

Aula: Herança e Polimorfismo

Instituto Metr pole Digital - UFRN

Disciplina: LP2

Docente: Emerson Alencar



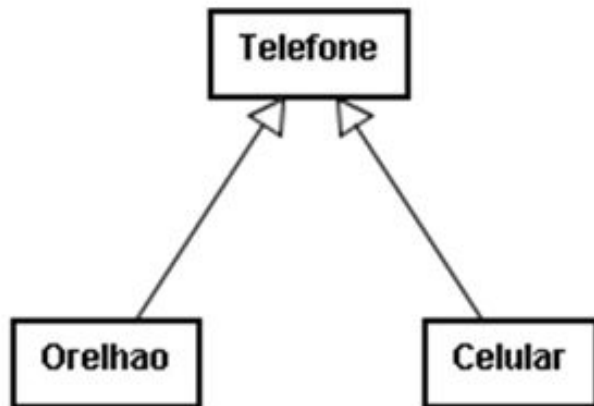
Herança

A Herança é um mecanismo que permite que uma classe possa herdar o comportamento e características de outra classe, ao mesmo tempo em que novos comportamentos e características podem ser estabelecidas.

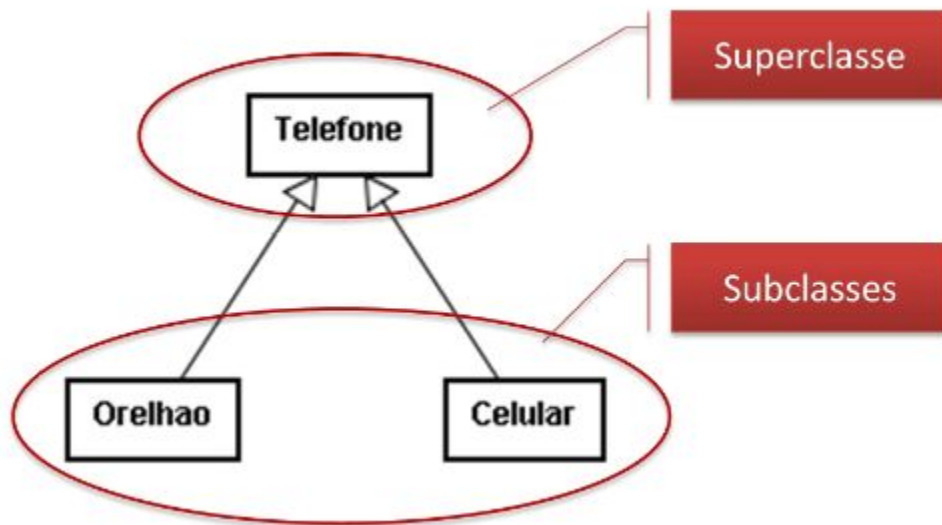
- A relação de herança é dada entre classes
- Podemos chamar de superclasse e outra de subclasse

