+Devs2Blu

Linguagem de programação JAVA

Profª. Heloisa Moura

+Devs2Blu

Linguagem de programação JAVA

O que vamos ver:

- Conceito de Herança
- Polimorfismo
- Classe abstrata

Herança

É a capacidade de uma classe herdar propriedades e comportamentos de outra classe (superclasse), possibilitando o reaproveitamento de código.

+Devs2Blu

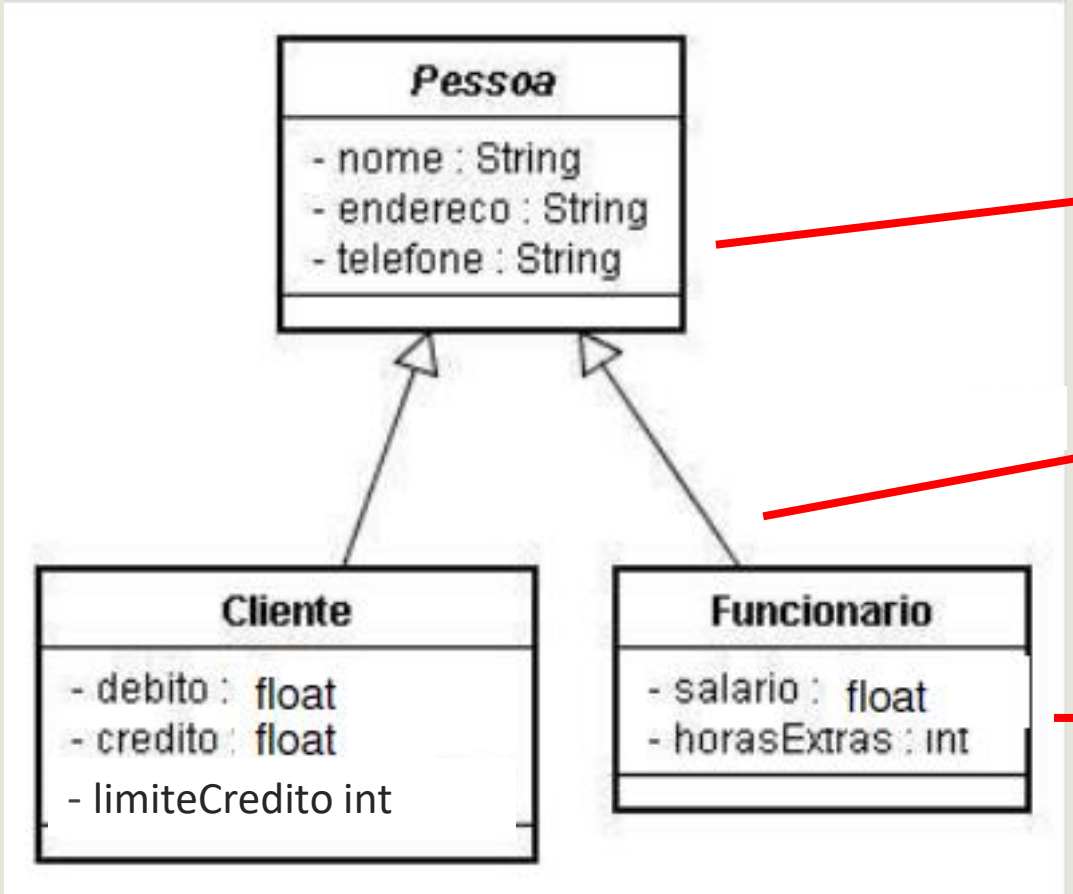
Linguagem de programação JAVA - Herança

- É possível criar uma hierarquia de classes, definindo classes mais gerais e classes mais específicas.
- Uma sub-classe (+específica) herda métodos e atributos de sua super-classe (+geral);
- A sub-classe pode reescrever métodos, dando uma forma mais específica para um método herdado.

+Devs2Blu

Linguagem de programação JAVA - Herança

Exemplo em UML



Super-classe:
Generalização ou abstração

Herança:
relacionamento “é um”

