


Fundamentos de Java

Herança e Polimorfismo




Softblue
cursos online

Tópicos Abordados

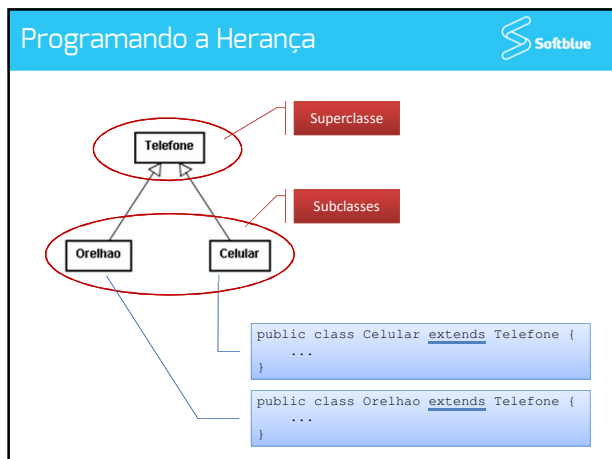


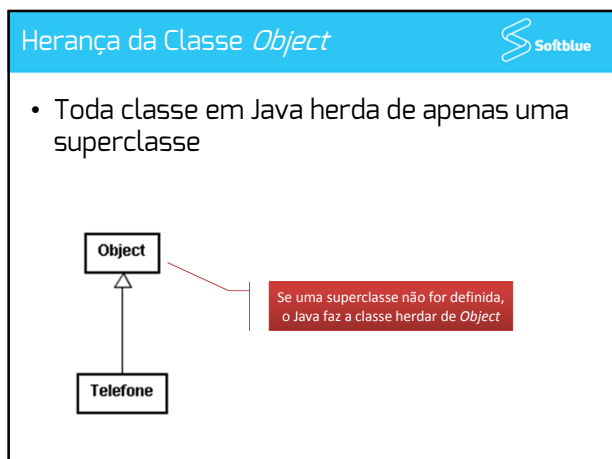
- Herança
- O modificador *protected*
- Sobrescrita de métodos
- A palavra-chave *super*
- Polimorfismo
- O operador *instanceof*

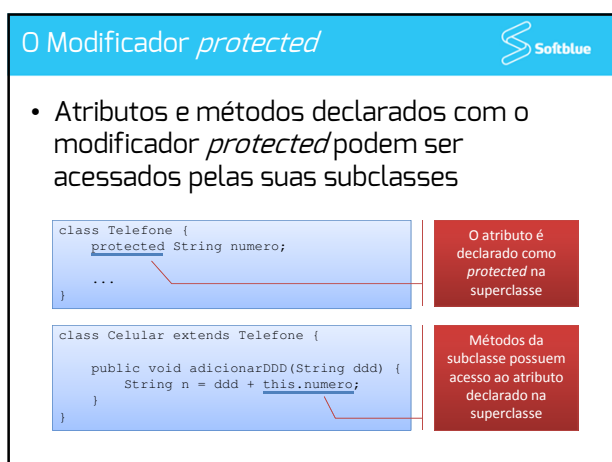
Herança



- A herança é um mecanismo que permite que uma classe possa herdar o comportamento de outra classe, ao mesmo tempo em que novos comportamentos podem ser estabelecidos
- A vantagem da herança é agrupar coisas comuns para poder reaproveitar código







Sobrescrita de Métodos



- Técnica também conhecida como *overriding*
- Quando uma classe herda de outra, ela pode redefinir métodos da superclasse, isto é, sobrescrever métodos
 - Os métodos sobrescritos substituem os métodos da superclasse
 - A assinatura do método sobrescrito deve ser a mesma do método original

Sobrescrita de Métodos



```
class Telefone {  
    public void telefonar() {  
        //código para telefonar  
    }  
}
```

```
class Orelhao extends Telefone {  
    public void telefonar() {  
        //código para telefonar do orelhão  
    }  
}
```

```
Orelhao o = new Orelhao();  
o.telefonar();
```



