

Disciplina Regular 1

Fundamentos do

Desenvolvimento Java

Graduação em Engenharia de Software - 2020

Etapa 3 Aula 2

Herança e Polimorfismo

Herança

Classe Object

- A classe Object é a superclasse de todas as classes do Java e é definida em `java.lang`.
- Esta herança é implícita, não precisa ser especificada no código.
- Toda classe herda os métodos definidos pela classe Object.
- O método `toString`, por exemplo, é chamado sempre que uma conversão para `String` for necessária.

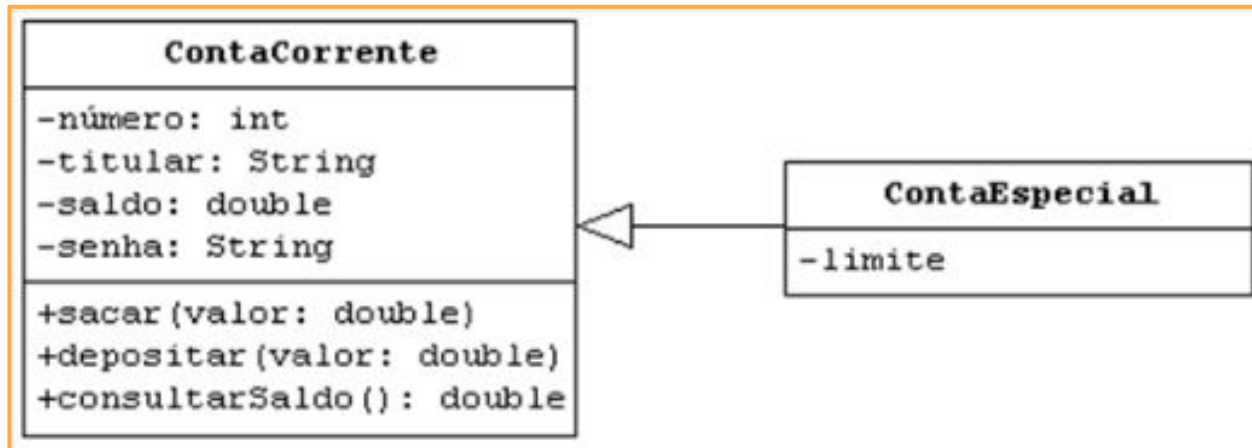
Conceito de Herança

- É a possibilidade de se criar classes a partir de outras já existentes.
- **As classes derivadas contêm todos os atributos e métodos da classe base.**
- Para criar uma classe a partir de outra usa-se a palavra reservada **extends**

```
class SuperClasse {}  
class SubClasse extends SuperClasse {}
```

Conceito de Herança

- A Conta Especial **é uma** Conta-Corrente com um limite adicional.

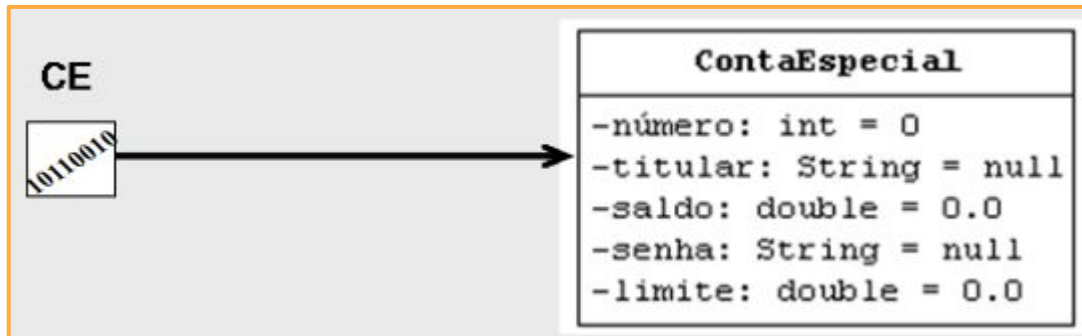


Conceito de Herança

- Um objeto criado a partir de uma subclasse possuirá todos os atributos tanto da subclasse quanto da superclasse.
- Portanto, a criação do objeto **ce**:

```
ContaEspecial ce = new ContaEspecial();
```

- Ficaria assim na memória:



Hierarquia de Construtores

- O comando:

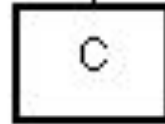
```
ContaEspecial ce = new ContaEspecial();
```

- Chama o construtor padrão da classe.
- O construtor da classe, por sua vez, chama o construtor padrão da superclasse e assim sucessivamente até alcançar o construtor padrão da classe Object.