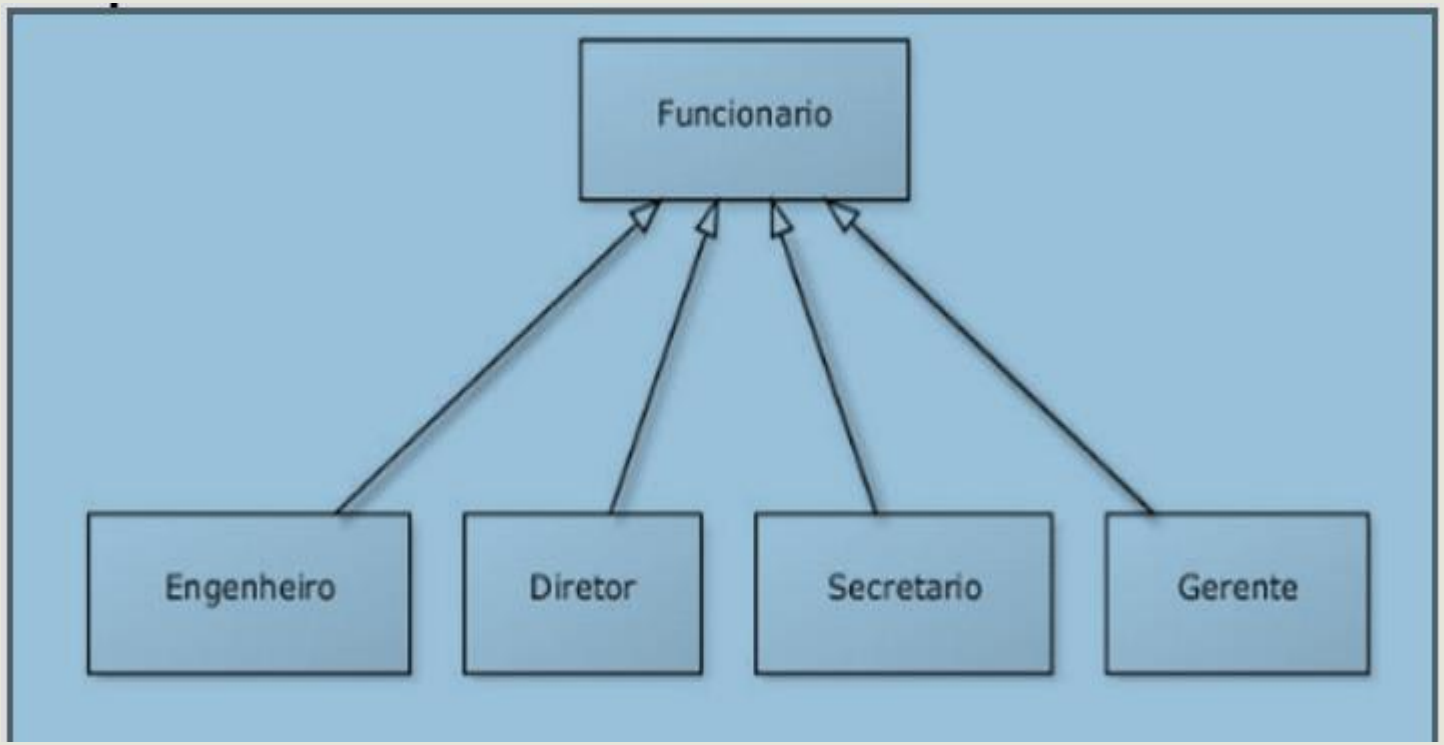
Sub-classe:
Especializaçao

+Devs2Blu

Linguagem de programação JAVA - Herança

Além disto, pode-se estender o projeto com maior facilidade no futuro.

Exemplo:



+Devs2Blu

Linguagem de programação JAVA - Herança

Exemplo Pratico
exerciciospratica.Heranca

Construtor superclasse/subclasse

- Um fato importante: os construtores são algo único a cada classe, portanto não são herdados;
- A primeira coisa que um construtor faz é rodar o construtor de sua superclasse, pois ações importantes podem estar acontecendo lá - como inicialização de variáveis que poderão ser usadas na subclasse;
- É possível invocar os construtores de uma superclasse através da subclasse

Exemplo prático
exerciciospratica.Heranca.construtores.superclassesubclasses

Palavra reservada super

Uso do **this** – referência a elementos da classe (atributos/métodos)

.

Uso do **super** – referência a elementos da super-classe.

```
class Pessoa {

    String nome;
    String endereco;
    String telefone;

  void mostrar() {

     System.out.println("Nome:" + this.nome);
     System.out.println("Endereço: " + this.endereco);
     System.out.println("Telefone: " + this.telefone);
  }

}
```

+Devs2Blu

Linguagem de programação JAVA

```
class Funcionario extends Pessoa{
                    private  float salario;
                    private  int horasExtras;
                   void mostrarInfo(){
                            System.out.println("Nome:" + super.nome);
                            System.out.println("Endereco: " + super.endereco);
                            System.out.println("Telefone: " + super.telefone);
                            System.out.println("Salario: " + this.salario);
                            System.out.println("H.extras: " + this.horasExtras);
                   }
          }
```

+Devs2Blu

Linguagem de programação JAVA - palavra reservada super

Uma solução melhor...

```
class Funcionario extends Pessoa{
    private float salario;
    private int horasExtras;
    void mostrar(){
        super.mostrar();
        System.out.println("Salario: " + this.salario);
        System.out.println("H.extras: " + this.horasExtras);
    }
}
```

+Devs2Blu

Linguagem de programação JAVA - palavra reservada super

Exemplo Prático
exerciciospratica.Heranca.palavrasuper

Modificador de acesso protected

- Fica entre o **private** e o **public** . Um atributo **protected** só pode ser acessado (visível) pela própria classe, suas subclasses e classes encontradas no mesmo pacote;
- Modificador usado em Herança, por sua visibilidade em subclasses.

