

# Laboratório de Programação Orientada a Objetos

Prof. Dr. Rodrigo Plotze

[rodrigoplotze@gmail.com](mailto:rodrigoplotze@gmail.com)

# **JAVA – LINGUAGEM E PLATAFORMA**

# Java - Linguagem e Plataforma

- Histórico da Linguagem

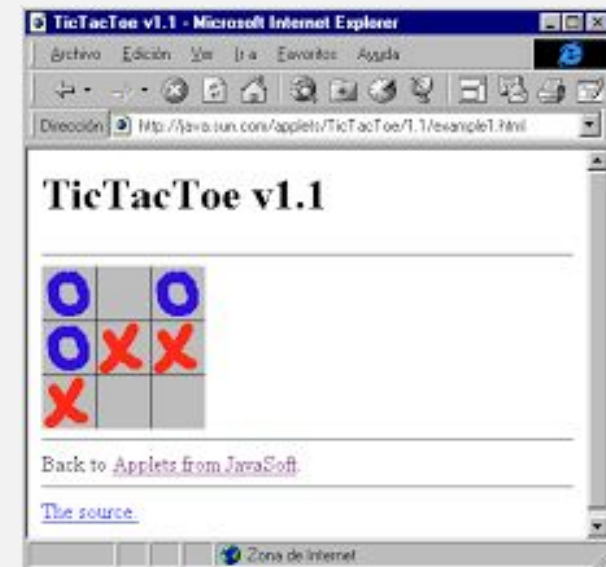
- **1991 - Projeto Green**

- Nascimento da linguagem Java (*James Gosling*)
    - Uma nova linguagem de programação
    - Utilização em pequenos dispositivos eletrônicos inteligentes

- Criada pela **Sun Microsystems**

- **1995** Java adaptado para Internet

- Applets

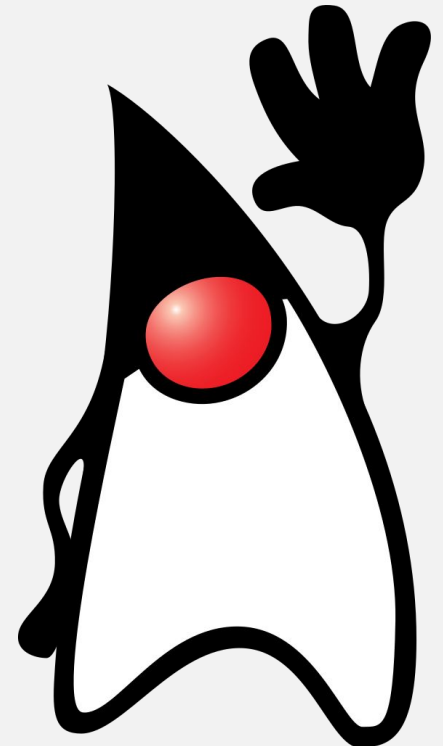


# Java - Linguagem e Plataforma

- Histórico da Linguagem
  - **2004** 3 milhões de desenvolvedores em todo o mundo.
  - **2006/2007** Grande parte do código passou a ser aberto, Software Livre (GNU/GPL)
  - **2008**
    - **Sun** é adquirida pela **Oracle** por **7.4 bilhões** de dólares.

*Mascote da linguagem Java*

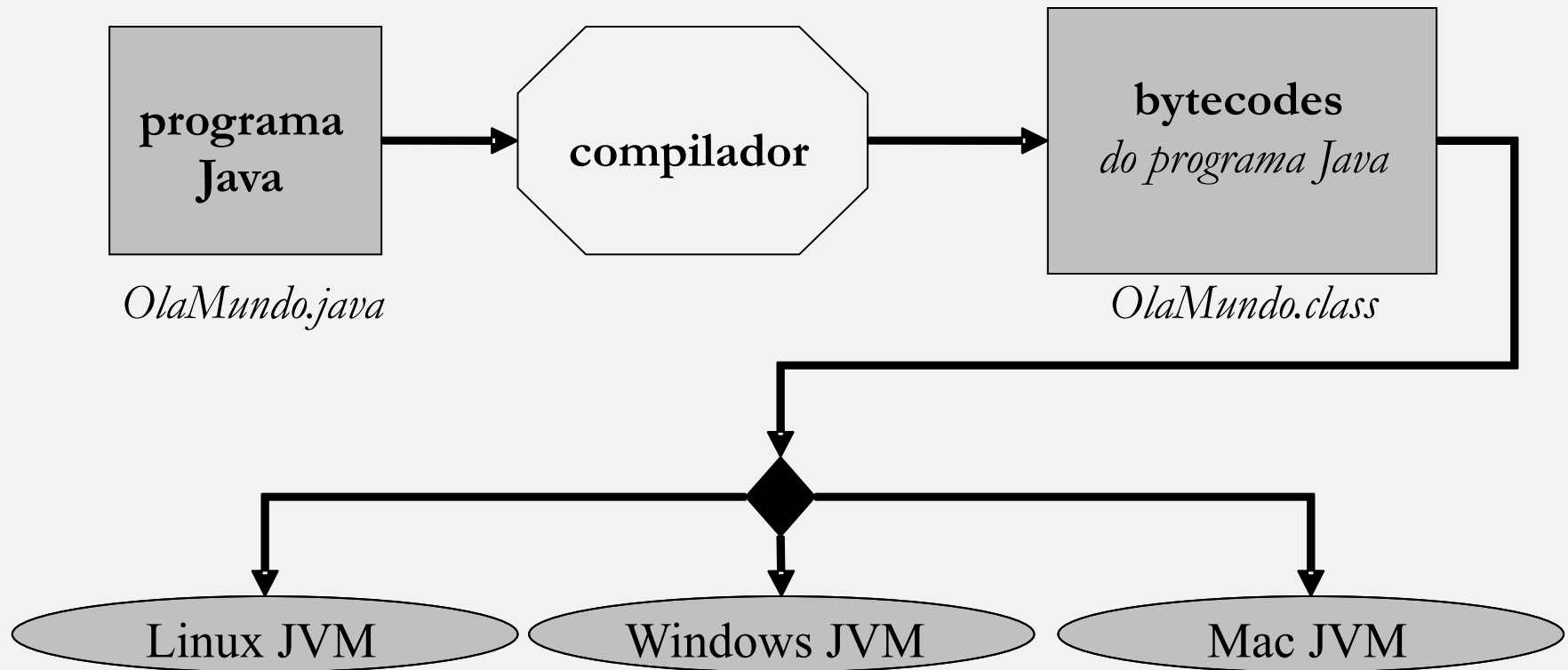
**DUKE**



# Java - Linguagem e Plataforma

- Máquina Virtual Java ou ***Java Virtual Machine*** (JVM)
  - Permite que aplicações Java “rodem” independente da plataforma na qual foram desenvolvidas
    - Multiplataforma
  - Cada Sistema Operacional (SO) possui sua própria JVM
  - Em linhas gerais funciona como “um computador dentro do seu computador”

# Java - Linguagem e Plataforma



# Java - Linguagem e Plataforma

- Processo de Compilação/Execução de um programa Java
  - Compilador: ***javac***
  - Interpretador: ***java***
- Exemplo
  - ***javac olamundo.java***
    - Produz o arquivo ***olamundo.class***
  - ***java olamundo***

