Programação Orientada a Objetos

+ Elementos básicos de Java

Profa.: Márcia Sampaio Lima

EST - UEA

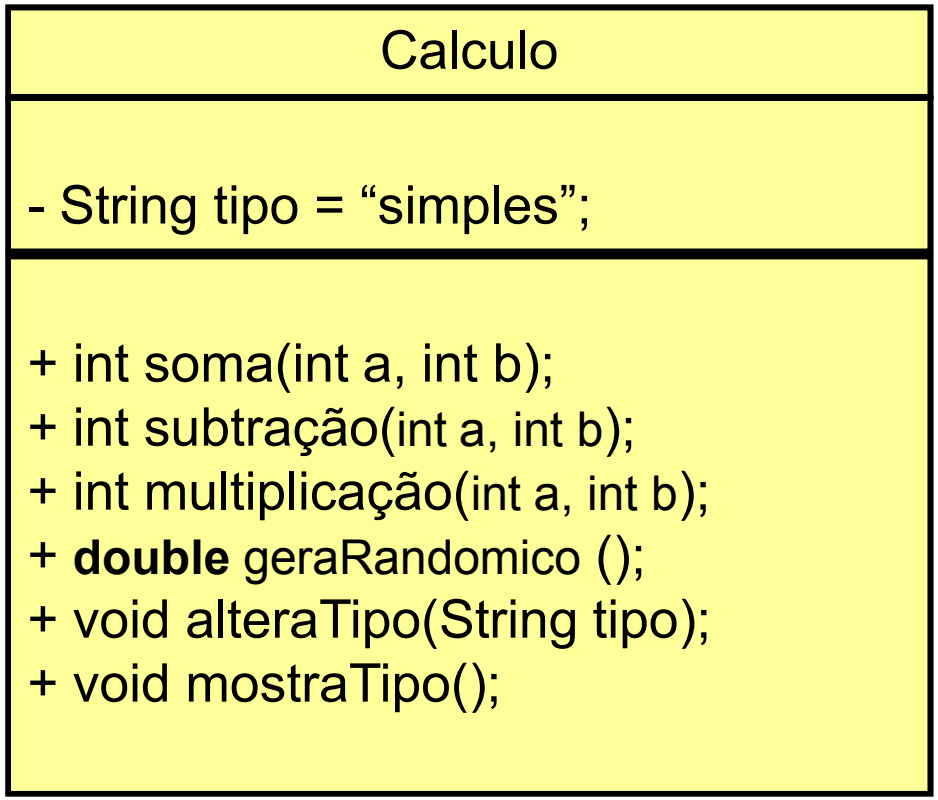
Ref.: Slides Prof. Flávio José Mendes Coelho, M.Sc.

Métodos

Métodos
Anatomia de um método

- Um método implementa a lógica de uma operação.
- Há dois tipos de métodos:
  - retornam valor;
  - não retornam valor;

Calculadora



Métodos
Anatomia de um método

- Método que retorna valor (com parâmetros):

```
public class Calculo {
  ...
  public int soma(int a, int b) {
    int s;
    s = a + b;
    return s;
  }
}
```

Métodos
Anatomia de um método

- Método que retorna valor (com parâmetros):

```
public class CalculadoraTeste {

   public static void main(String[] args) {
      Calculo c = new Calculo();
      int result = c.soma(3, 4);
      System.out.println(resul);
   }

}
```

Métodos
Anatomia de um método

- Método que retorna valor (com parâmetros):

```
public class CalculadoraTeste {

  public static void main(String[] args) {
     Calculo c = new Calculo();
     System.out.println(c.soma(3, 4));
  }

}
```

Métodos
Anatomia de um método

- Método que retorna valor (sem parâmetros):

```
package calcular;

public class Calculo {
   ....
   public double geraRandomico(){
       return Math.random();
   }
...
}
```

Métodos
Anatomia de um método

- Método que retorna valor (sem parâmetros):

```
package calcular;

public class CalculadoraTeste {

   public static void main(String[] args) {

       Calculo calculadora = new Calculo();
       System.out.println(calculadora.geraRandomico());

   }

}
```

Métodos

Anatomia de um método

- Método que **não** retorna valor (com parâmetros):

```
public class Calculo {
   private String tipo="simples";
  ...
   public void alteraTipo(String novoTipo){
      this.tipo = novoTipo;
   }

   public String getTipo() {
      return tipo;
   }

   public void setTipo(String tipo) {
      this.tipo = tipo;
   }
}
```

Métodos
Anatomia de um método

- Método que **nao** retorna valor (com parâmetros):

```
public static void main(String[] args) {

   Calculo calculadora = new Calculo();

   calculadora.setTipo("cientifico");

   System.out.println(calculadora.getTipo());

}
```

Métodos
Anatomia de um método

- Método que **não** retorna valor (com parâmetros):

```
public class Calculo {
   ...
   public void mostraSoma(int a, int b) {
      System.out.println( a + b);
   }
}
```

Métodos
Anatomia de um método

- Método que **não** retorna valor (com parâmetros):

```
public class CalculadoraTeste {
  public static void main(…) {
     Calculo calculadora = new
  Calculo();
     calculadora.mostraSoma(4 , 6);
  }
}
```

Métodos
Anatomia de um método

- Método que **não** retorna valor (sem parâmetros):

```
public class Calculo {

    …..
    public void mostraTipo(){
            System.out.println("Tipo: "+this.tipo);
    }
}
```

Métodos
Anatomia de um método

- Método que **não** retorna valor (sem parâmetros):

```
public class CalculadoraTeste {
  public static void main(…) {
     Calculo calculadora = new Calculo();
     calculadora.setTipo("cientifico");
     calculadora.mostraTipo();
  ….
  }
}
```

