ED
Polimorfismo
Interfaces

Profa. Célia Taniwaki

Polimorfismo

- Polimorfismo
  - significa muitas (Poli) formas (morfo)
  - Em Programação Orientada a Objetos, refere-se ao fato de **um mesmo método produzir resultados diferentes**.
- Pode acontecer em 2 casos:
  - Métodos sobrecarregados:
    - Mesmo método com lista de parâmetros diferentes (assinaturas diferentes).
    - O Java sabe qual método executar pelo número de parâmetros passados.
      - Ex.: método média que recebe 2 valores e devolve a média aritmética dos 2 valores e método média que recebe 3 valores e devolve a média aritmética dos 3 valores
  - Métodos reescritos:
    - Método existente na classe mãe, reescrito na classe filha (mesma assinatura).
    - O Java sabe qual método executar devido ao nome do objeto associado ao método chamado.
      - Ex.: método calcSalario das classes Vendedor e Horista

Método sobrecarregado

```
class Media {

  public static double media (double a, double b)
  {
    return (a + b) / 2;
  }

  public static double media (double a, double b, double c)
  {
    return (a + b + c) / 3;
  }

  public static void main(String args[])
  {
    System.out.println (media (5, 6));
    System.out.println (media (5, 6, 7));
  }
}
```

Método reescrito ou sobrescrito

- Ex.: método calcSalario das classes Vendedor e Horista
- Nesse exemplo, há a criação do objeto v, da classe Vendedor e do objeto h, da classe Horista
- As chamadas v.calcSalario( ) e h.calcSalario( ) são exemplos de Polimorfismo
- Na classe Empresa, criamos um ArrayList de Funcionario, e fizemos um método que percorre a lista chamando o método calcSalario (Polimorfismo mais forte)

```
public void exibeTotalSalario(){
   double total = 0.0;
   for (Funcionario f: lista)
      total += f.calcSalario();
   System.out.println("O total gasto em salário é " + total);
}
```

Interfaces

- Permite estabelecer um “contrato” entre as classes, forçando que elas implementem os métodos definidos na interface
- Semelhante a classes abstratas, pois força a implementação de métodos abstratos
- Porém difere de classe abstrata, pois
  - Não se relaciona com as classes através de herança
  - Contém apenas métodos abstratos (não é necessário especificar abstract na definição do método)
  - Toda variável definida numa interface é implicitamente do tipo final, ou seja, é uma constante. Uma vez atribuído um valor a essa variável, esse valor não pode mais ser alterado.
- Não permite implementação de nenhum método, apenas sua assinatura (especificação)

Implementação de interface

- Declaração de interface semelhante à declaração de uma classe, porém utiliza-se **interface** no lugar de **class**

```
public interface NomeInterface  {
        //  assinaturas dos métodos
}
```

- A classe que implementa os métodos especificados numa interface deve ter em sua declaração a cláusula **implements**

```
public class NomeClasse implements NomeInterface {
    // implementação dos métodos definidos na interface
}
```

Interface e Polimorfismo

- O nome da interface pode ser utilizado como tipo de um vetor (ou ArrayList) ou como tipo de um parâmetro passado a um método
  - Ex.: interface chamada Tributavel
    - Pode-se declarar um List:
      - List<Tributavel> lista = new ArrayList<Tributavel>();
      - Cada elemento dessa lista pode ser um objeto de qualquer classe que implementa Tributavel
    - Pode-se ter um método que recebe obj Tributavel
      - public void adicionaTributavel(Tributavel obj)
      - Esse método aceita como parâmetro um objeto de qualquer classe que implementa Tributavel

Interface e Polimorfismo

- Dessa forma, todos os objetos que estão no List de tipo Tributavel implementam o método ou os métodos definidos na interface Tributavel
- Portanto, é possível chamar obj.metodo( )
- Ex: método que calcula total de tributos

```
public double calculaTotalTributos( ) {
   double total = 0.0;
   for (Tributavel t : lista)
      total = total + t.getValorTributo( );
   return total;
}
```