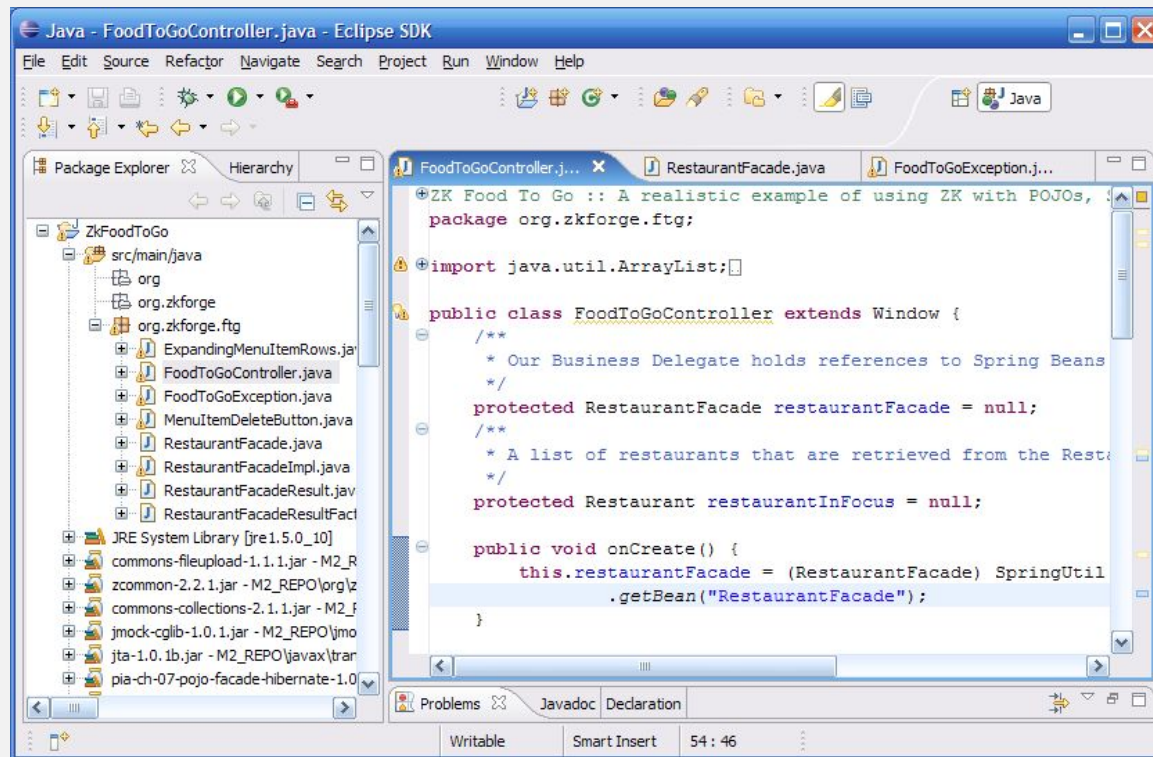
# Java - Linguagem e Plataforma

- O que você precisa para programar **Java**?
  - **Java Development Kit (JDK)**
    - **Java SE – Standard Edition**
      - Aplicações Desktop
    - **Java EE – Enterprise Edition**
      - Web Services, Aplicações Web,, etc.
    - **Java ME – Micro Edition**
      - Aplicações Móveis



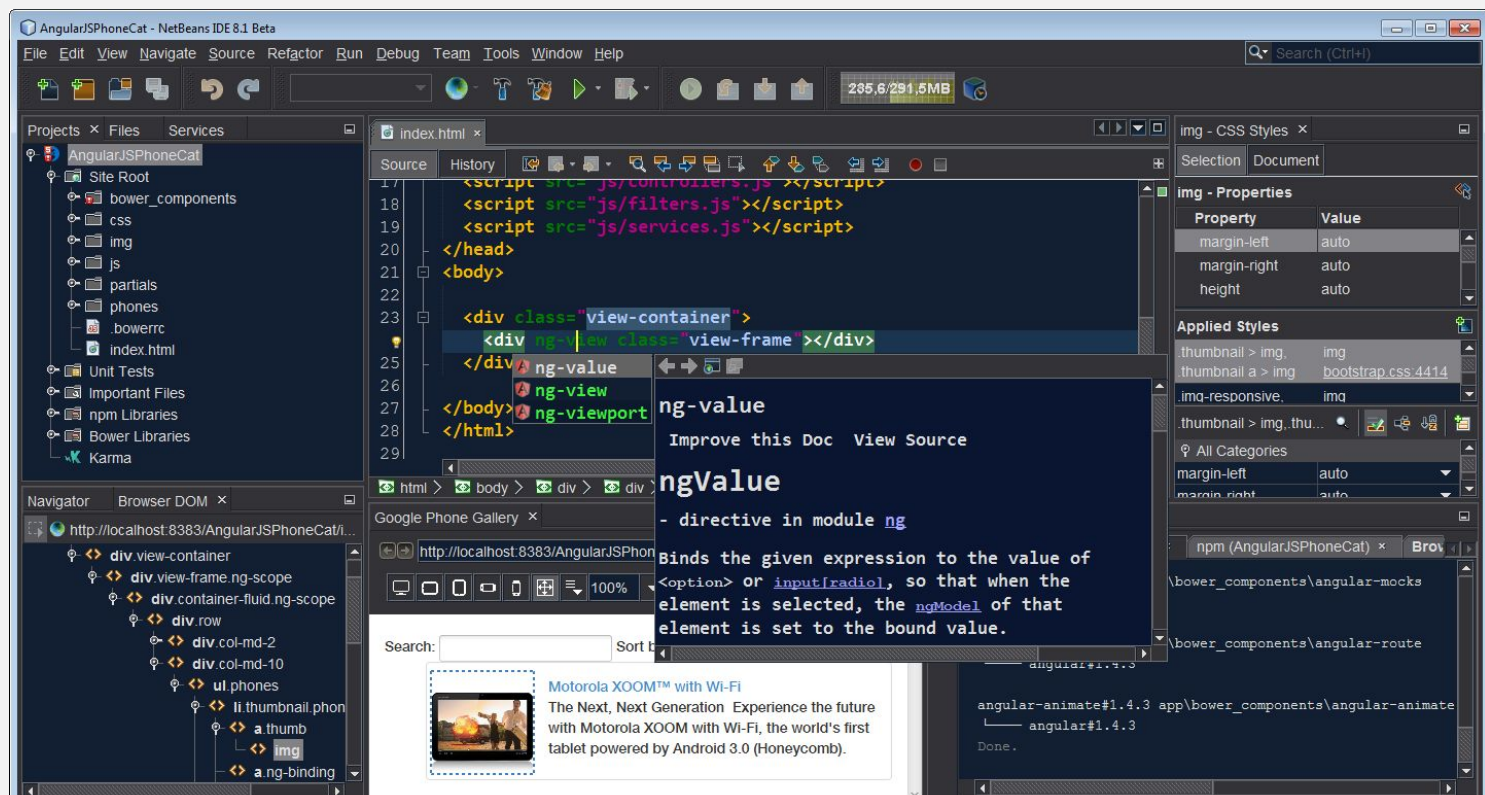
# Java - Linguagem e Plataforma

- O que você precisa para programar **Java**?
  - Ambiente de Programação (IDE)
    - Eclipse: <http://www.eclipse.org/>



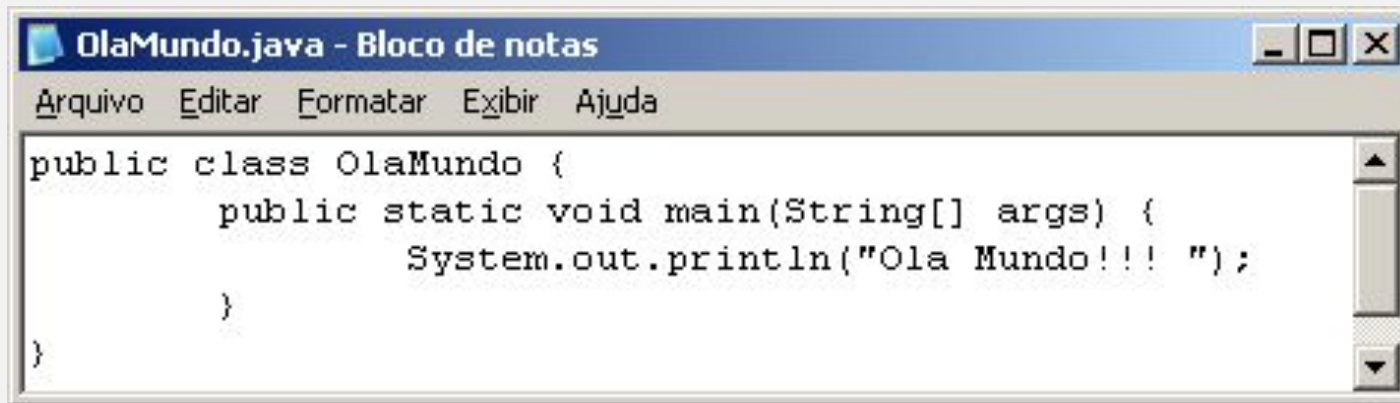
# Java - Linguagem e Plataforma

- O que você precisa para programar **Java**?
  - Ambiente de Programação (IDE)
  - NetBeans: <http://www.netbeans.org/>



