são formados por produtos, mas se excluir um item o produto continua existindo

Associação

- É uma relacionamento entre classes que **não pode ser caracterizado como herança, nem como agregação e nem como composição**
- **Não apresenta significado preciso**
- Mais **comum em aplicações voltadas para comércio e serviços**
- Exemplos:
 - A cidade onde uma pessoa mora
 - O grupo de um produto
 - A marca de um equipamento



Associação Recursiva

- É possível relacionar uma classe a ela mesma.
- Representa semanticamente a conexão entre dois objetos da mesma classe.



- Neste exemplo, o atributo cônjuge será criado na classe Pessoa que poderá conter a instância de outra pessoa que representará o seu cônjuge.

Identificando o tipo de relacionamento

- ◆ Como saber qual relacionamento deve ser utilizado?
 - Sempre deve ser estabelecida a **reflexão entre duas classes** candidatas ao relacionamento.
 - Primeiro pense se **existe herança?**
 - Existem atributos ou métodos comuns entre as classes? A subclasse “é do tipo” da superclasse?
 - **Se Sim:** é **herança**
 - **Se Não:** Pense se **existe todo-parte?**
 - **Se Sim:** Pense se a parte vive sem o todo?
 - **Se Sim:** é **agregação**
 - **Se Não:** é **composição**
 - **Se Não:** é **associação**