Hierarquia de Construtores

A.java - Bloco de notas

```
Arquivo  Editar  Formatar  Exibir  Ajuda
public class A {
    public A() {
        system.out.println("construtor de A");
    }
}
```



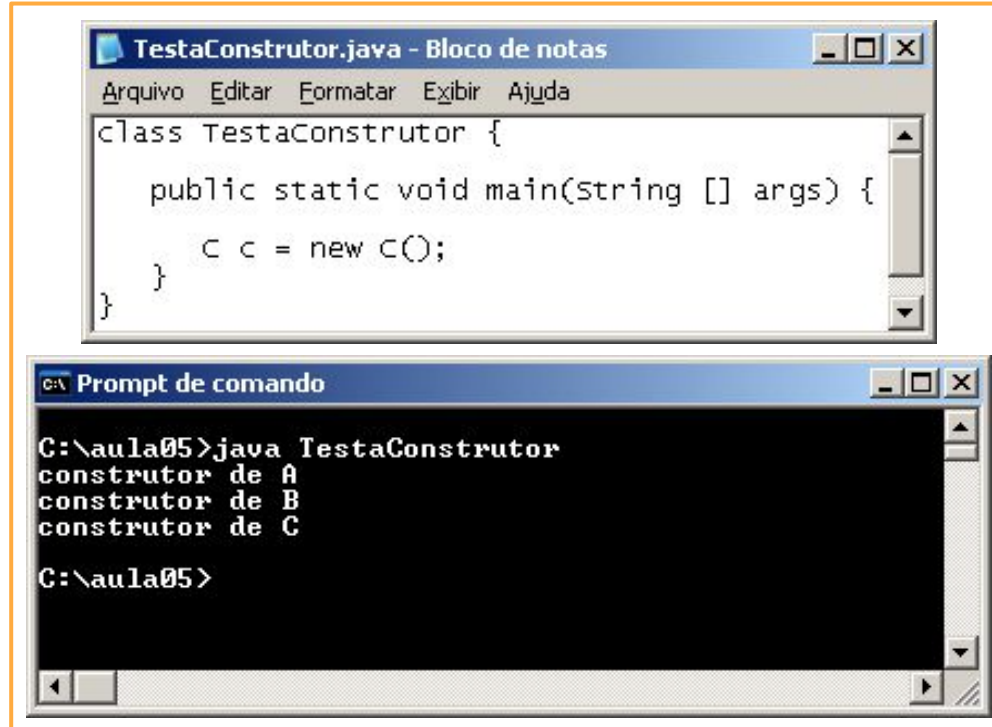
B.java - Bloco de notas

```
Arquivo  Editar  Formatar  Exibir  Ajuda
public class B extends A {
    public B() {
        system.out.println("construtor de B");
    }
}
```

C.java - Bloco de notas

```
Arquivo  Editar  Formatar  Exibir  Ajuda
public class C extends B {
    public C() {
        system.out.println("construtor de C");
    }
}
```

Hierarquia de Construtores



The image shows two windows from a Java development environment. The top window, titled 'TestaConstrutor.java - Bloco de notas', contains the following Java code:

```
class TestaConstrutor {  
    public static void main(String [] args) {  
        C c = new C();  
    }  
}
```

The bottom window, titled 'Prompt de comando', shows the output of running the program:

```
C:\aula05>java TestaConstrutor  
construtor de A  
construtor de B  
construtor de C  
  
C:\aula05>
```