A vantagem da herança é agrupar coisas comuns para poder reaproveitar o código

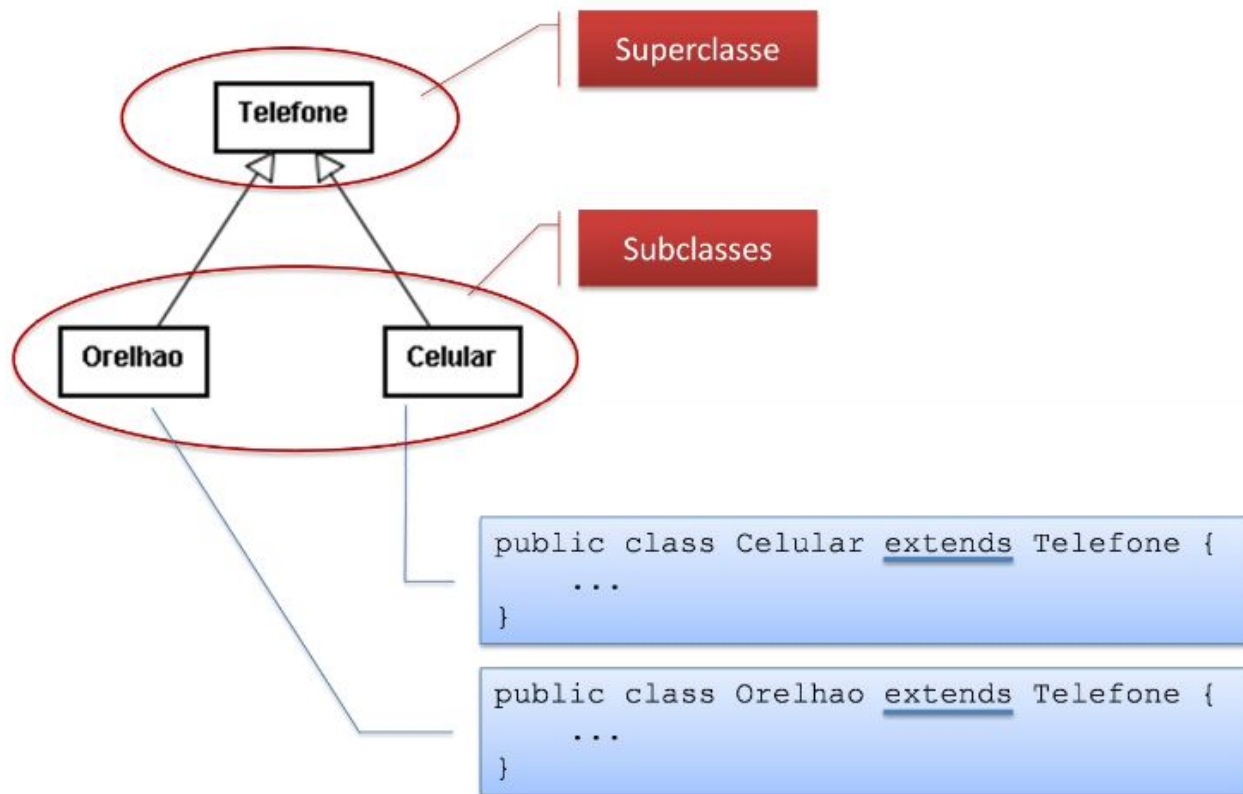
Representando e Programando Herança



Representando e Programando Herança

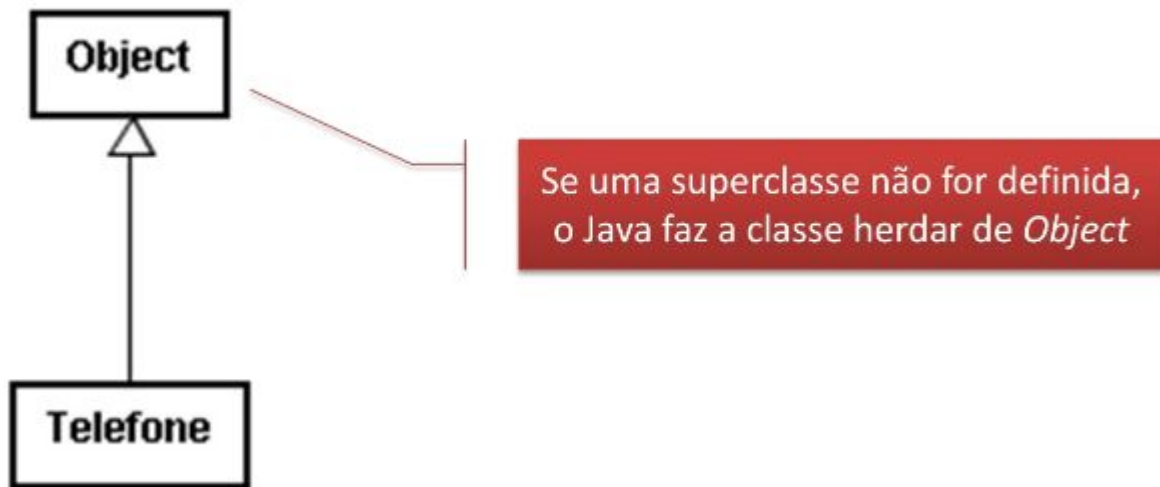


Representando e Programando Herança



Herança da classe *Object*

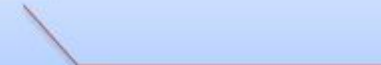
Todas as classes herdam apenas de uma superclasse



Modificador *protected*

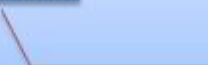
Atributos e métodos declarados com o modificador `protected` podem ser acessados pelas suas subclasses e classes do mesmo pacote.

```
class Telefone {  
    protected String numero;  
    ...  
}
```



O atributo é
declarado como
protected na
superclasse

```
class Celular extends Telefone {  
    public void adicionarDDD(String ddd) {  
        String n = ddd + this.numero;  
    }  
}
```



Métodos da
subclasse possuem
acesso ao atributo
declarado na
superclasse

Sobrescrita de métodos

- Técnica conhecida como override
- 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

Sobrescrevendo Métodos do Object

