

Programação Orientada a Objetos

Departamento de Computação
Universidade Federal de Sergipe

Classes Abstratas, Subtipos e Interfaces

Prof. Kalil Araujo Bispo
kalil@dcomp.ufs.br



Roteiro

- Classes Abstratas
- Subtipos
- Verificação Dinâmica de Tipos
- Ampliação e Estreitamento
- Interfaces

Classe Abstrata

- Classe que não podem ser instanciada
 - Podem ter referência
- Servem de modelo para outras classes
- Os métodos abstratos de uma classe abstrata devem ser sobrescritos nas classes filhas

Classes Abstratas

```
abstract public class Conta {  
    private double saldo;  
  
    public void setSaldo(double saldo) {  
        this.saldo = saldo;  
    }  
  
    public double getSaldo() {  
        return saldo;  
    }  
  
    public abstract void imprimeExtrato();  
}
```

Classes Abstratas


```
public class ContaPoupanca extends Conta {  
  
    @Override  
    public void imprimeExtrato() {  
        System.out.println("### Extrato da Conta ###");  
        Date date = new Date();  
        System.out.println("Saldo: " + this.getSaldo());  
        System.out.println("Data: " + sdf.format(date));  
    }  
  
}
```

Subtipos

- **Princípio da Substituição (Liskov)**
 - Se S é um subtipo de T, então objetos do tipo T podem ser substituídos por objetos do tipo S, sem alterar qualquer propriedade desejável do programa (correção, tarefa executada, etc)
- Em Java, `extends` é um mecanismo de herança mas também define uma relação de subtipos

Subtipos

```
abstract class Figura {  
    private float x,y;  
    void move(int dx,dy) {  
        x+=dx;  
        y+=dy;  
    }  
    abstract float area( );  
}
```



```
class Circulo extends Figura {  
    private float raio;  
  
    @override  
    float area( ) {  
        return 2*PI*raio;  
    }  
}
```

```
class Retangulo extends Figura {  
    private float alt, comp;  
  
    @override  
    float area( ) {  
        return alt * comp;  
    }  
}
```

Subtipos

```
Figura[] figs = new Figura[10];  
....  
float areaTotal = 0;  
for (int i=0; i<10; i++)  
    AreaTotal += figs[i].area( );
```

- **figs é uma variável polimórfica**
 - Pelo princípio da substituição fig[i] pode referenciar uma Figura, um Círculo ou um Retângulo.
- **Na chamada figs[i].area(), a escolha do método é feita dinamicamente**
 - Depende do tipo de figura que está sendo referenciada por figs[i].

Subtipos

- Casts são essenciais para verificação estática de tipos (compilação)
- Casts e isinstance:
 - ((Tipo) variável)
 - variável **isinstance** Tipo
 - O tipo de variável deve ser supertipo de Tipo
 - O Cast “((Tipo) variável)” gera uma exceção se “variável **isinstance** Tipo” retornar false

Verificação Dinâmica de Tipos

- Com instanceof

```
...  
Conta c = this.procurar("5657-x");  
if (c instanceof Poupanca) {  
    Poupanca p = ((Poupanca) c);  
    p.renderJuros(0.01);  
}  
else  
    System.out.print("Poupança inexistente!");  
...
```

Verificação Dinâmica de Tipos

- Com instanceof

```
...  
Conta c = this.procurar("5657-x");  
if (c instanceof Poupanca) {  
    ((Poupanca) c).renderJuros(0.01);  
}  
else  
    System.out.print("Poupança inexistente!");  
...
```

Verificação Dinâmica de Tipos

- Com instanceof

```
...  
Conta c = this.procurar("5657-x");  
//O que acontece se c não for do tipo  
//Poupança?  
Poupanca p = ((Poupanca) c);  
p.renderJuros(0.01);  
...
```

Ampliação e Estreitamento

- **Ampliação**

- Ocorre quando uma instância de uma subclasse é atribuída a uma variável da superclasse (o inverso não é possível).