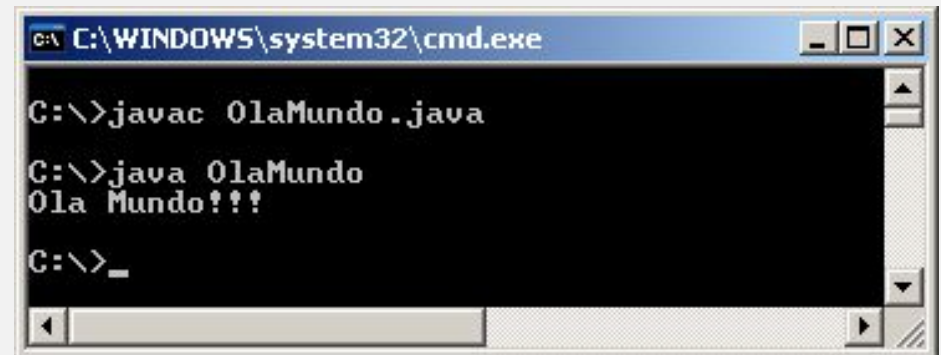
# Java - Linguagem e Plataforma

- O que você precisa para programar **Java**?
  - Ambiente de Programação (IDE)
    - Bloco de Notas



```
OlaMundo.java - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda

public class OlaMundo {
    public static void main(String[] args) {
        System.out.println("Ola Mundo!!! ");
    }
}
```



```
C:\WINDOWS\system32\cmd.exe

C:\>javac OlaMundo.java

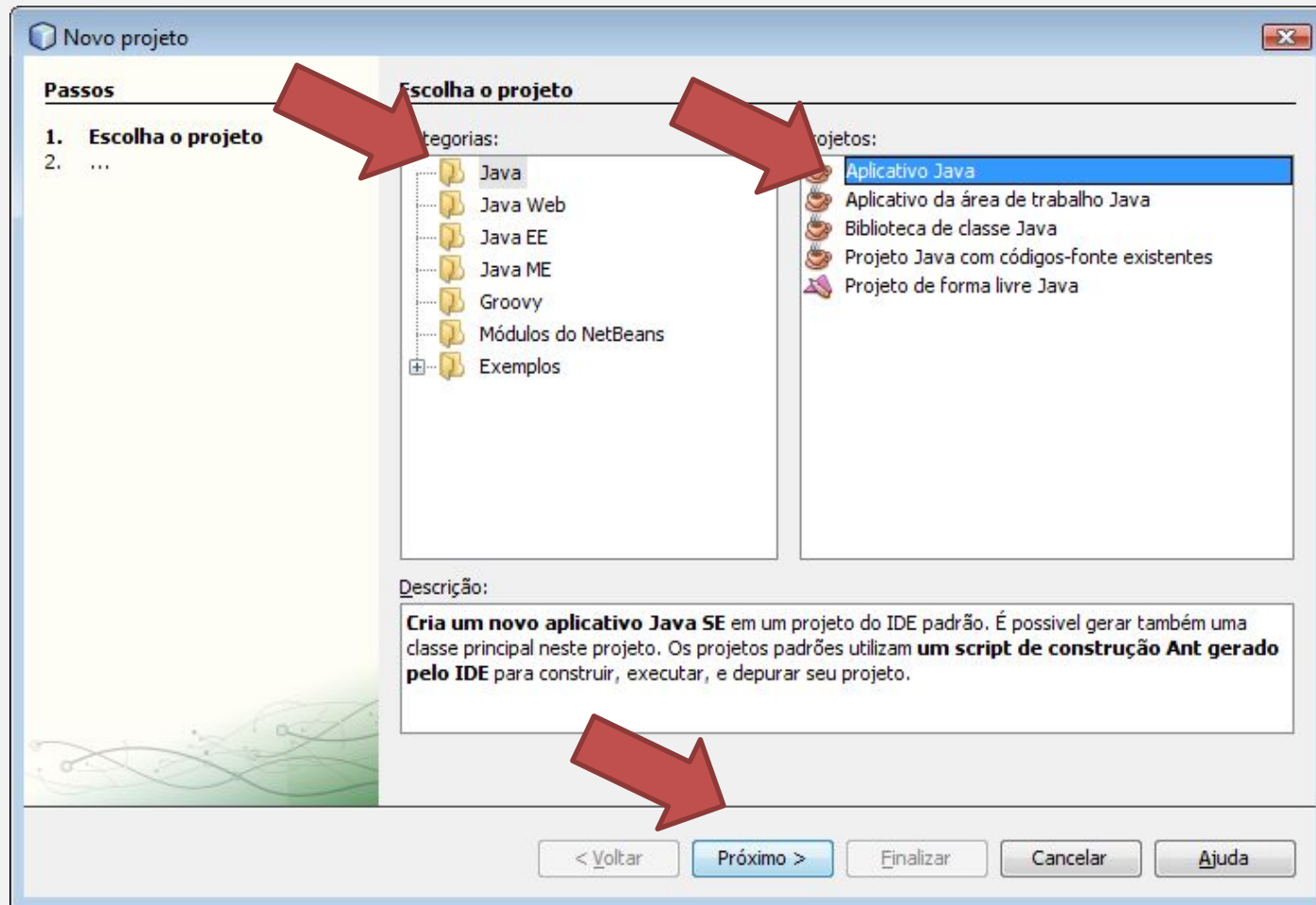
C:\>java OlaMundo
Ola Mundo!!!

C:\>_
```

