

# Programação Orientada a Objetos

*Métodos*

Rone Ilídio  
Thiago Oliveira



# Métodos

- São módulos de programas, ou seja, trechos de código com determinada função.
- Cada método deve possuir um nome (identificador).
- Declaração básica de um método:

```
valor_retorno nome_metodo(tipo p1, tipo p2, ...)  
{  
    . . . sequência de comandos . . .  
}
```



```
import javax.swing.*;
```

```
public class CalculaQuadrado extends JApplet
```

```
{
```

```
    public void init()
```

```
{
```

```
    String saida = "";
```

```
    for (int contador = 1; contador<=10; contador++)
```

```
{
```

```
        int resultado = quadrado(contador);
```

```
        saida = saida + contador + "*" + contador + "=" + resultado + "\n" ;
```

```
}
```

```
    JOptionPane.showMessageDialog(null, saida);
```

```
}
```

```
public int quadrado(int y)
```

```
{
```

```
    return y * y;
```

```
}
```

```
}
```



# Chamada de métodos

- Métodos da mesma classe:

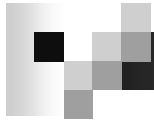
- ☐ nome\_método(lista\_parâmetros)
- ☐ ou, se não tiver parâmetros: nome\_método()
- ☐ ex: quadrado(contador)

- Métodos de objetos:

- ☐ nome\_objeto.nome\_método();
- ☐ ex: pessoa.setNome("Maria");

- Métodos de classes importadas - *static*:

- ☐ nome\_classe.nome\_método(lista\_parâmetros)
- ☐ ex: JOptionPane.showMessageDialog(null, "Olá!");
- ☐ Métodos static só podem chamar métodos e variáveis static!



# Métodos da Classe Math

- Os métodos dessa classe permitem realizar certos cálculos matemáticos comuns.
- Está presente no pacote `java.lang`
  - Ou seja, não precisa ser importada.
- Seus principais métodos são:



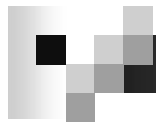
# Métodos da Classe Math

- `exp(x)`: método exponencial  $e^x$
- `log(x)`: logaritmo de  $x$  na base  $e$
- `abs(x)`: retorna o valor absoluto
  - `abs(-1.34) = 1.34` (int, long, float, double)
- `ceil(x)`: retorna o menor inteiro maior que  $x$ 
  - `ceil(9.2) = 10`
- `floor(x)`: retorna o maior inteiro menor que  $x$ 
  - `floor(9.2) = 9`
- `round(x)`: arredonda para o inteiro mais próximo



# Métodos da Classe Math


- `max(x,y)`: retorna o maior entre  $x$  e  $y$
- `min(x,y)`: retorna o menor entre  $x$  e  $y$
- `sqrt(x)`: retorna a raiz quadrada de  $x$
- `pow(x,y)`: retorna  $x^y$
- `cos(x)`: retorna o co-seno de  $x$ , em radianos
- `sin(x)`: seno de  $x$ , em radianos
- `tan(x)`: tangente de  $x$ , em radianos



# Exercício

- Crie um applet que receba 3 valores (double) e retorne o maior deles.
  - ☐ Receba e converta 3 valores para double.
  - ☐ Crie um método denominado Maximo, que será responsável em calcular qual é o maior valor.
  - ☐ Utilize o método `Math.max(x,y)` nesse método que será criado.
  - ☐ Exiba o resultado utilizando `JOptionPane`.





```
import javax.swing.*;

public class Maximo extends JApplet {
    public void init() {
        String num1, num2, num3;
        num1 = JOptionPane.showInputDialog("Entre com o 1º número:");
        num2 = JOptionPane.showInputDialog("Entre com o 2º número:");
        num3 = JOptionPane.showInputDialog("Entre com o 3º número:");
        double n1, n2, n3;
        n1 = Double.parseDouble(num1);
        n2 = Double.parseDouble(num2);
        n3 = Double.parseDouble(num3);
        JOptionPane.showMessageDialog(null, "Maior é" + maximo(n1,n2,n3));
    }
    public double maximo(double num1, double num1, double num3) {
        double maior = Math.max(num1, num2);
        return Math.max(maior, num3);
    }
}
```



# Coerção de argumentos

- É a transformação automática de um tipo de dados para outro.
  - Ex: `Math.sqrt(x)`, espera que `x` seja `double`.
  - Se `x` for inteiro, automaticamente ocorre a conversão de `x` para `double` e o método executa normalmente!
- Aceita somente de tipos inferiores para tipos superiores.



# Coerção de argumentos

- double → nenhuma
- float → double
- long → float, double
- int → float, double, long
- char → float, double, long, int
- short → float, double, long, int
- byte → short, float, double, long, int
- boolean → nenhuma



# Conversão de tipos

- Para converter de tipos superiores para inferiores segue-se o exemplo:
  - `int x = (int) Math.max(1.0, 3.5);`
  - O método retorna um `double` e esse `double` é convertido para inteiro.
- Permite outras operações:
  - `int x = (int) Math.max(1.0, 3.5) / 2;`
  - Primeiro o método é executado, depois ocorre a conversão e por último a divisão.



