Interfaces (Java 7)

Interfaces, como classes e métodos abstratos, fornecem declarações de comportamentos que outras classes devem implementar.

A herança simples é restritiva quando pretendemos usar um determinado comportamento em diferentes ramos de uma árvore hierárquica.

Com herança múltipla, uma classe pode herdar de mais que uma superclasse obtendo ao mesmo tempo atributos e comportamentos de todas as superclasses. No entanto a múltipla herança torna a linguagem de programação muito mais complicada.

Java só possui herança simples. Para resolver o problema da necessidade de comportamentos comuns em classes em diferentes ramos de uma árvore hierárquica, Java tem outra hierarquia de comportamentos de classes, a hierarquia de interfaces. Assim quando criámos uma classe, essa classe só tem uma superclasse, mas pode escolher diferentes comportamentos da hierarquia de interfaces.

Uma interface em Java é uma colecção de comportamentos abstratos que podem ser incluídos em qualquer classe para adicionar a essa classe comportamentos que não são fornecidos pelas superclasses.

Uma interface em Java só contém definições de métodos abstratos e constantes estáticas – não contém variáveis de instância nem implementação de métodos.

As interfaces são como as classes, declarados em ficheiros fonte, compilados em ficheiros *.class* e podem ser usados para tipos de dados de variáveis.

São diferentes das classes porque não podem ser instanciados.

Exemplo:

```
public interface Imposto {
    double calculoDoImposto();
}
```

Ao contrário das classes, uma interface pode ser adicionada a uma das classes que já é uma subclasse de outra classe.

Pode-se pretender usar esta interface em muitas classes diferentes: roupa, comida, carros, etc. Seria inconveniente que todos estes objetos derivassem de uma única classe.

Para além disso cada tipo diferente poderá ter um modo diferente de cálculo de imposto. Assim define-se a interface Imposto e cada classe deve implementar essa interface.

Uma interface pode conter vários métodos (incluindo métodos *overloaded*) e campos de dados, mas estes têm de ser *static final*.

Exemplo:

```
abstract class FormaGeom {
  public abstract void desenhar();
  public String toString() { return "Forma Geometrica"; };
}

interface Dimensoes {
  double area();
  double perimetro();
}

class Ponto {
  protected double x,y;
  public Ponto(double x, double y){this.x = x; this.y = y;}
  public String toString() {
      return  super.toString() +
              ": Ponto de Coordenadas = " +
              "(" + x + ", " + y + ")";
  }
}

class  SegmRecta  extends  FormaGeom  {
  protected Ponto p1, p2;
  public SegmRecta(double x1,double y1,double x2,double y2) {
      p1 = new Ponto(x1, y1);
      p2 = new Ponto(x2, y2);
  }
  public double comprimento() {
      double dx = p1.x - p2.x;
      double dy = p1.y - p2.y;
      return Math.sqrt(dx*dx + dy*dy);
  }
  public void desenhar() {
    System.out.println("\tDesenhar um segmento de recta");
  }
  public String toString() {
      return  super.toString() +
              ": Segmento de Recta de comprimento = " +
              comprimento();
  }
}
```

```
class Circunferencia extends FormaGeom implements Dimensoes{
   protected double raio;
   public Circunferencia( double r )  { raio = r; }
   public double area() { return Math.PI * raio * raio; }
   public double perimetro() { return 2 * Math.PI * raio; }
   public void desenhar() {
       System.out.println("\tDesenhar uma circunferencia");
   }
   public String toString() {
       return  super.toString() +
               ": Circunferencia de raio = " + raio;
   }
 }

class Quadrado extends FormaGeom implements Dimensoes {
   protected double lado;
   public Quadrado( double l )  {lado = l; }
   public double area() { return lado * lado; }
   public double perimetro() { return 4 * lado; }
   public void desenhar() {
       System.out.println("\tDesenhar um quadrado");
   }
   public String toString()
       { return  super.toString() +
                ": Quadrado de lado = " + lado; }
}

class Triangulo extends FormaGeom implements Dimensoes {
   protected double base, altura;
   public Triangulo( double b, double a )  {
       base = b;  altura = a;
   }
   public double area() { return base * altura / 2; }
   public double perimetro() {
       return base+2*Math.sqrt(base*base/4+altura*altura);
   }
   public void desenhar() {
       System.out.println("\tDesenhar uma triangulo");
   }
   public String toString(){
       return  super.toString() +
           ": Triangulo de base = " + base +
           " e de altura = " + altura;
   }
}
```

```
public class Teste {
  public static void main(String args []) {
       FormaGeom fg []= new FormaGeom [4];
       fg[0] = new SegmRecta(4, 5, 8, 9);
       fg[1] = new Circunferencia(4);
       fg[2] = new Quadrado(3);
       fg[3] = new Triangulo(4, 3);
       for(int i= 0; i<fg.length; i++) {
           System.out.println( fg[i] );
           fg[i].desenhar();
       }
       Dimensoes d []= new Dimensoes [3];
       d[0] = (Circunferencia) fg[1];
       d[1] = (Quadrado) fg [2];
       d[2] = (Triangulo) fg [3];
       for(int i= 0; i<d.length; i++) {
           System.out.println( d[i] + "\n\t Area = " +
                               d[i].area() +
                               "\n\t Perimetro = " +
                               d[i].perimetro() );
       }
  }
}
```

Saída produzida pelo programa:

```
Forma Geometrica: Segmento de Recta de comprimento = 5.65685
       Desenhar um segmento de recta
Forma Geometrica: Circunferencia de raio = 4
       Desenhar uma circunferencia
Forma Geometrica: Quadrado de lado = 3
       Desenhar um quadrado
Forma Geometrica: Triangulo de base = 4 e de altura = 3
       Desenhar uma triangulo
Forma Geometrica: Circunferencia de raio = 4
       Area = 50.2655
       Perimetro = 25.1327
Forma Geometrica: Quadrado de lado = 3
       Area = 9
       Perimetro = 12
Forma Geometrica: Triangulo de base = 4 e de altura = 3
       Area = 6
       Perimetro = 11.2111
```

Outro exemplo do uso de interfaces:

- Faça um programa que construa uma tabela para os quadrados dos inteiros de 1 a 5 e outra para os cubos dos inteiros dos múltiplos de 10 entre 10 e 50.

```
interface Funcao {
  public abstract int f(int x);
}

class Quadrado implements Funcao {
  public int f(int x) { return x*x; }
}

class Cubo implements Funcao {
  public int f(int x) { return x*x*x; }
}

class Tabela {
  public static void tabela(Funcao t, int a, int b, int incr)
     System.out.println( "x\t| f(x)\n--------------");
     for (int x=a; x<b; x+=incr)
       System.out.println(x + "\t| " + t.f(x));
     System.out.println();
  }

  public static void main(String args []){
     Funcao t = new Quadrado();
     tabela(t, 1, 5, 1);
     t = new Cubo();
     tabela(t, 10, 50, 10);
  }
}
```

Saída produzida pelo programa:

```
x     | f(x)
--------------
1     | 1
2     | 4
3     | 9
4     | 16
5     | 25

x     | f(x)
--------------
10    | 1000
20    | 8000
30    | 27000
40    | 64000
50    | 125000
```