# NETBEANS

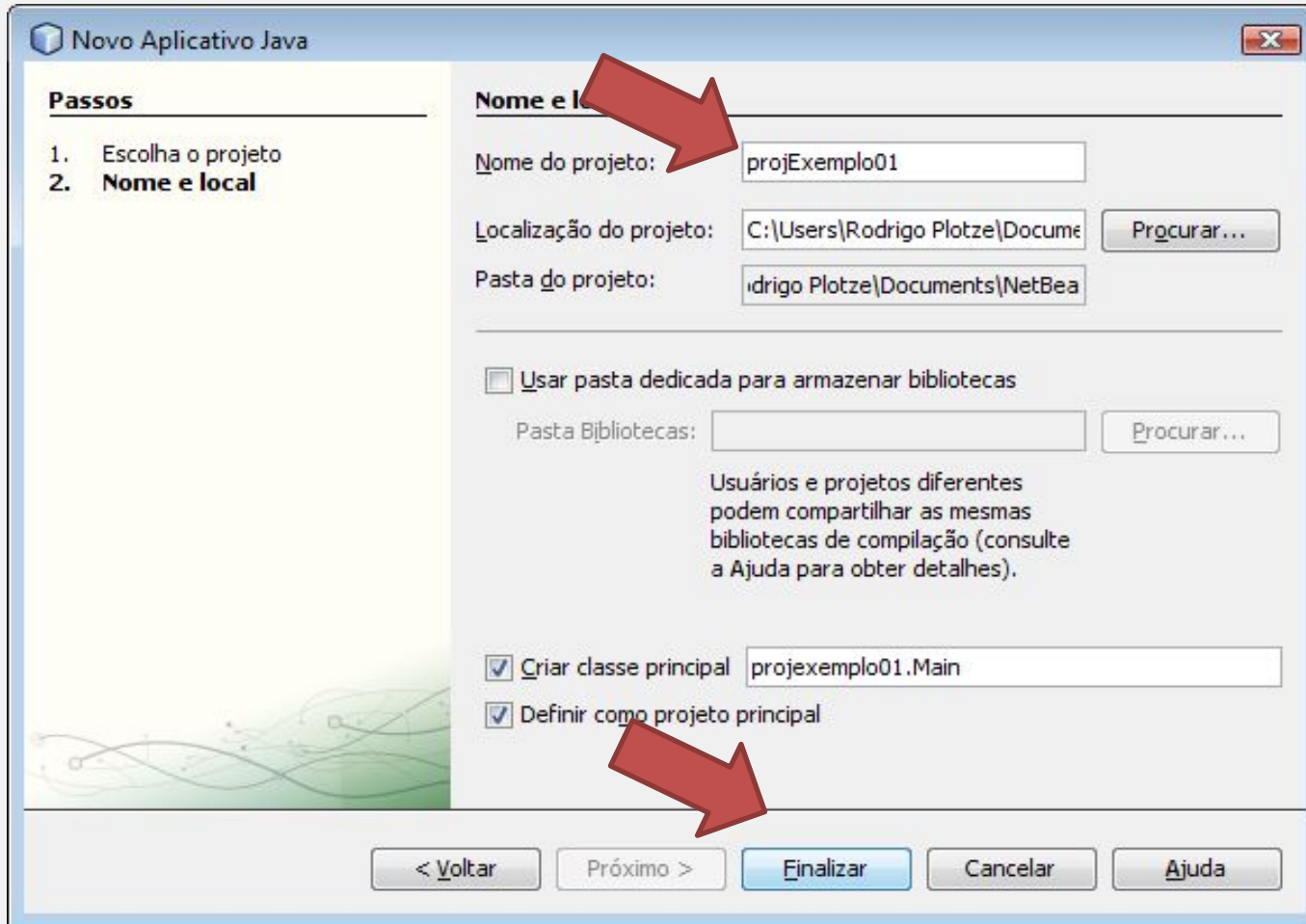
# NetBeans

## ■ *Arquivo > novo projeto*



# NetBeans

## ▪ *Definição do nome do projeto*



**Novo Aplicativo Java**

**Passos**

1. Escolha o projeto
2. **Nome e local**

**Nome e local**

Nome do projeto:

Localização do projeto:  Procurar...

Pasta do projeto:

☐ Usar pasta dedicada para armazenar bibliotecas

Pasta Bibliotecas:  Procurar...

Usuários e projetos diferentes podem compartilhar as mesmas bibliotecas de compilação (consulte a Ajuda para obter detalhes).