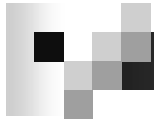
# Números Aleatórios

- O método `Math.random()` retorna um número `x` (double), de forma que  $0 \leq x < 1$ .
- Para produzir inteiros em um determinado intervalo ( $0 \rightarrow L$ ) deve-se ***escalonar*** o resultado desse método da seguinte forma:
  - `int valor = (int) (Math.random() * L);`
  - onde `L` é o limite do intervalo esperado;
  - trata-se o limite superior.




# Números Aleatórios

```
public class Randomico{  
    public static void main(String args[]){  
        int num=0;  
        for (int i=1; i<=10 ; i++){  
            num = 1 + (int)(Math.random() * 10);  
            System.out.println(num);  
        }  
    }  
}
```




# Exercício

- Crie um aplicativo que simule a jogada de um dado e receba um inteiro do usuário para comparar se é igual ao número gerado.
  - ☐ Crie um método que, quando chamado, gere números inteiros aleatórios de 1 a 6.
  - ☐ O aplicativo deve oferecer 10 chances para o usuário acertar.
  - ☐ Toda vez que o usuário entrar com um novo valor, um novo número aleatório é gerado.



```
import javax.swing.JOptionPane;
public class Dado{
    public static int jogar(){
        return 1 + (int)(Math.random()*6);
    }
    public static void main(String args[]){
        for (int i=1; i<=10 ; i++){
            String entrada = JOptionPane.showInputDialog("Digite número (1-6)");
            int numero = Integer.parseInt(entrada);
            int resultado = jogar();
            if (numero == resultado) {
                JOptionPane.showMessageDialog(null,"Você acertou!");
            } else {
                JOptionPane.showMessageDialog(null,"Número era: "+resultado);
            }
        }
    }
}
```





```
import javax.swing.JOptionPane;
import javax.swing.JApplet;
public class Dado extends JApplet{
    public static int jogar(){
        return 1 + (int)(Math.random()*6);
    }
    public void init(){
        for (int i=1; i<=10 ; i++){
            String entrada = JOptionPane.showInputDialog("Digite número (1-6)");
            int numero = Integer.parseInt(entrada);
            int resultado = jogar();
            if (numero == resultado) {
                JOptionPane.showMessageDialog(null,"Você acertou!");
            } else {
                JOptionPane.showMessageDialog(null,"Número era: "+resultado);
            }
        }
    }
}
```



# Exercícios

- Faça um applet que crie um número aleatório entre 0 e 100, e que ofereça 3 chances para o usuário acertar tal número.
  - ☐ Toda vez que o usuário entrar com uma tentativa, o applet deve retornar se o número inserido pelo usuário é maior ou menor que o número gerado pelo programa.
- Crie um applet onde o usuário insere um número (double) e o applet retorne a parte fracionária desse número.
- Crie um applet que gere 5000 números aleatórios de 1 a 5. Ao final, imprima quantas vezes cada número foi gerado.



# Exercícios

- Escreva o método abaixo, que deve retornar o resultado de uma operação entre a e b, definida pelo usuário pelo parâmetro opcao:
  - ☐ public double calculo(double a, double b, int opcao)
  - ☐ se digitar 1  $\rightarrow a * b$
  - ☐ se digitar 2  $\rightarrow a / b$
  - ☐ se digitar 3  $\rightarrow a^b$
  - ☐ se digitar 4  $\rightarrow \text{sqrt}(a+b)$
  - ☐ utilize switch-case para tratar esses valores.
- Crie um applet que utilize esse método e ofereça ao usuário um menu contendo as 4 opções.



# Classe String

- Não é exatamente um vetor de caracteres, é uma classe que possui como atributo um vetor de caracteres.
  - Assim, cada String é um objeto.
- Possui vários métodos que auxiliam no tratamento de textos.



# String - Métodos

## ■ Comparação

- ☐ String entrada...
- ☐ if (entrada.**equals**("POO")) {...}
- ☐ else if (entrada.**isEmpty**()) {...}

## ■ Tamanho

- ☐ int tamanho = entrada.**length**();

## ■ Maiúsculo e minúsculo

- ☐ **toUpperCase()** ou **toLowerCase()** retornam a String toda em maiúsculo ou em minúsculo.



# String - Métodos

## ■ Convertendo String em números

- ☐ `int inteiro = Integer.parseInt("12");`
- ☐ `double pi = Double.parseDouble("3.14");`

## ■ Concatenando

- ☐ `double x = 10.7;`
- ☐ `String a = "" + x;`
- ☐ `String b = "";`
- ☐ `a.concat(b);` ou `a = a+b;` ou `a +=b;`

## ■ Substituindo caracteres ou substrings

- ☐ `System.out.println(a.replace(".", ","));`



# String - Métodos

## ■ Caractere na posição

- `String poo = "Programacao Orientada a Objetos";`
- `char g = poo.charAt(3);`
- Primeiro caractere: **`charAt(0);`**

## ■ Substrings

- `String programacao = poo.substring(0, 10);`
- `String palavras[] = poo.split(" ");`
- `if (poo.startsWith("Prog")) {`  
    `JOptionPane.showMessageDialog(null, palavras[0]);`  
    `}`



# String - Métodos

- Procurando caracteres ou de substrings:
  - `String poo = "Programacao Orientada a Objetos";`
  - `int posicao = poo.indexOf("o");`
  - `JOptionPane.showMessageDialog(null, "" + posicao);`
  
  - `int ultimo = poo.lastIndexOf("o");`
  - `JOptionPane.showMessageDialog(null, "" + ultimo);`
  
  - Valores impressos: 2 e 29 respectivamente.