Herança

Professor: Henrique Delegrego

Herança

Classes sem herança

```java
public class Carro {
    String marca;
    String modelo;
    int qtdPassageiros;
    int ano;
    double velMax;

    int numPortas;
    double tamPortaMalas;
}
```

```java
public class Aviao {
    String marca;
    String modelo;
    int qtdPassageiros;
    int ano;
    double velMax;

    int qtdMotores;
    int qtdPilotos;
}
```

```java
public class Bicicleta {
    String marca;
    String modelo;
    int qtdPassageiros;
    int ano;
    double velMax;

    boolean freio;
}
```

Herança

Exemplo de herança

```
2 public class Veiculo {
3     String marca;
4     String modelo;
5     int qtdPassageiros;
6     int ano;
7     double velMax;
8 }
```

```
2 public class Aviao extends Veiculo {
3     int qtdMotores;
4     int qtdPilotos;
5 }
```

- Veiculo é pai de Aviao
- Aviao é filho de Veiculo
- Veiculo é uma **super classe**
- Aviao é uma **sub classe**

Agora:

- Vamos criar nossa herança

Herança

Construtores em herança

- As classes filhas não herdam os construtores da classe pai
- O construtor da super classe é chamado a partir do **super()**, essa deve ser a primeira instrução do construtor da subclasse

```java
// Classe pai
public class Veiculo {
    String marca;
    String modelo;
    int qtdPassageiros;
    int ano;
    double velMax;

    public Veiculo(String marca, String modelo, int qtdPassageiros, int ano, double velMax) {
        this.marca = marca;
        this.modelo = modelo;
        this.qtdPassageiros = qtdPassageiros;
        this.ano = ano;
        this.velMax = velMax;
    }
```

```java
// Classe filho
public class Carro extends Veiculo {

    int numPortas;
    double tamPortaMalas;

    public Carro(String marca, String modelo, int qtdPassageiros, int ano, double velMax, int numPortas,
            double tamPortaMalas) {
        super(marca, modelo, qtdPassageiros, ano, velMax); // Chama o construtor da classe pai
        this.numPortas = numPortas;
        this.tamPortaMalas = tamPortaMalas;
    }
```