Referência super

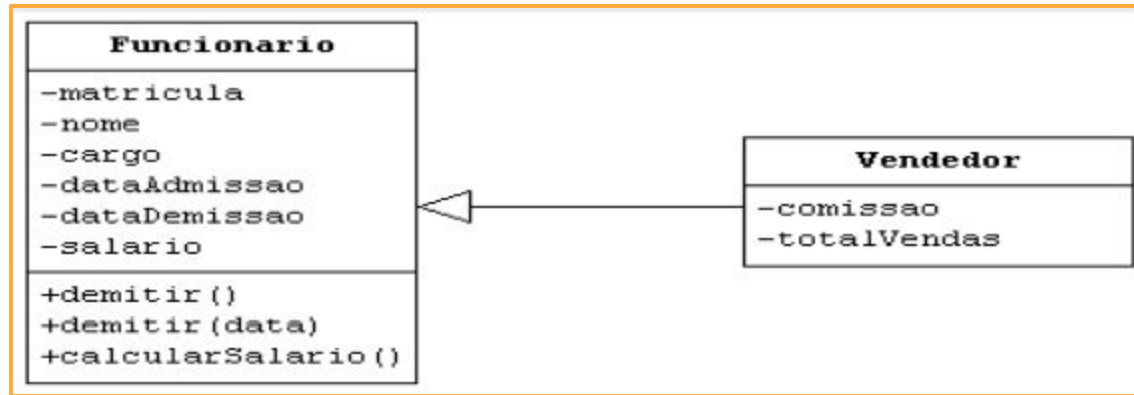
- Usada para referenciar a superclasse.
- `super()` se refere, dentro de um construtor, ao construtor default existente na superclasse.
- Deve ser a primeira linha do construtor.
- Se não houver uma instrução `super()` ou `this()` explícita em um construtor, Java coloca uma chamada `super()` implícita.

Sobrescrita de Métodos

- Consiste em definir, em alguma subclasse, um método com a mesma assinatura (mesmo nome e mesmos argumentos) e mesmo tipo de retorno de uma superclasse.

Exercício

- Criar classes de entidade para o diagrama abaixo:



- Criar uma classe de teste **FolhaPagamento** que instancie um funcionário e um vendedor e execute o método `calcularSalario`.