Sintaxe:

protected String testeVisibilidade;’

+Devs2Blu

Linguagem de programação JAVA - modificador de acesso protected

Exemplo Pratico

exerciciospratica.Heranca.modificadorprotected

Polimorfismo (sobrescrita/overriding) de métodos

- Quando uma classe é estendida, a subclasse herda todos os métodos "não-privados" da classe mãe. Algumas vezes, é desejável modificar o comportamento de um desses métodos na nova classe.
- Para fazer isso basta reescrever o método da classe mãe na classe filha, utilizando o mesmo nome e a mesma lista de parâmetros (com algumas restrições quanto ao tipo de retorno, uso de modificadores de acesso e utilização de exceções).
- Nesse caso, dizemos que o método está sendo sobrescrito. Sendo alterado para atender o comportamento daquele objeto em questão na mesma operação herdada.
- A isso damos o nome de **polimorfismo**, ou seja, várias formas de implementar a mesma operação.

Tipos de Polimorfismo

Polimorfismo de Sobrecarga (Overloading): Várias versões de métodos com o mesmo nome, mas assinaturas diferentes (número ou tipos de parâmetros). Se aplica também a construtores.

Polimorfismo de Sobrescrita (Overriding): Métodos que têm o mesmo nome e assinatura na classe pai e na classe filha, mas com comportamento diferente.

Regras da sobrescrita de métodos:

- Deve possuir a mesma assinatura do método(mesmo nome, mesma lista de parâmetros e mesmo tipo de retorno);
- Não pode possuir um modificador de acesso mais restritivo:
- Apenas métodos herdados podem ser sobrescritos (é necessário um relacionamento É UM).
- O método na classe pai não pode ser final (método final não pode ser sobrescrito)
- O método na classe pai não pode ser static(método static pertencem à classe, não à instância, e não podem ser sobrescrito)

Regras da sobrescrita de métodos:

- O método sobrescrito pode ter um **tipo de retorno mais específico** (subtipo) em relação ao tipo de retorno do método original (tipo covariante)

Por exemplo, se o método original retorna um objeto da classe **Animal** a versão sobrescrita pode retornar um objeto de uma classe que herda de **Animal** como **Cachorro**.

Regras da sobrescrita de métodos:

- Não pode lançar exceções mais amplas:

Por exemplo, se o método na classe pai lança uma exceção **IOException** o método sobrescrito não pode lançar **Exception**(que é mais genérica). Ele só pode lançar exceções do mesmo tipo ou mais específicas, como **FileNotFoundException**

+Devs2Blu

Linguagem de programação JAVA

Exemplo prático
exerciciospratica.Heranca.polimorfismo

+Devs2Blu

Linguagem de programação JAVA - Herança

Exercício 1: Funcionário e Assistente

- Implemente a Classe Funcionario com os atributos:
  nome e salario
  Implemente com funcionalidades os métodos:
  addAumento(double valor),
  ganhoAnual()
  exibeDados().
- Crie a classe Assistente, que herda de Funcionario e possui um número de matrícula. Sobrescreva o método exibeDados().
- Crie também as classes Tecnico e Administrativo, que herdam de Assistente.
- Crie a classe de teste.

Classes abstratas

- Classe abstrata é aquela que não criamos objetos dela. São classes muito genéricas. Isso quer dizer que criamos objetos de suas subclasses concretas.
- Ao colocar a palavra chave **abstract** na declaração da classe, não podemos mais instancia-la.
- Se ela não pode ser instanciada, para que serve? Serve para o polimorfismo e herança dos atributos e métodos, que são recursos muito poderosos. A palavra **abstract** não pode ser aplicada a atributos.