Herança

Métodos em herança

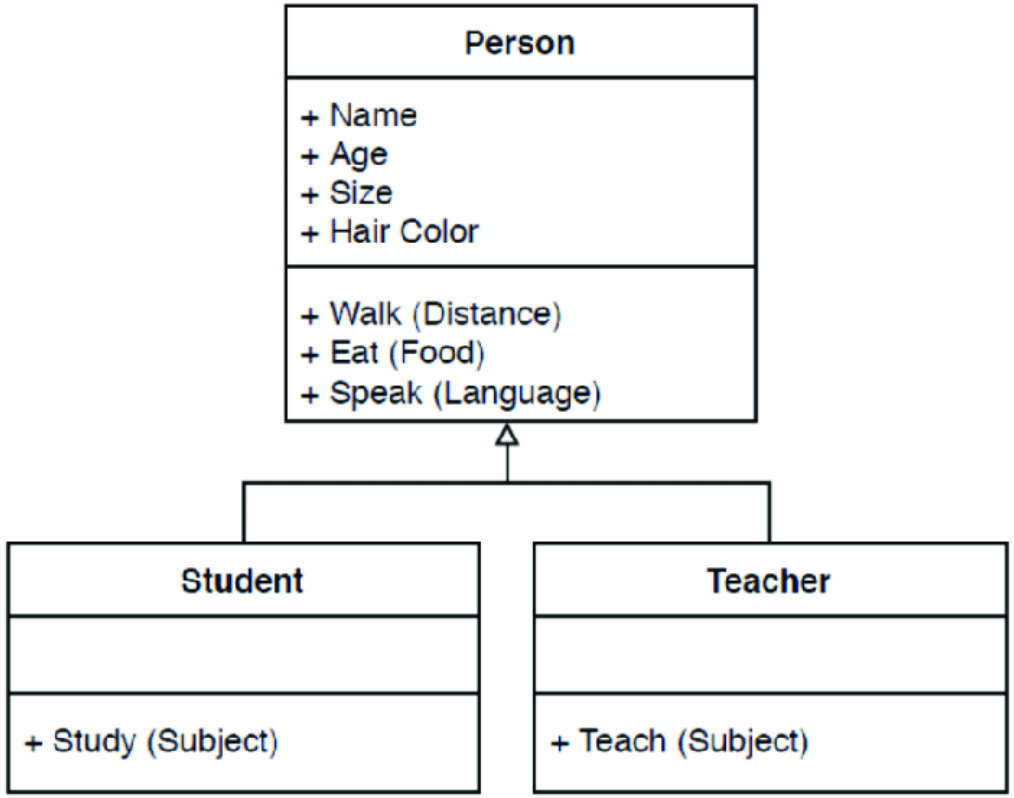
- Métodos são herdados, se um método é definido em uma super classe, ele é automaticamente herdado pela sub classe
- Se quiser acessar a versão um método da classe pai na classe filho, pode ser (opcionalmente) especificado usando **super.nomeMetodo()**

```java
// Na classe Veiculo (pai)
public String acelerar() {
    return "Estou acelerando";
}

// Na classe Carro (filho de Veiculo)
public String abrirPorta() {
    return "Estou abrindo uma porta";
}

public static void main(String[] args) {

    Carro c = new Carro("Honda", "Civic", 5, 2010, 200, 5, 400);
    Aviao a = new Aviao("Airbus", "A370", 400, 2010, 600, 4, 2);

    System.out.println(c.acelerar());
    System.out.println(a.acelerar());
    System.out.println(c.abrirPorta());
    System.out.println(a.abrirPorta());
    // Abrir porta é um método exclusivo de Carro
    // Não pode ser chamado por classes "irmãos"

}
```

Herança

Diagrama

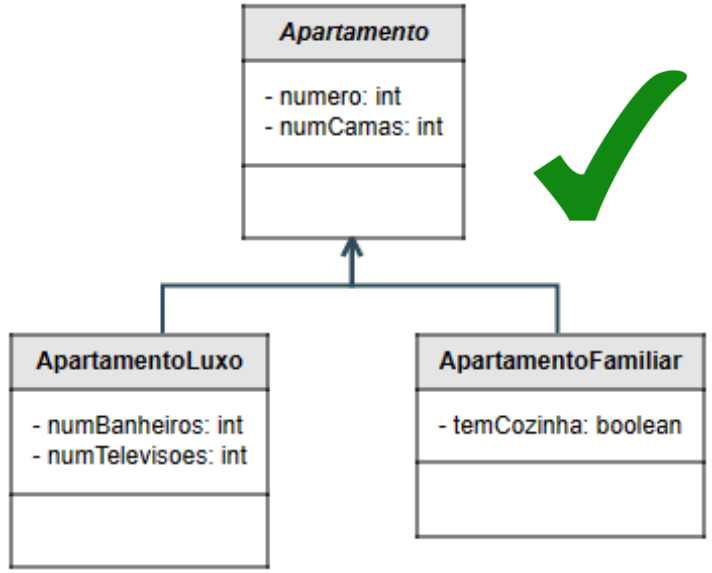
- A herança na UML é representada por uma linha contínua com um triângulo apontando da sub classe para a super classe
- Atributos e métodos da super classe não são mencionados nas sub classes

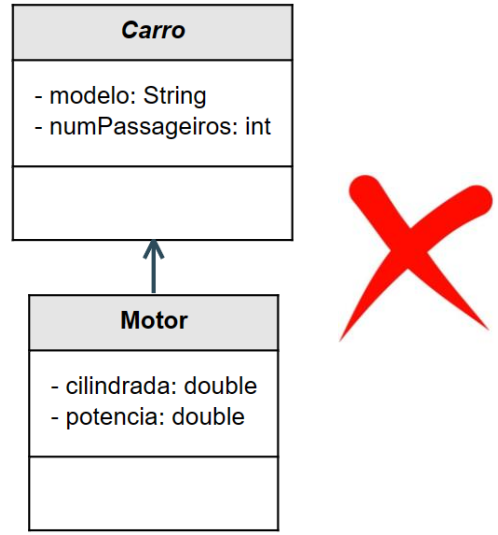


Herança

Como identificar uma herança?