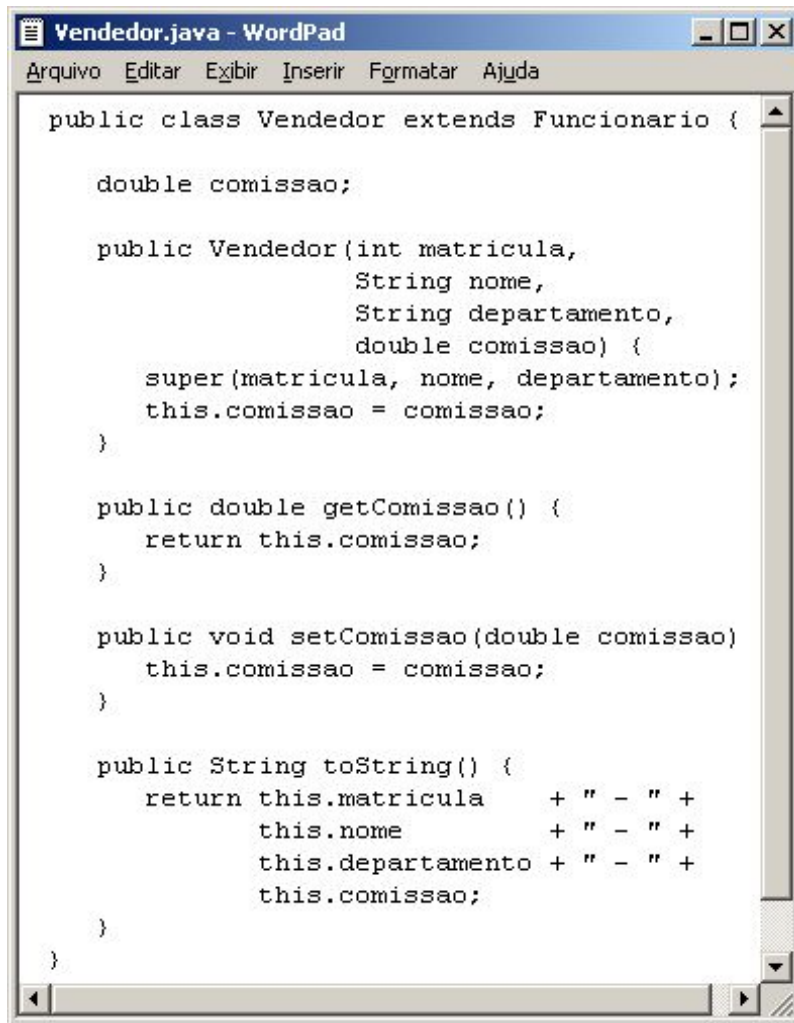
Polimorfismo

Conceito de Polimorfismo

- Polimorfismo é a capacidade de decidir, em tempo de execução, qual método deve ser chamado.
- Para que o polimorfismo seja possível, são necessários:
 - Uma hierarquia de classes onde existam métodos sobrescritos = Herança.
 - A possibilidade de se criar um objeto de uma subclasse e colocá-lo em uma referência de uma superclasse.
- Facilita o desenvolvimento de sistemas pois evita a necessidade de se conhecer, em tempo de compilação, qual objeto deve ser chamado.