☒ Criar classe principal

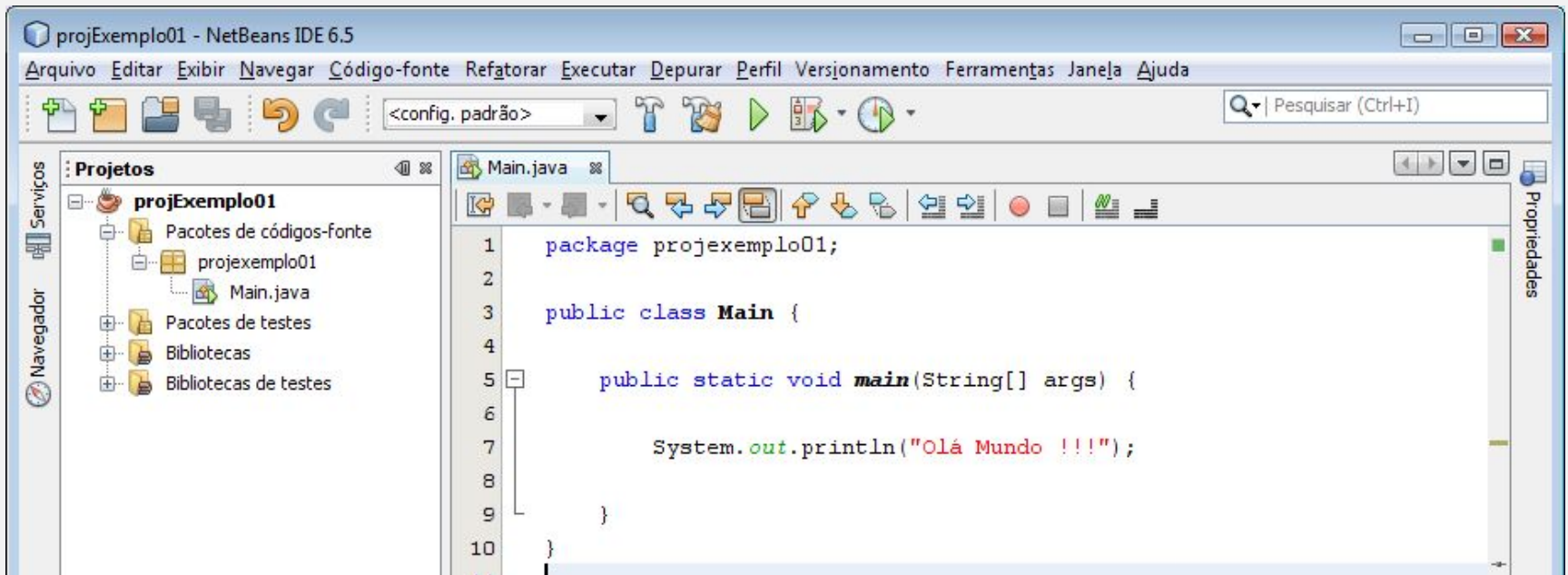
☒ Definir como projeto principal

< Voltar Próximo > Finalizar Cancelar Ajuda



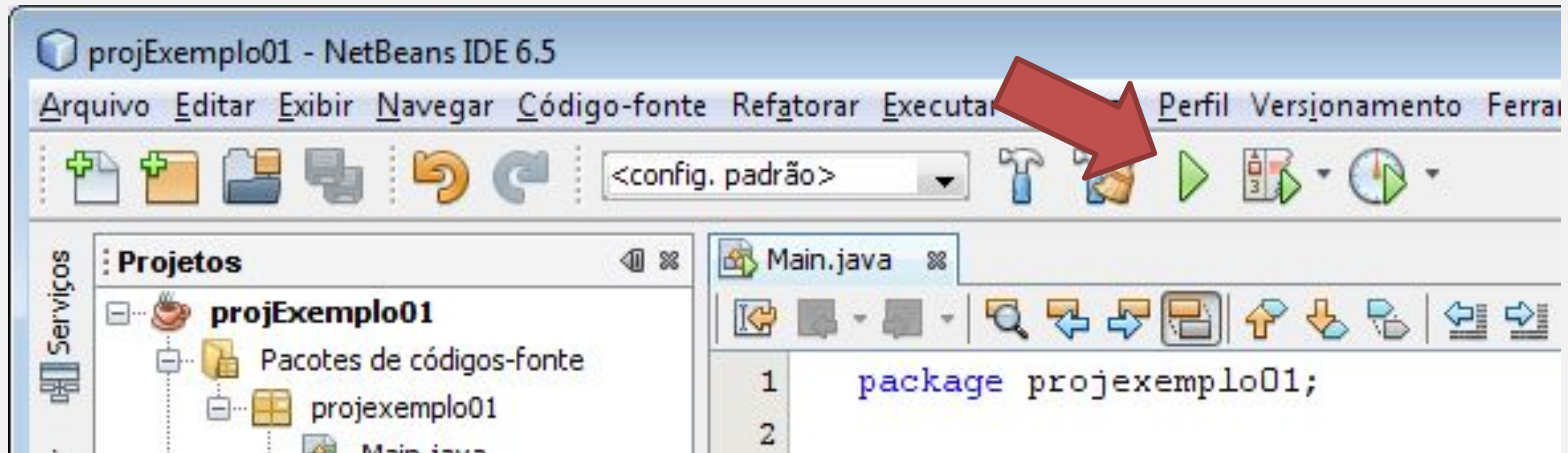
# NetBeans

## ■ *Novo projeto*



# NetBeans

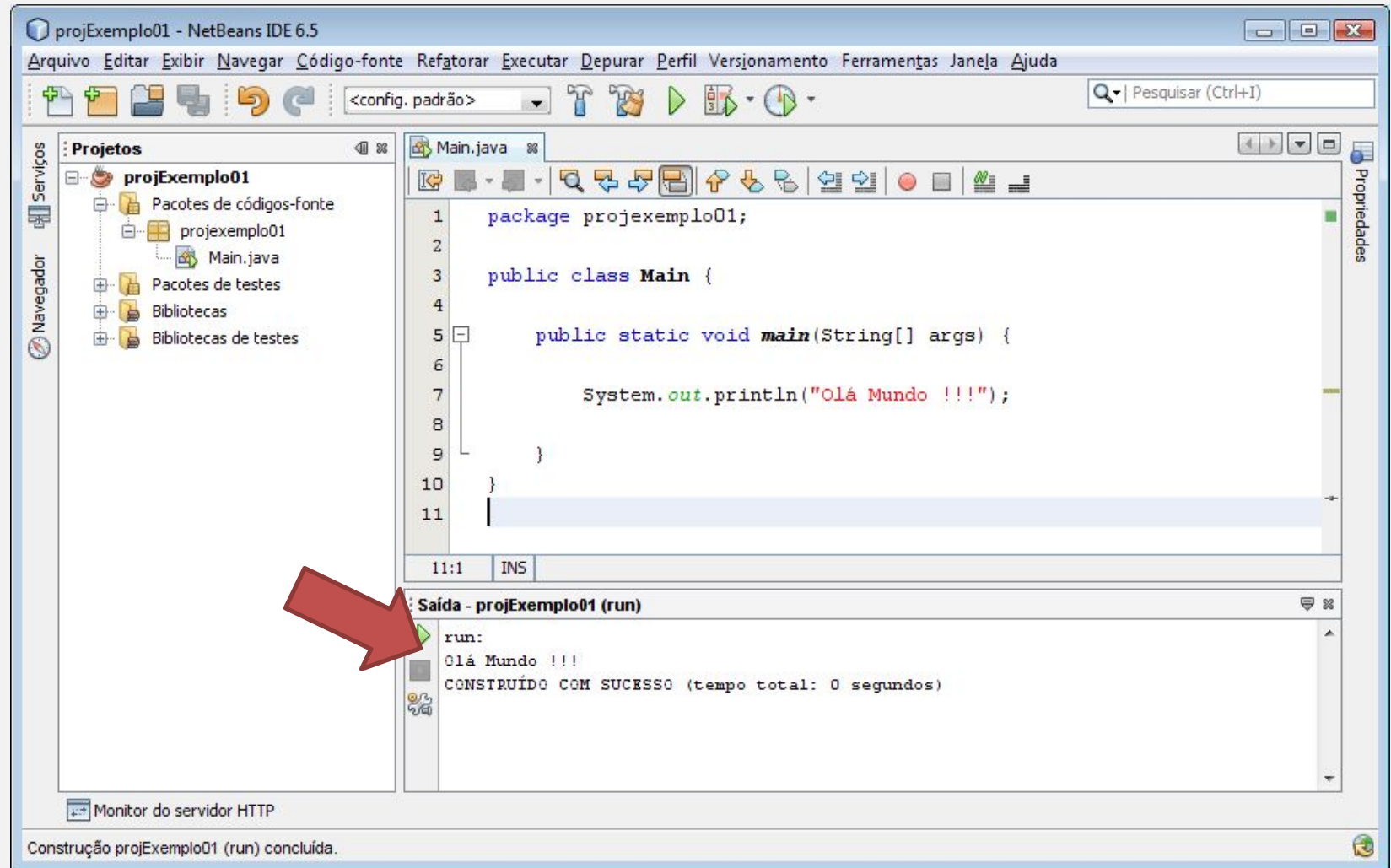
- Executando Aplicações



- Ou, pressione a tecla **F6**
- Ou, pressione a combinação **Shift+F6**



# NetBeans

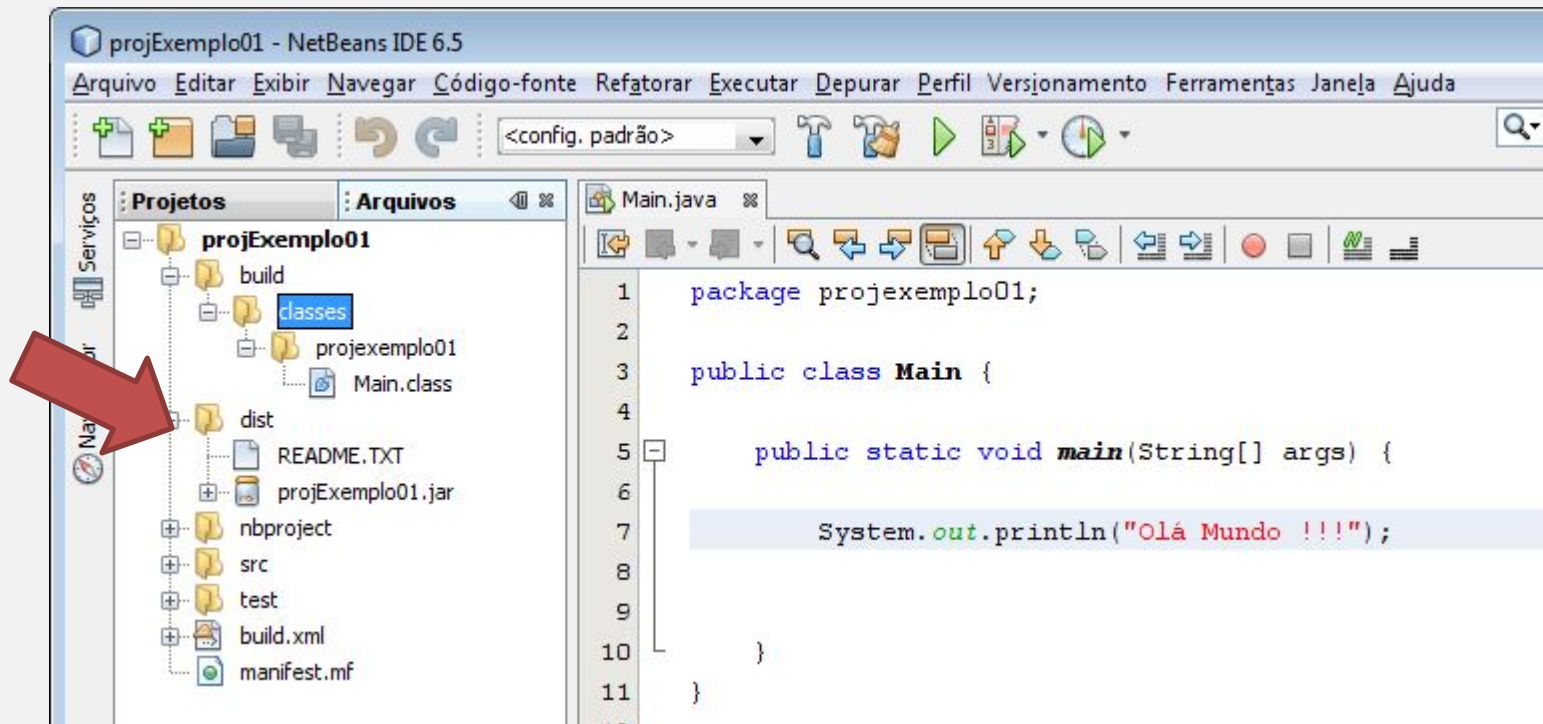


# NetBeans

- Saída
  - Janela □ Saída □ Saída (CTRL+4)
- Tarefas
  - Janela □ Tarefas (CTRL+6)