- Para identificar uma herança você deve se perguntar:
  - A classe que eu classifiquei como filho **é um** tipo do pai?
  - Há comportamentos ou atributos em comum?





Agora:

- Fazer exercício 1 e 2

Herança

Classes abstratas

- Classe que não pode ser instanciada
- Serve como classe base para outras classes que vão herdá-la
- Podem conter:
  - Métodos abstratos: sem corpo, apenas a assinatura
  - Métodos normais (concretos): com implementação completa
- A classe abstrata será identificada pela palavra chave **abstract**

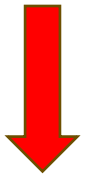
```
public abstract class MinhaClasse
```

Herança

Métodos abstratos

- Podem ser criados somente nas classes abstratas
- Métodos abstratos não terão corpo na classe abstrata
- Obrigatoriamente terão que ser implementados (sobrescritos) nas classes filhos
- Não podem ser **private, final ou static**

```
public abstract String meuMetodo();
```



```
@Override
public String meuMetodo() {
    // Na classe filho
    return "Esse é o método da classe filho";
}
```

Herança

Como identificar um método abstrato?

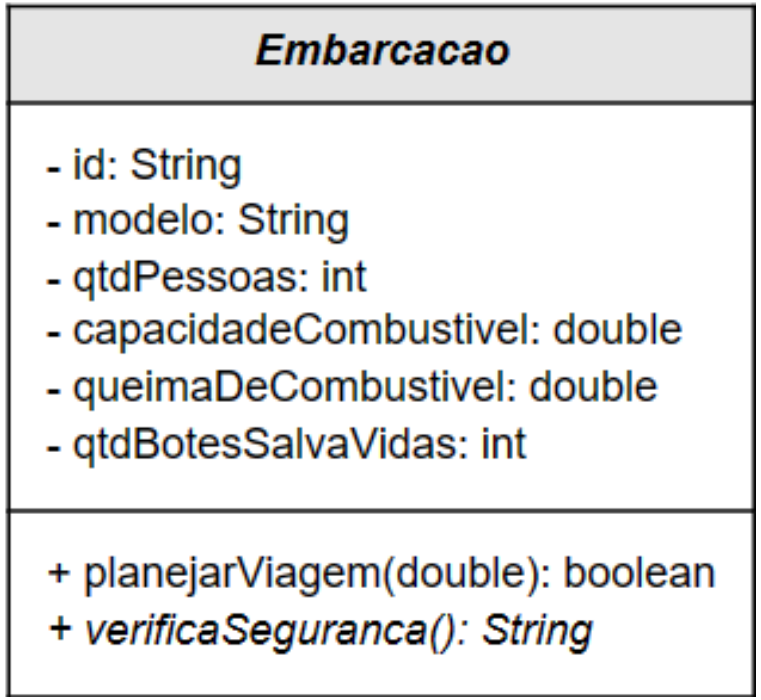
- Para identificar um método abstrato no seu código você deve se perguntar:
  - A implementação é diferente dependendo da classe?
  - A lógica do método depende totalmente da subclasse?
- Caso sim, então o método é abstrato

Polimorfismo

- Permite que um objeto seja tratado como uma instância da sua **superclasse**
- Ocorre quando uma subclasse reimplementa um método da superclasse com a mesma assinatura, sobrescrevendo o método

Herança

Diagrama de classes abstratas



- Classes e métodos abstratos são identificados pela sua notação em itálico

Agora:

- Terminar a lista de exercícios