+Devs2Blu

Linguagem de programação JAVA - Herança

Exemplo pratico
exerciciospratica.Heranca.classesabstratas

Pessoa
Cliente
Funcionario

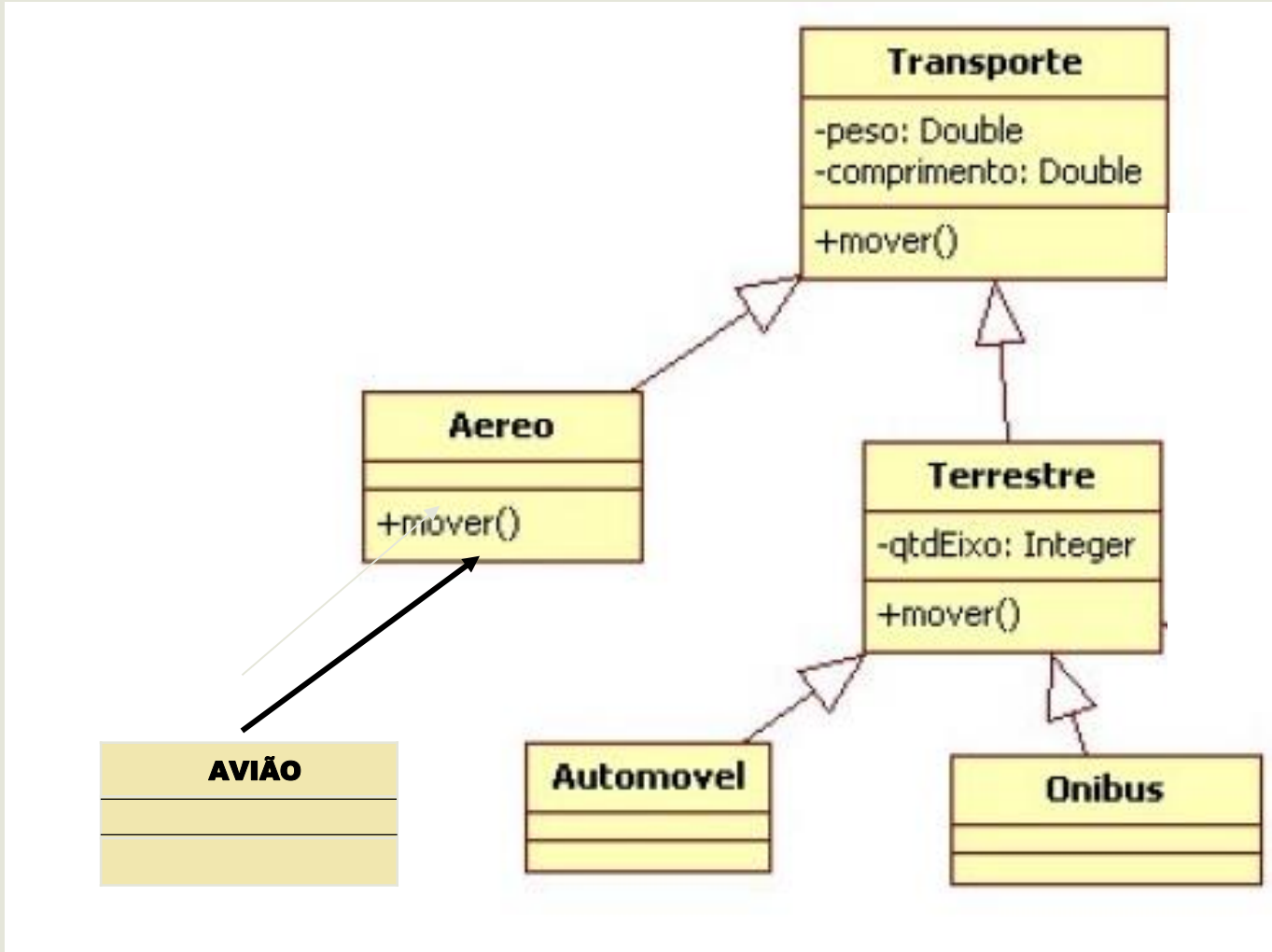
Exemplo pratico1

exerciciospratica.Heranca.classesabstratas.exemploum

```
public abstract class Transporte{
}
```

+Devs2Blu

Linguagem de programação JAVA - Herança



Palavra reservada final

- A palavra reservada que aplicada na classe, não permite estende-la.
- Nos métodos impede que os mesmos sejam sobrescrito (overriding) na subclasse.
- Nos valores de atributos, não permite que os mesmos sejam alterados, depois que já tenha sido atribuído um valor.
- A finalidade de se ter uma classe **final** é pra que a mesma seja preservada. Seu comportamento não seja alterado.
- Um exemplo de classes **final** é a Classe **String** e a Classe **Math** do pacote **java.lang**. Classes essas bastante utilizada. Imagine se qualquer outra classe pudesse mudar seus comportamentos.

O que aconteceria???

Palavra reservada final

- Os atributos marcados com **final** se tornam variáveis CONSTANTES dentro da classe e não pode ter seus valores alterados. Por isso do nome constante.
- As constantes por convenção do Java são declaradas com letras maiúsculas. É comum ter o modificador static em sua declaração para facilitar o seu acesso direto, através do nome da classe.

Exemplo de constante da Classe Math

```
public static final double PI = 3.14159265358979323846;
```

Exemplo prático

exerciciospratica.Heranca.classesfinal

+Devs2Blu

Linguagem de programação JAVA - Herança

EXERCÍCIOS