- Método `toString()`: As classes podem sobrescrever este método para mostrarem uma mensagem que as representem. O método `System.out.println()`, utiliza esse 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()  
    }  
}
```

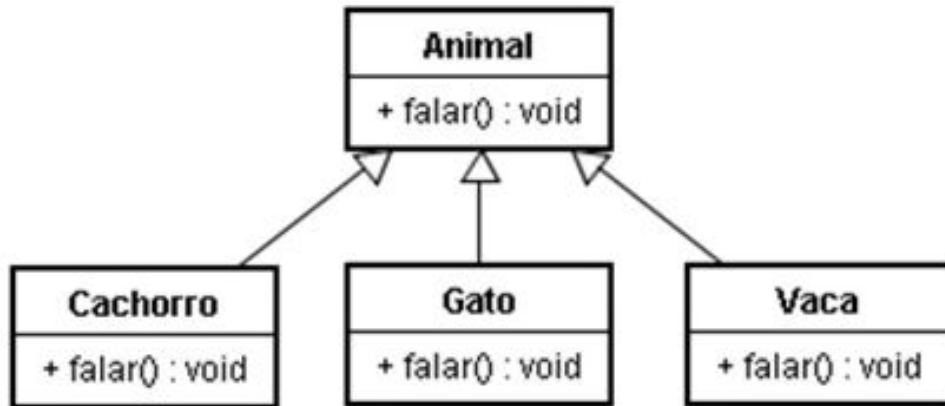
Chama o método
da superclasse

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

Polimorfismo



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

Polimorfismo

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

Resultado: "Au"

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

Resultado: "Miau"

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

Resultado: "Mu"

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

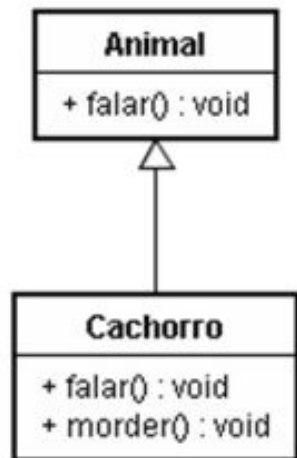
Polimorfismo

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

Resultado: "Au"

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

Polimorfismo



```
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

Operador instanceof

Utilizado para verificar se um determinado objeto pertence a uma determinada classe.

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

```
a instanceof Cachorro
```

true

```
a instanceof Animal
```

true

```
a instanceof Gato
```

false

```
a instanceof Object
```

true

Por hoje é só...