
Newton
Quem se prepara, não para.

Programação Orienta a Objetos

3º período

Professora: Michelle Hanne

Sumário

- ✓ Passagem de Parâmetro
- ✓ Classe Abstrata
- ✓ Interface

Herança

- Herança provê **reuso** de classes já construídas.

- Alguns benefícios do uso de herança:
  - evitar duplicação de código;
  - reuso de código;
  - manutenção mais fácil (desde que não haja abuso do recurso);
  - extensibilidade.

Passagem de Parâmetro

Newton
Quem se prepara, não para.

- Em Java a passagem de parâmetros para métodos se dá sempre por valor.

- Não existe passagem de parâmetro por referência em Java.

- O que ocorre quando um objeto é passado por parâmetro?

Passagem de Parâmetro

Newton
Quem se prepara, não para.

- Quando um objeto é passado por parâmetro, na verdade a sua referência está sendo passada por parâmetro.

- Alterações realizadas no objeto dentro do método tem impacto no argumento passado para o método.

- Por exemplo, se o objeto ***a*** for passado como parâmetro para um método ***m(A x)*** e em ***m*** ocorrer alguma alteração em ***x***, após a execução de ***m,*** **a** estará com as alterações sofridas por ***x***.

Passagem de Parâmetro - Exemplo

```
public class Teste{

    public static void m(A obj){
        obj.setA(10);
        obj.setB(20);
        obj = new A(35, 45, 55);
    }
```

Passagem de Parâmetro - Exemplo

Newton
Quem se prepara, não para.

```
public static void main (String[] args){
      A obj1, obj2;

      obj1 = new A(1,2,3);
      obj2 = new A(7,8,9);

      obj1.ImprimeValores();
      obj2.ImprimeValores();

      m(obj1);

      obj1.ImprimeValores();
   }
}
```

Passagem de Parâmetro - Exemplo

Newton
Quem se prepara, não para.

Saída do programa:

1 - 2 - 3
7 - 8 - 9
10 - 20 - 3

Passagem de Arranjo como Parâmetro

Newton
Quem se prepara, não para.

- Para passar um arranjo como parâmetro, deve-se indicar o nome do arranjo sem colchetes na chamada do método.

```
metodo(arranjo);
```

- O método que recebe o arranjo como parâmetro deve indicar isso na sua lista de parâmetros.

```
void metodo(int b[])
```

Passagem de Arranjo como Parâmetro

Newton
Quem se prepara, não para.

- Quando um arranjo é passado como parâmetro, o que o método recebe é uma cópia da sua referência. Desta forma, alterações sofridas pelo arranjo no método refletem no arranjo que foi passado como parâmetro.

Passagem de Arranjo como Parâmetro - Exemplo


```
public class TesteArranjo {

    public static void alteraArranjo(int b[]){

       for(int i=0; i<b.length; i++){
         b[i] = i*2;
       }

       System.out.println("\n**Arranjo b**");
       for (int valor : b)
           System.out.print(valor + " - ");


       b = new int[3];

       System.out.println("\n**Novo arranjo b**");
       for (int valor : b)
           System.out.print(valor + " - ");
    }
```

Passagem de Arranjo como Parâmetro - Exemplo


```
public static void main(String[] args){
        int[] a = {1,2,3,4,5};


        System.out.println("\n**Arranjo a antes da chamada do método**");
        for (int valor : a)
            System.out.print(valor + " - ");

        alteraArranjo(a);

        System.out.println("\n**Arranjo a após a chamada do método**");
        for (int valor : a)
            System.out.print(valor + " - ");

    }

}
```

Passagem de Arranjo como Parâmetro - Exemplo

Arranjo a antes da chamada do método
1 - 2 - 3 - 4 - 5 –
Arranjo b
0 - 2 - 4 - 6 - 8 -
Novo arranjo b
0 - 0 - 0 -
Arranjo a após a chamada do método
0 - 2 - 4 - 6 - 8 -

Classes Abstratas

Newton
Quem se prepara, não para.

- Há situações em que é útil definir uma classe sabendo-se que não serão instanciados objetos a partir dela.

- **Essas classes são denominadas classes abstratas.**

- **Classes abstratas são aquelas para as quais não se pode instanciar objeto.**

- Um dos motivos pelos quais não se pode instanciar objeto de classes abstratas é que elas são **“semi-completas”:** alguns métodos podem não ter sido definidos.

Classes Abstratas

Newton
Quem se prepara, não para.