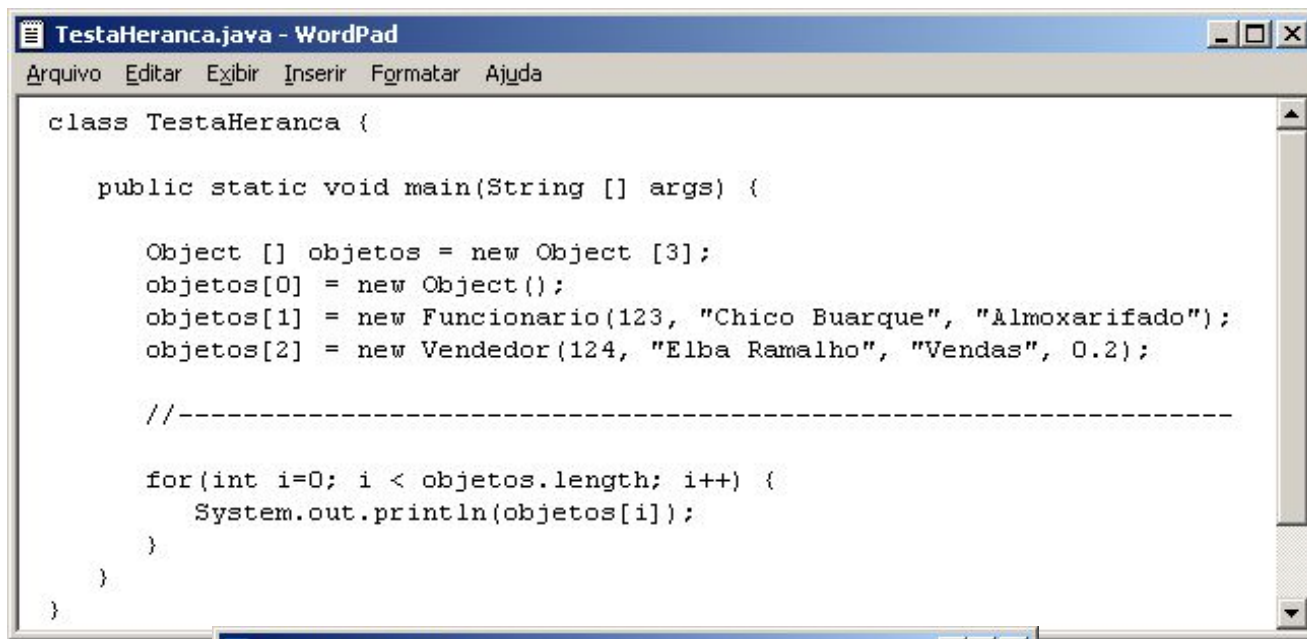
Conceito de Polimorfismo

- Sobrecarga e sobrescrita são exemplos de usos do polimorfismo.
- O Polimorfismo pode ser verificado também em classes diferentes, que não têm nada a ver uma com a outra.
- O Polimorfismo “para valer” é o obtido a partir da sobrescrita de métodos.
- Vamos ver um exemplo de uso do Polimorfismo através do método “toString”.



```
public class Vendedor extends Funcionario {  
  
    double comissao;  
  
    public Vendedor(int matricula,  
                    String nome,  
                    String departamento,  
                    double comissao) {  
        super(matricula, nome, departamento);  
        this.comissao = comissao;  
    }  
  
    public double getComissao() {  
        return this.comissao;  
    }  
  
    public void setComissao(double comissao)  
    {  
        this.comissao = comissao;  
    }  
  
    public String toString() {  
        return this.matricula + " - " +  
               this.nome + " - " +  
               this.departamento + " - " +  
               this.comissao;  
    }  
}
```

**Sobrescrevemos o
método
“toString” herdado
de Object nas duas
classes**



```
class TestaHeranca {  
  
    public static void main(String [] args) {  
  
        Object [] objetos = new Object [3];  
        objetos[0] = new Object();  
        objetos[1] = new Funcionario(123, "Chico Buarque", "Almoxarifado");  
        objetos[2] = new Vendedor(124, "Elba Ramalho", "Vendas", 0.2);  
  
        //-----  
  
        for(int i=0; i < objetos.length; i++) {  
            System.out.println(objetos[i]);  
        }  
    }  
}
```



```
C:\aula06>java TestaHeranca  
java.lang.Object@923e30  
123 - Chico Buarque - Almoxarifado  
124 - Elba Ramalho - Vendas - 0.2  
C:\aula06>
```

O método “toString”
existe nos três objetos
por herança, mas na hora
de executar cada objeto
roda o seu, independente
do tipo da referência

Operador instanceof

- Retorna um valor lógico indicando se o objeto é uma instância de uma determinada classe.

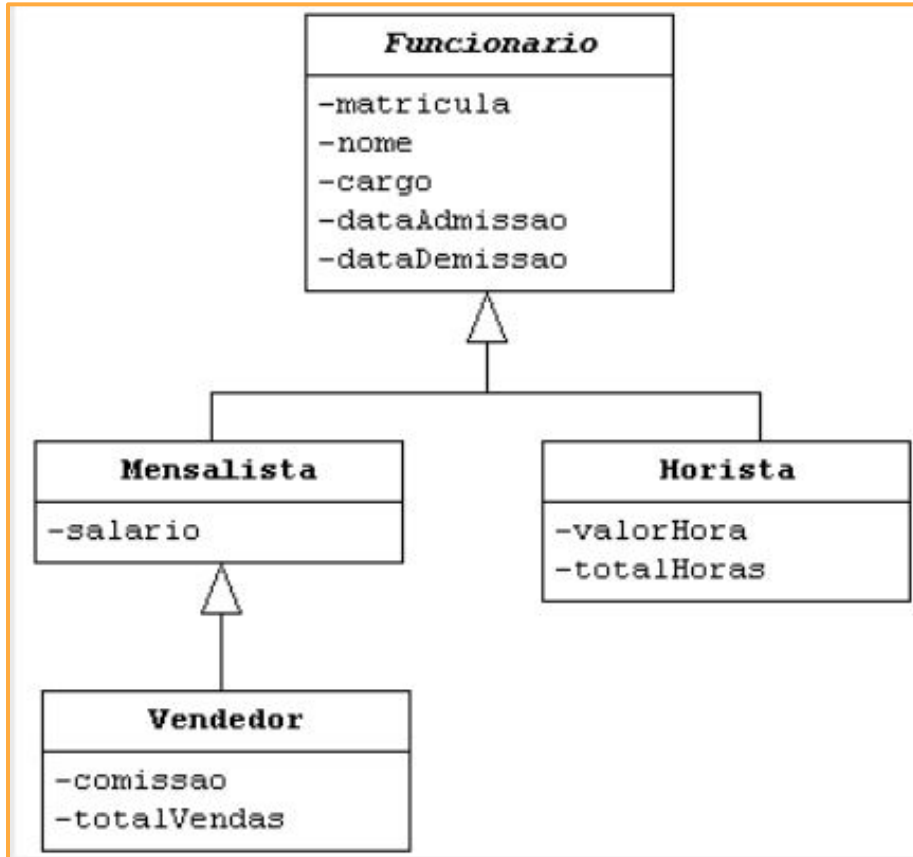
```
if (contas[i] instanceof ContaEspecial) {  
    ContaEspecial ce = (ContaEspecial)  
    contas[i];  
    total += ce.getLimite();  
}
```