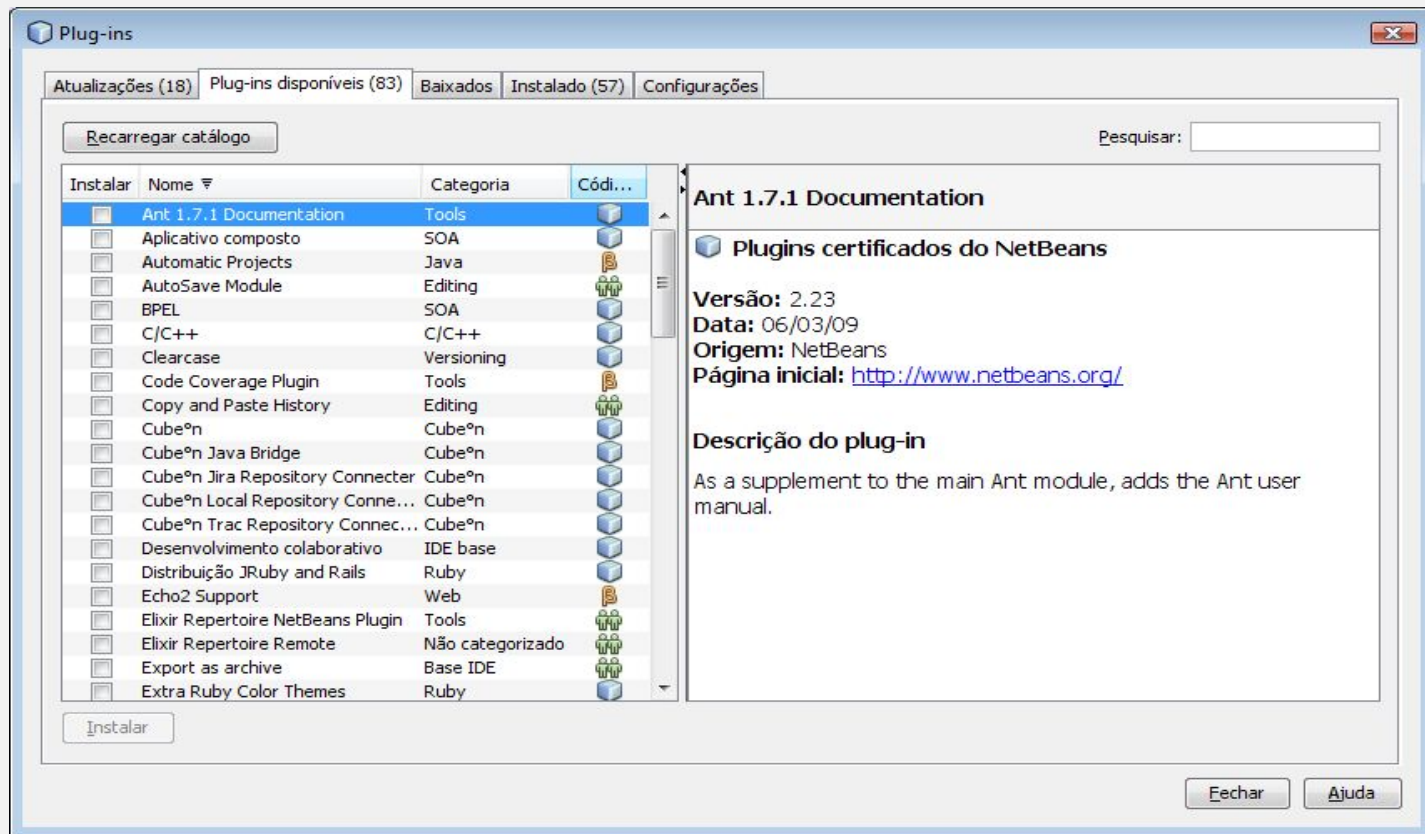
# NetBeans

- Build e Deploy da Aplicação
  - Janela ☐ Arquivos
  - Executar ☐ Limpar e Construir



# NetBeans

- Plugins
  - Ferramentas □ Plugins



# NetBeans

## ■ Atalhos do Editor

- Code Completion
  - CTRL+Space
  - Exemplo: IF, FOR
- Code Generation Dialogs
  - ALT+INSERT
- Duplicar Linhas
  - CTRL+SHIFT + Up ou Down
- Mover Linhas
  - ALT+SHIFT + Up ou Down

- Renomear  
CTRL+R  
Exemplo: Variáveis

# FUNDAMENTOS

# Fundamentos

- Estrutura Básica

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

## Convenções

Todos os nomes de classes em Java iniciam com uma letra maiúscula e tem uma letra maiúscula para cada palavra no nome da classe.

Ex: **NomeDaClasse**

# Fundamentos

- Estrutura Básica de um Programa Java

- ***Importante!!!***

- O nome do arquivo .java ***deve*** ter o mesmo nome da classe.

- Exemplo:

- public class **NomeDaClasse** {  
    }  
}

- Nome do arquivo □ **NomeDaClasse.java**

## Convenções

Java é Case Sensitive, ou seja, existe diferença entre letras maiúsculas e minúsculas. Assim, **NomeDaClasse** é *diferente* de **nomedaclasses** .



# Fundamentos

- Estrutura Básica de um Programa Java
  - **public class NomeDaClasse**
    - Define o início da classe, todos os métodos que fazem parte da classe devem ser definidos entre o início  
“{” (abre chave) e o fim “}” (fecha chave).

# Fundamentos

- Estrutura Básica de um Programa Java
  - `public static void main(String[] args)`
    - Determina o ponto de início da execução da classe
    - É possível escrever classes sem o métodos ***main***
      - No entanto para utiliza-lá é preciso instanciar esse classe em um outra classe que possua o método main
    - Um projeto pode ter um ***único*** método ***main***

# Fundamentos

- Os pacotes ou ***packages*** Java são um conjunto pré-definido de classes
- Cada pacote tem um grupo de classes que possui algum tipo de relação.

# Fundamentos

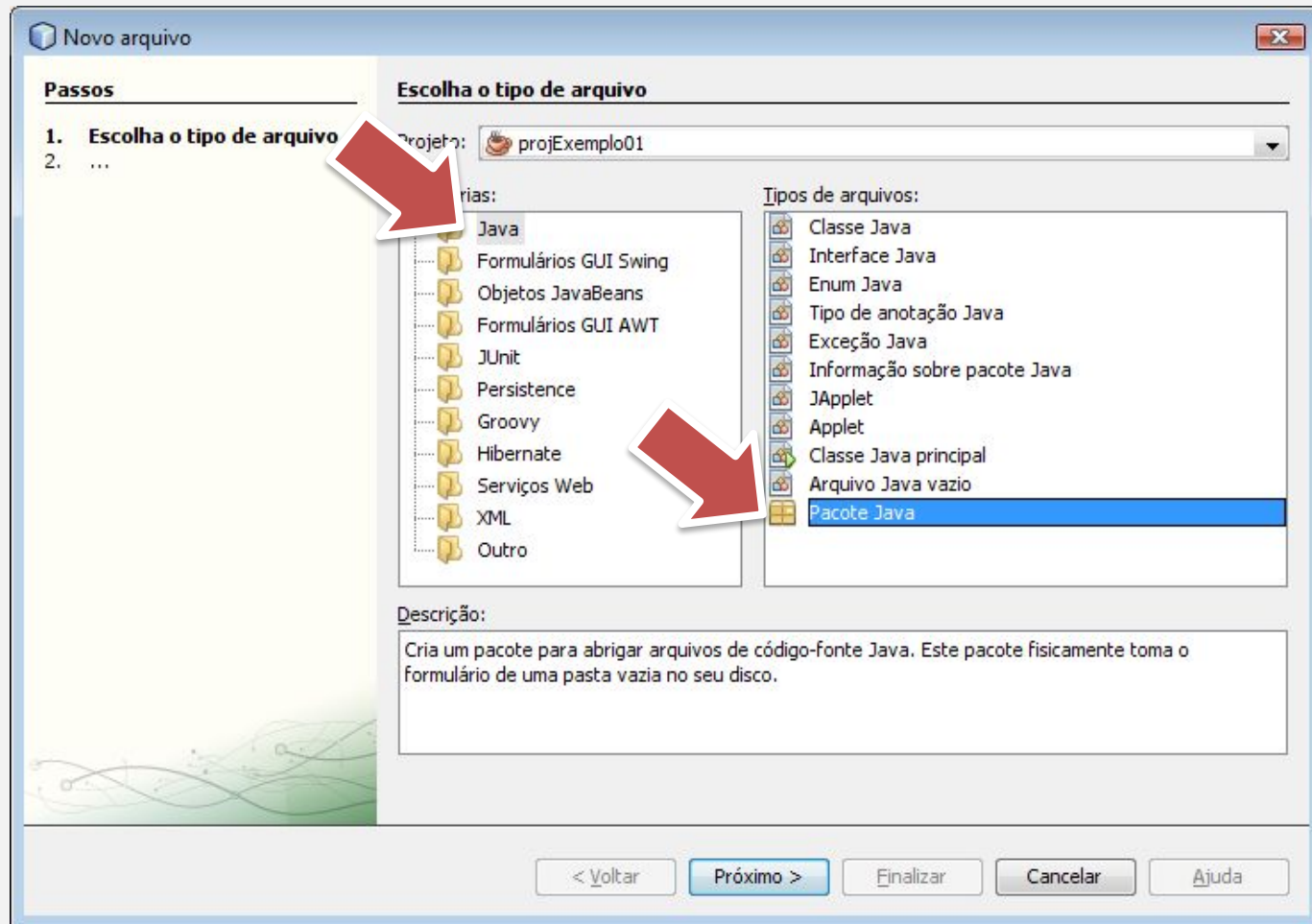
- Exemplo:
  - O pacote ***javax.swing*** possui classes para construção de interfaces gráficas.
  - A classe ***JOptionPane*** está dentro do pacote ***javax.swing***

# Fundamentos

- Nomenclatura
  - Pacotes Principais (***Core Packages***)
    - Começam sempre com java
    - Fazem parte do Java SDK
  - Pacotes de Extensões (***Extension Packages***)
    - Começam com javax
    - São desenvolvidos pela Sun ou por outros programadores.