- A linguagem Java possui o recurso de criação de classes abstratas.

- Características de uma classe abstrata:

  - é designada pela palavra chave **abstract.**

```
abstract class FiguraGeometrica{...}
```

  - podem possuir métodos sem definição de corpo (método abstrato).

```
public abstract void CalculaArea();
```

  - pode haver hierarquia de classes abstratas.

Classes Abstratas

Newton
Quem se prepara, não para.

- O objetivo do uso de classes abstratas é definir características “semi-completas” a partir das quais outras classes podem ser construídas.

- Em outras palavras, o propósito de uso de classes abstratas é fornecer uma superclasse apropriada da qual outras classes possam herdar interface e/ou implementação.

- As classes não abstratas herdeiras de uma classe abstrata são denominadas **classes concretas**.

Classes Abstratas

```
public abstract class Figura {
  private int cor;

  public abstract void desenhar();

  public abstract void mover();

  public void setCor(int i){
    if (i>0)
     cor = i;
  }
  public int getCor(){
     return cor;
  }

}
```

Classes Abstratas

Newton
Quem se prepara, não para.

```
public abstract class  FiguraBidimensional extends Figura{
   protected float area;

  public abstract void calcularArea();

  public float getArea(){
     return area;
  }

}
```

Classes Abstratas

Newton
Quem se prepara, não para.

```
public abstract class FiguraTridimensional {

  protected float volume;


  public abstract void calcularVolume();

  public float getVolume(){
     return volume;
  }

}
```

Classes Abstratas

Newton
Quem se prepara, não para.

```
public class Quadrado extends FiguraBidimensional{
private float lado;


  public void calcularArea(){
     area = lado * lado;
  }

  public float getLado(){
     return lado;
  }

  public void mover(){
   // Corpo de método mover
  }

    public void desenhar(){
   // Corpo de método desenhar
  }
```

Classes Abstratas

Newton
Quem se prepara, não para.

```
public void setLado(float lado){
    if (lado > 0){
      this.lado = lado;
      calcularArea();

    }
  }
}
```

Classes Abstratas

Newton
Quem se prepara, não para.

```
public class Cubo extends FiguraTridimensional{
  private float lado;


  public void calcularVolume(){
    volume = lado*lado*lado;
  }

  public float getLado(){
     return lado;
  }

  public void setLado(float lado){
    if (lado > 0){
      this.lado = lado;
      calcularVolume();

    }
  }

}
```

Classes Abstratas

Newton
Quem se prepara, não para.

```
public class Main {
    public static void main(String[] args){
      //Figura f = new Figura();
      //FiguraTridimensional t = new FiguraTridimensional();
      //FiguraBidimensional b = new FiguraBidimensional();
      //ERRO: os três comandos anterioroes resultam em erro, pois não é
      //possível instanciar objetos de classes abstratas

      Cubo c = new Cubo();
      Quadrado q = new Quadrado();

      c.setLado(10);
      System.out.println(c.getVolume());
      //c.desenhar(); ERRO: a classe Cubo não implementou o método desenhar()

      q.setLado(3);
      System.out.println(q.getArea());
      q.desenhar();
    }
}
```

Interface

Newton
Quem se prepara, não para.

- Uma interface especifica ***quais*** operações uma classe (ou um conjunto de classes) deve possuir, mas não especifica ***como*** elas devem ser implementadas.

- Exemplos:

  - Aparelho de rádio:

    - Possui um conjunto de operações padronizadas, tais como mudar de estação, ajustar o volume, escolher entre AM e FM

    - Mas diferentes rádios podem implementar as operações de maneiras diferentes, por exemplo com uso de botões ou comandos de voz.

Interface

- Carro

  - Um carro possui um conjunto de serviços que o motorista utiliza, por exemplo volante, caixa de câmbio, embreagem, acelerador e freio

  - Mas alguns desses serviços podem variar de carro para carro. Por exemplo, a forma de engatar a marcha ré pode diferenciar de carro para carro

Interface

- ***Interface*** de Java é um recurso que permite especificar os serviços de uma classe.

```
interface Forma {...}
```

- Uma interface declara:
  - métodos implicitamente públicos e abstratos (métodos não podem ser estáticos e também não podem ser finais):
  - campos implicitamente públicos, estáticos e finais.

Interface

- Objetos não podem ser criados diretamente a partir de uma interface.
- Classes que implementam uma interface devem implementar todos os métodos daquela interface e todos eles devem ser públicos.

```
class Circulo implements Forma {...}
```

- Uma interface pode ser implementada por várias classes.
- Uma classe pode implementar várias interfaces.

```
public class B extends A implements J,K
```

Interface

Newton
Quem se prepara, não para.