Como o método foi sobrescrito, é chamado o método da subclasse

Sobrescrita de Métodos



```
class Telefone {  
    public void telefonar() {  
        //código para telefonar  
    }  
}
```

```
class Orelhao extends Telefone {  
    public void telefonar(int numero) {  
        //código para telefonar do orelhão  
    }  
}
```

```
Orelhao o = new Orelhao();  
o.telefonar();
```



Não há sobrescrita de método. Métodos sobrescritos devem ter a mesma assinatura (tipo de retorno, nome do método e parâmetros)

Sobrescrevendo Métodos de *Object*



- Método ***toString()***
 - As classes podem sobrescrever este método para mostrarem uma mensagem que as representem
 - O método *System.out.println()*, por exemplo, utiliza este método
- Método ***equals(Object)***
 - É a forma que o Java tem de comparar objetos pelo seu conteúdo ao invés de comparar as referências (como acontece ao usarmos "==")

Usando o *super*



- O método que foi sobrescrito pode ser acessado pelo método que o sobrescreveu através da palavra-chave *super*

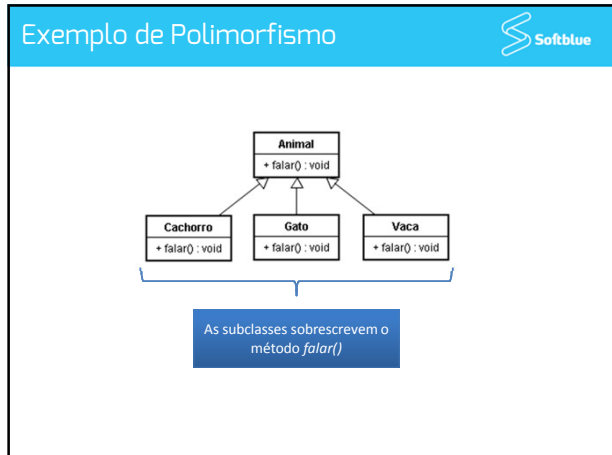
```
class Orelhao extends Telefone {  
    public void telefonar() {  
        super.telefonar()  
    }  
}
```

Chama o método da superclasse

Polimorfismo



- É a capacidade que um método tem de agir de diferentes formas, dependendo do objeto sobre o qual está sendo chamado
- Quando ocorre a chamada de um método, a JVM decide qual método invocar dependendo do objeto instanciado na memória



Exemplo de Polimorfismo

```
class Animal {
    public void falar() {
    }
}

class Cachorro extends Animal {
    public void falar() {
        System.out.println("Au");
    }
}

class Gato extends Animal {
    public void falar() {
        System.out.println("Miau");
    }
}

class Vaca extends Animal {
    public void falar() {
        System.out.println("Mu");
    }
}
```

Cada animal implementa o método *falar()* do seu modo

Exemplo de Polimorfismo

```
Animal a = new Cachorro();
a.falar();

Animal a = new Gato();
a.falar();


Animal a = new Vaca();
a.falar();

Cachorro c = new Cachorro();
Animal a = (Animal) c;
a.falar();
```

O método invocado é determinado pelo tipo do objeto que está armazenado na memória

A forma como objeto é referenciado não influencia na decisão sobre qual método será invocado

Exemplo de Polimorfismo



Animal

+ falar(): void

Cachorro

+ falar(): void

+ morder(): void

Animal a = new Cachorro();

a.falar();

Resultado: "Au"

Animal a = new Cachorro();

a.morder();

Método inexistente

Animal a = new Cachorro();


Cachorro c = (Cachorro) a;

c.morder();

OK

O tipo pelo qual o objeto é referenciado determina quais métodos e/ou atributos podem ser invocados

O Operador *instanceOf*



Utilizado para verificar se um objeto pertence à determinada classe

Animal a = new Cachorro();

a instanceof Cachorro

true

a instanceof Animal

true


a instanceof Gato

false

a instanceof Object

true

Normalmente é utilizado antes de realizar um cast, para garantir que a operação é válida



6