# Fundamentos

## ■ Criando *pacotes*



# Fundamentos

## ■ Identificadores Válidos

- Os identificadores em Java consistem em letras, dígitos, sublinhados (“\_”) e sinais de cifrão (“\$”),
- **Não** podem iniciar com um dígito
- **Não** podem conter espaços em branco.
- Identificadores válidos:
  - Valor1, \$Valor1, \_Valor1
- Identificadores inválidos:
  - 1Valor, Valor 1

# Fundamentos

## ■ Comentários

- Múltiplas linhas: `/* ... */`
- Uma linha: `//`

```
/*  
    Este programa tem como objetivo principal a exibição na  
    tela da mensagem "Olá Mundo!!!"  
*/  
public class Main { // Início da Classe  
    // Método Principal  
    public static void main(String[] args) {  
        // Escrita da mensagem na tela  
        System.out.println("Olá Mundo!!! ");  
    }  
} // Final da Classe
```



# Fundamentos

- A declaração de variáveis em Java utiliza o seguinte formato

***<tipodedados> <nomedavariável>***

- Exemplo:

```
int x;  
float y;  
double z;
```

# Fundamentos

- As constantes em Java são declaradas através do modificador ***final*** e utilizam o seguinte formato

***final*** <tipodedados> <nomedaconstante>

- Exemplo

```
final int X = 10;  
final double VALOR = 1.58;
```

# Fundamentos

- Comandos de Atribuição
  - Utilizado para atribuir valores a variáveis, sendo representado pelo símbolo = (*igual*)
- Exemplo:

```
int x = 10;  
int a,b,c;  
  
a = b = c = 10;
```

# Fundamentos

## ■ Tipos de Dados Primitivos

Tipo	Quantidade de bits	Valores
char	16	'\u0000' a '\uFFFF'
byte	8	-128 a + 127
int	32	-2.147.483.648 a + 2.147.483.647
short	16	-32.768 a + 32.767
long	64	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807
float	32	-3.40292347E+38 a +3.40292347E+38
double	64	-1.79769313486231570E+308 a +1.79769313486231570E+308
boolean	8	true ou false

A linguagem Java utiliza como padrão para ponto flutuante o tipo de dados *double*.

# Fundamentos

- Como em qualquer linguagem de programação, Java também possui um conjunto de ***palavras reservadas***.
- Essas palavras não podem ser utilizadas como nomes de ***identificadores***, tais como:
  - Nome da classe
  - Nome de variável ou constante
  - Nome de métodos
  - Nome de objetos

# Fundamentos

## ■ Palavras reservadas

<code>abstract</code>	<code>default</code>	<code>goto</code> *	<code>Package</code>	<code>this</code>
<code>assert</code> ***	<code>do</code>	<code>if</code>	<code>Private</code>	<code>throw</code>
<code>boolean</code>	<code>double</code>	<code>implements</code>	<code>protected</code>	<code>throws</code>
<code>break</code>	<code>else</code>	<code>import</code>	<code>public</code>	<code>transient</code>
<code>byte</code>	<code>enum</code> ****	<code>instanceof</code>	<code>Return</code>	<code>try</code>
<code>case</code>	<code>extends</code>	<code>int</code>	<code>Short</code>	<code>true</code>
<code>catch</code>	<code>false</code>	<code>interface</code>	<code>static</code>	<code>void</code>
<code>char</code>	<code>final</code>	<code>long</code>	<code>strictfp</code> **	<code>volatile</code>
<code>class</code>	<code>finally</code>	<code>native</code>	<code>Super</code>	<code>while</code>
<code>const</code> *	<code>float</code>	<code>new</code>	<code>Switch</code>	
<code>continue</code>	<code>for</code>	<code>null</code>	<code>Synchronized</code>	
*	não usado			
**	adicionado à versão 1.2			
***	adicionado à versão 1.4			
****	adicionado à versão 1.5			

# Fundamentos

## ▪ Operadores Aritméticos

Função	Sinal	Exemplo
Adição	+	$x + y$
Subtração	-	$x - y$
Multiplicação	*	$x * y$
Divisão	/	$x / y$
Resto da divisão inteira	%	$x \% y$
Sinal negativo	-	$-x$
Sinal positivo	+	$+x$
Incremento unitário	++	$++x$ ou $x++$
Decremento unitário	--	$--x$ ou $x--$

# Fundamentos

- Operadores Relacionais

Função	Caractere(s) utilizado(s)	Exemplo
Igual	<code>==</code>	<code>x == y</code>
Diferente	<code>!=</code>	<code>x != y</code>
Maior que	<code>&gt;</code>	<code>x &gt; y</code>
Maior ou igual a	<code>&gt;=</code>	<code>x &gt;= y</code>
Menor que	<code>&lt;</code>	<code>x &lt; y</code>
Menor ou igual a	<code>&lt;=</code>	<code>x &lt;= y</code>



# Fundamentos

```
public static void main(String[] args) {  
  
    String saida = "";  
    int num1, num2;  
  
    num1 = Integer.parseInt(JOptionPane.showInputDialog("N1: "));  
    num2 = Integer.parseInt(JOptionPane.showInputDialog("N2: "));  
  
    if ( num1 == num2)  
        saida += num1 + " igual a " + num2 + "\n";  
  
    if ( num1 != num2)  
        saida += num1 + " diferente de " + num2 + "\n";  
  
    if ( num1 > num2)  
        saida += num1 + " maior que " + num2 + "\n";  
}
```

# Fundamentos

```
if ( num1 < num2)
    saida += num1 + " menor que " + num2+ "\n";

if ( num1 >= num2)
    saida += num1 + " maior ou igual a " + num2 + "\n";

if ( num1 <= num2)
    saida += num1 + " menor ou igual a " + num2 + "\n";

JOptionPane.showMessageDialog(null,saida, "Resultado",
                                JOptionPane.INFORMATION_MESSAGE);
}
```

# Fundamentos

- Operadores Lógicos

Função	Caractere(s) utilizado(s)	Exemplo
E lógico ou AND	&&	x && y
Ou lógico ou OR		x    y
Negação ou NOT	!	!x

# Fundamentos

## ■ Conversores de Tipo

Supondo a variável x	Converter em	y recebe o valor convertido
int x = 10	float	float y = <b>(float)</b> x
int x = 10	double	double y = <b>(double)</b> x
float x = 10.5	int	int y = <b>(int)</b> x
String x = "10"	int	int y = <b>Integer.parseInt(x)</b>
String x = "20.54"	float	float y = <b>Float.parseFloat(x)</b>
String x = "20.54"	double	double y = <b>Double.parseDouble(x)</b>
String x = "Java"	Vetor de bytes	byte b[] = <b>x.getBytes()</b>
int x = 10	String	String y = <b>String.valueOf(x)</b>
float x = 10.35	String	String y = <b>String.valueOf(x)</b>
double x = 254.34	String	String y = <b>String.valueOf(x)</b>
byte x[] – (x é um vetor de bytes)	String	String y = <b>new String(x)</b>

# Fundamentos

## ▪ *Saída de Dados*

- Comandos de Saída para console
  - `System.out.print`
  - `System.out.println`
  - `System.out.printf`

```
public class Main {  
    public static void main(String[] args) {  
        System.out.print ("Eu estou ");  
        System.out.print ("programando em Java !!!");  
    }  
}
```

# Fundamentos

- Códigos de Barra Invertida

- `\n` = nova linha
- `\r` = enter
- `\t` = tabulação (tab)
- `\b` = retrocesso
- `\"` = aspas
- `\\` = barra