Abstrações

Classes Abstratas

- É uma classe que não permite criar objetos.
- Para indicar que uma classe é abstrata, deve ser usada a palavra reservada **abstract**.
- Apesar dessas características, classes abstratas devem ter construtores de forma a não quebrar a “hierarquia de construtores” que você estudou.

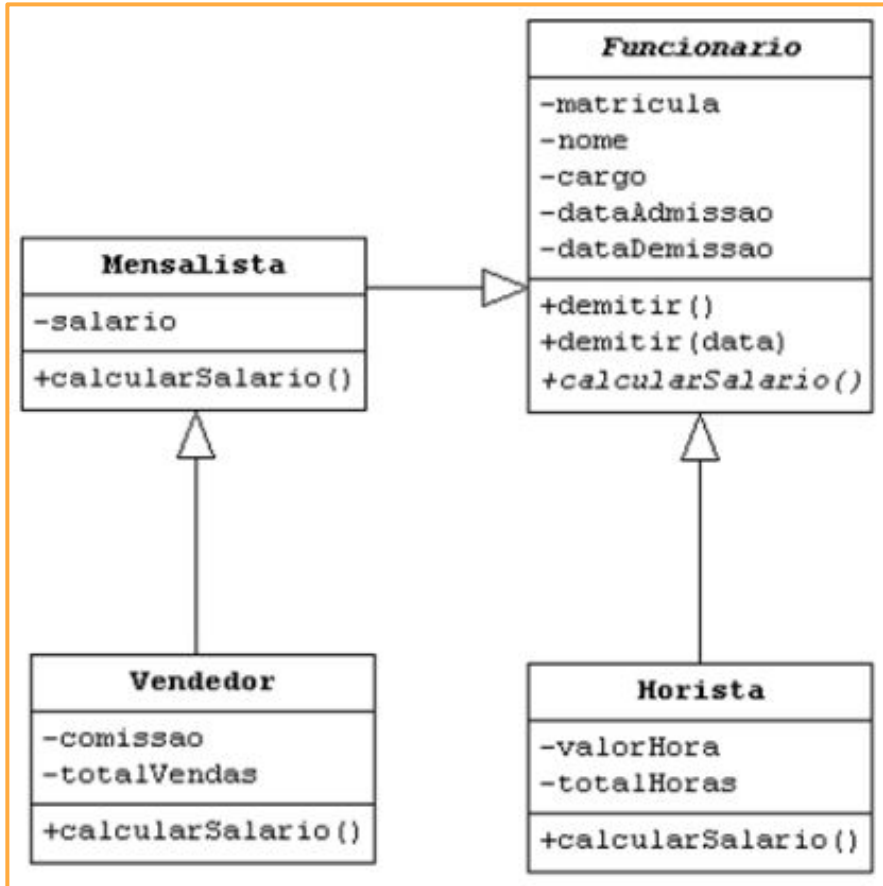
Classes Abstratas



- A classe **Funcionario** é declarada como abstrata.
- Ainda pode ter propriedades e métodos concretos.

Classes Abstratas

- Métodos abstratos são métodos existentes em classes abstratas que não possuem implementação.
- Devem ser obrigatoriamente implementados pelas classes filhas concretas.
- A existência de um método abstrato obriga que a classe seja abstrata.



Modificador final

- Uma propriedade final é uma constante.
- Um método final não pode ser sobrescrito.
- Uma classe final não pode ter subclasses.

```
public double final PI = 3.1415f;  
  
public final boolean verificarSenha(String senha) {  
    ...  
}  
  
public final class String {  
    ...  
}
```