- **Interface é um caso especial de classe abstrata.**

- Interface costuma ser utilizada no lugar de classe abstrata quando não há implementação padrão alguma de método a herdar

Interface

- Pode haver hierarquia de interfaces

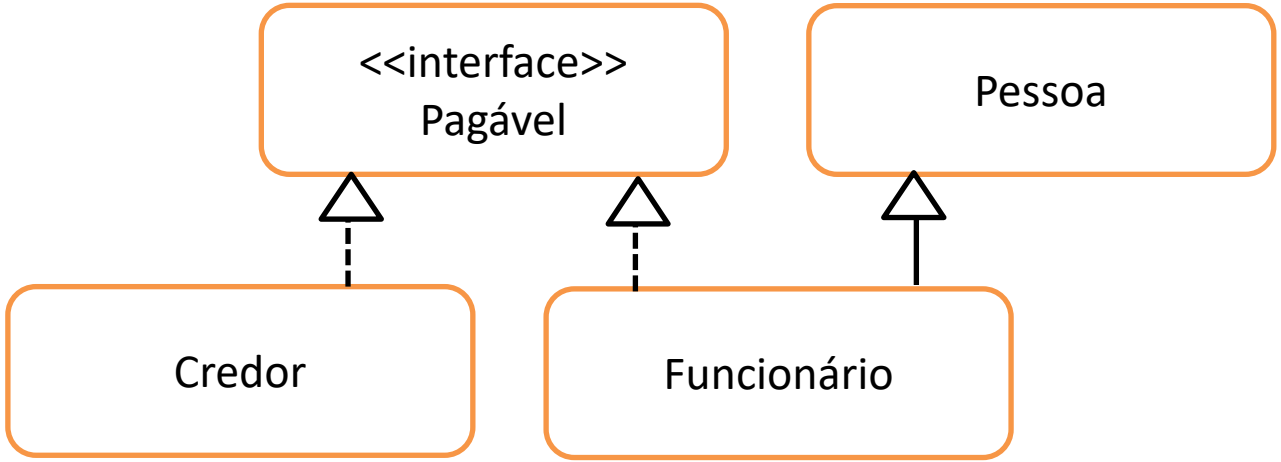
```
public interface K {
    int n=10;
}

public interface J extends K {
    public int f=4;

    public void m1();
}
```

Interface

- Exemplo:



Interface

```
public interface Pagavel{
     public abstract float getPagamento();
}
```

Interface - Exemplo

```
public class Pessoa {
    private String nome;
    private String cpf;

    public void setNome(String n){
      nome = n;
    }

    public String getNome(){
      return nome;
    }

    public void setCPF(String m){
      cpf = m;
    }

    public String getCPF(){
      return cpf;
    }
}
```

Interface - Exemplo

Newton
Quem se prepara, não para.

```
public class Funcionario extends Pessoa implements Pagavel{

    private float salario;
    private float comissao;
    private static final float salarioMinimo = 500;

    public int setSalario(float salario){
        int result = 0;
        if (salario > salarioMinimo)
            this.salario = salario;
        else
            result = -1;
        return result;
    }
```

Interface - Exemplo

```
    public float getSalario(){
      return this.salario;
    }

    public void setComissao(float comissao){
      if (comissao > 0)
         this.comissao = comissao;
    }

    public float getComissao(){
      return this.comissao;
    }

    //método implementado da interface Pagável

     public float getPagamento(){
       return this.salario + this.comissao;
     }

}
```

Interface - Exemplo

Newton
Quem se prepara, não para.

```
public class Main {

   public static void main(String[] args){

      Funcionario f = new Funcionario();

      f.setCPF("123");
      f.setComissao(100);
      f.setSalario(700);
      System.out.println(f.getPagamento());
   }
}
```

Referências

SILVA, Fabricio Machado da. **Paradigmas de programação**. SAGAH, 2019. ISBN digital: 9788533500426

Barnes, David e Kölling, M. **Programação Orientada a Objetos com Java**. São Paulo: Pearson Prentice Hall, 2004.

Deitel, H. M.; Deitel, P. J. **Java - Como Programar**. 6. ed. Prentice-Hall, 2005. Capítulo 4 e 5.

PRESSMAN, Roger S. **Engenharia de Software**. Rio de Janeiro: MacGraw Hill, 2002.