Polimorfismo

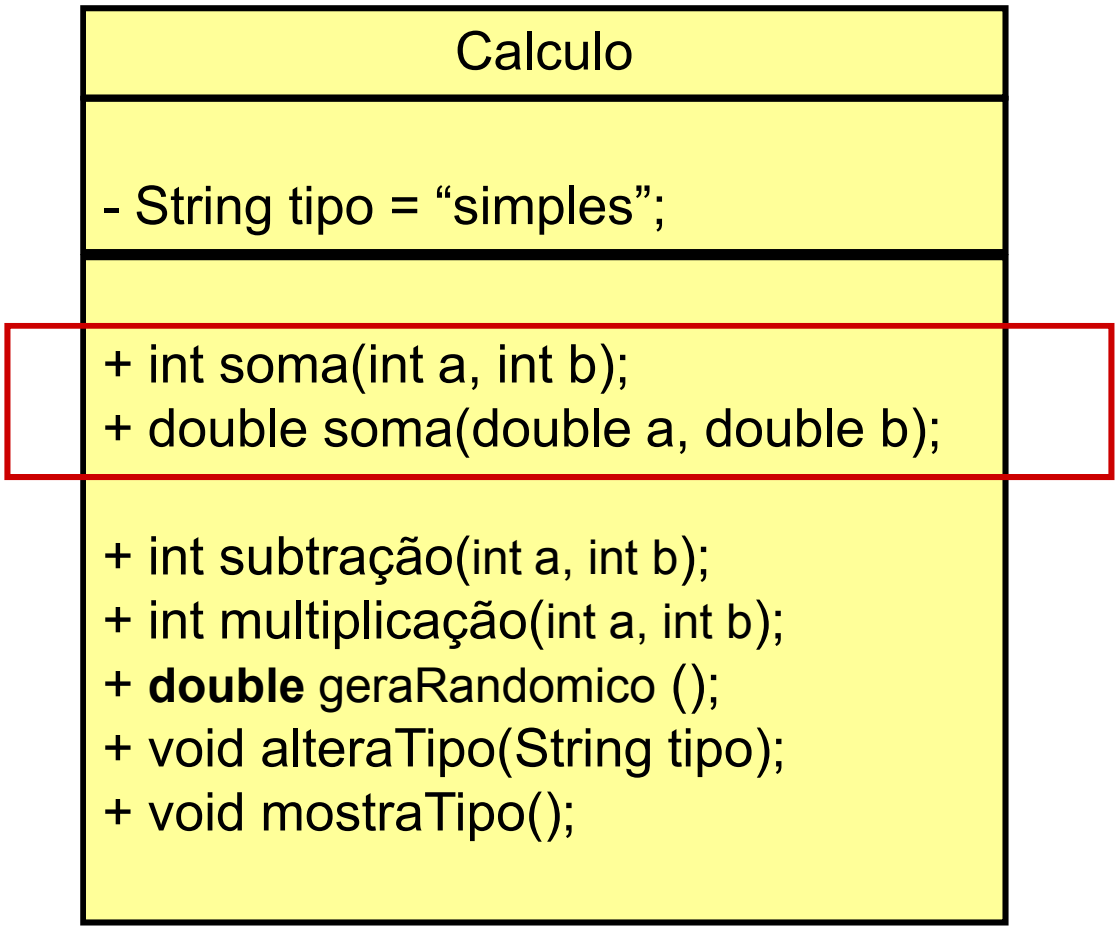
- Tipo: Sobrecarga = *Overloading*
  - Métodos, na mesma classe, com mesmo nome, funcionalidades similares.
  - Mesmo nome, mas assinaturas diferentes.
    - Número de argumentos.
    - Tipo dos argumentos.
  - Não há garantias que os métodos possuam o mesmo comportamento.

Polimorfismo

- Tipos:
  - Métodos homônimos, porém com parâmetros diferentes.
  - O método executado depende do tipo de dado passado como parâmetro.

```
   int soma (int x, int y);
double soma (double x, double y);
String soma (String x, String y);
```

Calculadora



Sobrecarga de Métodos

```
package calcular;

public class Calculo {

   .....
   public int soma (int a, int b){

      return (a+b);
   }

   public double soma (double a, double b){

      return (a+b);
   }

}
```

Execução

```
public static void main(String[] args) {

   Calculo calculadora = new Calculo();
   double opD1,opD2;
   int opI1, opI2;

   opD1 = 2.8;
   opD2 = 0.6;
   opI1 = 1;
   opI2 = 5;

   System.out.println (calculadora.Soma(opD1, opD2));
   System.out.println (calculadora.Soma(opI1, opI2));

}
```

Sobrecarga de Métodos

- Já utilizamos a sobrecarga de métodos outras vezes. Onde?

Sobrecarga de Métodos

- Já utilizamos a sobrecarga de métodos outras vezes. Onde?

  Métodos Construtores!!!

Sobrecarga de Métodos

```
public class Gato {

    String nome;
    double peso;
    String raca;
    int idade;

    public Gato() {
        this.nome="";
        this.peso=0;
        this.raca="";
        this.idade=0;
    }
    public Gato(String Pnome,double Ppeso,String raca,int Pidade){
        this.nome=Pnome;
        this.peso=Ppeso;
        this.raca=raca;
        this.idade=Pidade;
    }
```

.....

Bibliografia

- Fowler, Martin. UML Essencial. 2o. ed. – Porto Alegre: Bookman, 2000.
- Deitel, Harvey. Java - Como Programar. 6a. ed. Porto Alegre: Bookman, 2006.
- Eckel, Bruce. Thinking in Java. 3a. edição. Prentice-Hall, 2001.