- **Estreitamento**

- É o inverso de ampliação. Só é possível quando o objeto sendo estreitado é uma instância do subtipo (ou de seus descendentes).
- Pode causar erros se isso não for garantido.
- Java exige conversão explícita, lançando uma `ClassCastException` quando é inválida.

Ampliação e Estreitamento

```
public class Teste {  
    public static void main(String[] args) {  
        Animal a = new Cachorro();  
        Cachorro c = (Cachorro)a;  
        c.latir();  
  
        // Forma resumida:  
        a = new Gato();  
        ((Gato)a).miar();  
    }  
}
```

Upcast

Downcast

Interfaces

- Java não permite herança múltipla
- Usa o conceito de interface como uma forma de implementar subtipagem múltipla
- Uma interface é semelhante a uma classe abstrata onde são definidos apenas
 - Protótipos dos métodos (assinatura)
 - Constantes

Interfaces

- Uma classe pode implementar várias interfaces (subtipagem múltipla)
- Uma interface pode estender (herdar) várias interfaces
- Ao implementar uma interface, a classe deve implementar os métodos declarados na interface ou será obrigatoriamente uma classe abstrata

Interfaces

```
interface Aluno {  
    void estudar ( );  
    void estagiar ( );  
}
```

```
class Graduando implements Aluno {  
    public void estudar ( ) { ... }  
    public void estagiar ( ) { ... }  
}
```

Interfaces

```
interface Cirurgiao { void operar(); }
```

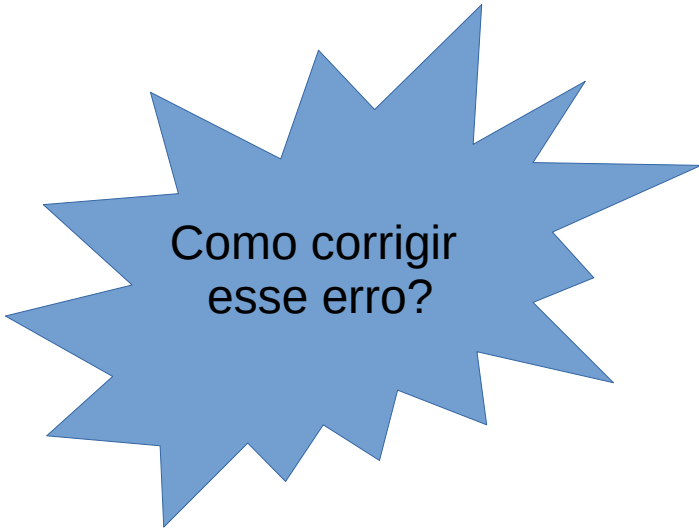
```
interface Neurologista { void consultar(); }
```

```
public class Medico { public void consultar() { } }
```

```
public class NeuroCirurgiao extends Medico implements  
Cirurgiao, Neurologista {  
    public void operar() { }  
}
```

Interfaces

```
public class Hospital {  
    static void plantaoCirurgico(Cirurgiao x) { x.operar(); }  
    static void atendimentoGeral(Medico x) { x.consultar(); }  
    static void neuroAtendimento(Neurologista x) { x.consultar(); }  
    static void neuroCirurgia(NeuroCirurgiao x) { x.operar(); }  
  
    public static void main(String[] args) {  
        Medico doutor = new NeuroCirurgiao();  
        plantaoCirurgico(doutor);  
        atendimentoGeral(doutor);  
        neuroAtendimento(doutor);  
        neuroCirurgia(doutor);  
    }  
}
```



Como corrigir
esse erro?

Interfaces

- **Classe Abstrata x Interface:**
 - Somente as assinaturas dos métodos podem ser incluídas em interfaces, ou seja, não é permitido incluir qualquer código
 - Utiliza a palavra reservada `interface` no lugar de `class`;
 - Não permite a declaração de variáveis (exceto constantes)
 - Uma interface pode estender uma ou mais interfaces