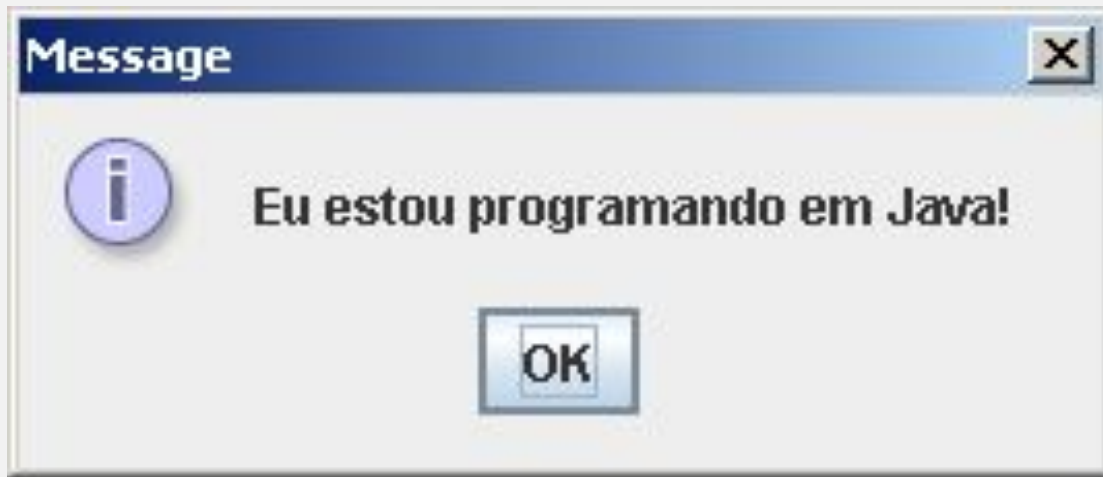
# Fundamentos

## ■ Exemplo

```
public class Main {  
  
    public static void main(String[] args) {  
  
        System.out.print("Este é um \n");  
        System.out.print("exemplo da utilização dos \n");  
        System.out.print("\n \"Códigos de Barra Invertida\" (\\) \n");  
  
    }  
}
```

# Fundamentos

- Os comandos de saída também podem ser utilizados para exibição de informações através de *caixas de diálogo*





# Fundamentos

```
/*
 * Para utilização do métodos JOptionPane é
 * necessária a inclusão do pacote abaixo.
 */
import javax.swing.JOptionPane;

public class Main {

    public static void main(String[] args) {

        JOptionPane.showMessageDialog(null, "Eu estou
                                         programando em Java!");

    }

}
```

# **ATIVIDADE PRÁTICA**

# Atividade Prática

## ■ **Exercício 1**

- Escreva um programa na linguagem Java que contenha duas variáveis do tipo *double* relativas a nota parcial e final de um aluno. Calcule e apresente na tela a média final utilizando método ***println***.

# Atividade Prática

## ■ **Exercício 2**

- Considere o seguinte trecho de código:

```
String nome = "João da Silva";  
double altura = 1.95;  
double peso = 92.58;
```

- Escreva um programa em linguagem Java que apresente na tela o IMC e o nome do João. Para isso, utilize a caixa de diálogo.

# Fundamentos

## ■ *Entrada de Dados*

- Utilizados para leitura de informações do teclado.
- Para o modo console a leitura pode ser feita através dos métodos do pacote *java.util.Scanner*

```
Scanner dados = new Scanner(System.in);  
dados.nextInt();  
dados.nextFloat();  
dados.nextDouble();  
dados.next();
```

# Fundamentos

```
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {

        Scanner entrada = new Scanner( System.in );
        int num1, num2, soma;

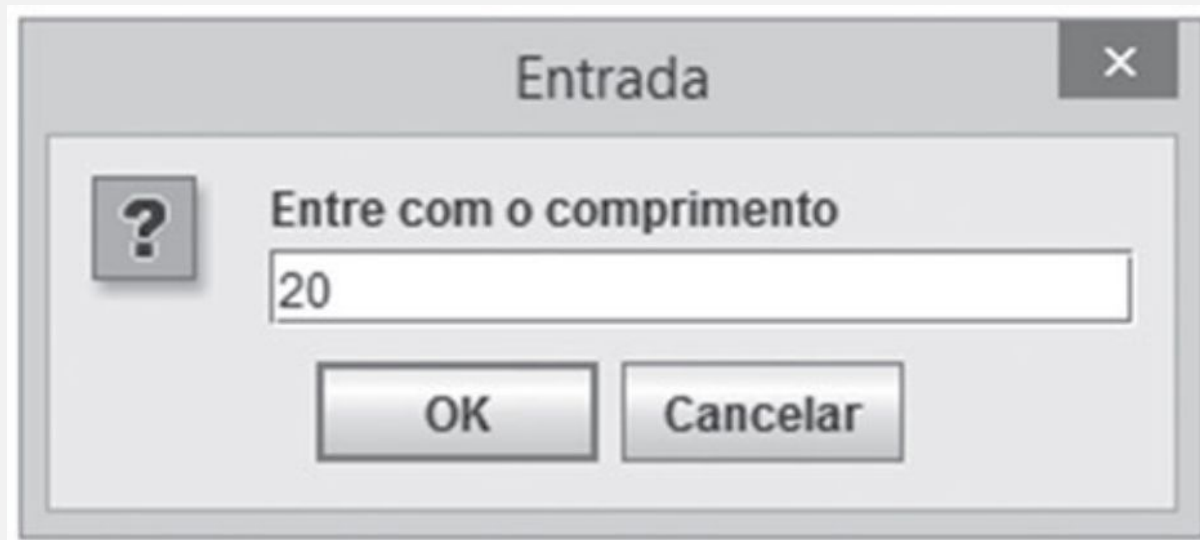
        System.out.print("Informe o primeiro número: ");
        num1 = entrada.nextInt();
        System.out.print("Informe o segundo número: ");
        num2 = entrada.nextInt();

        soma = num1+num2;
        System.out.print("Resultado = " + soma);

    }
}
```

# Fundamentos

- A entrada de informações também pode ser feita através de ***Caixas de Diálogos***.
- Essas caixas estão disponíveis também no pacote `javax.swing.JOptionPane`



# Fundamentos

```
String str_n1, str_n2;
int n1, n2, soma;

str_n1 = JOptionPane.showInputDialog("Primeiro número: ");
str_n2 = JOptionPane.showInputDialog("Segundo número: ");

// conversao String-->Integer
n1 = Integer.parseInt(str_n1);
n2 = Integer.parseInt(str_n2);

soma = n1+n2;

JOptionPane.showMessageDialog(null, "Soma de " + n1
    + " e " + n2 + " é igual a " + soma, "Resultado",
    JOptionPane.PLAIN_MESSAGE);
```



# Fundamentos

## ■ Ícones para Caixas de Diálogo

Tipo da caixa de dialogo	Ícone	Descrição
JOptionPane.ERROR_MESSAGE		Ícone para mensagens de erro
JOptionPane.INFORMATION_MESSAGE		Ícone para mensagens de informação
JOptionPane.WARNING_MESSAGE		Ícone para alertar o usuário sobre potenciais problema
JOptionPane.QUESTION_MESSAGE		Ícone para questionar o usuário. Normalmente utilizado como resposta, com os botões de Sim ou Não.
JOptionPane.PLAIN_MESSAGE	no icon	Não exibe nenhum ícone

# **ATIVIDADE PRÁTICA**

# Atividade Prática

## ■ **Exercício 1**

- Crie uma classe que receba o valor da base e da altura de um triângulo retângulo e apresente na tela sua área.

Elabores duas soluções:

- (1) utilizando *println* e *Scanner*
- (2) usando *MessageDialog* e *InputDialog*

# Atividade Prática

## ■ ***Exercício 2***

- Crie uma classe para calcular e exibir na tela o peso ideal.  $IMC = (peso / (altura^2))$

# Atividade Prática

## ■ ***Exercício 3***

- Crie uma classe que receba dois valores numéricos e apresente na tela o resultado das quatro operações básicas: soma, subtração, multiplicação e divisão.

# **ESTRUTURAS CONDICIONAIS E DE REPETIÇÃO**

# Estruturas Condicionais e de Repetição

- Estrutura Condicional Simples

```
if ( condicao )  
{  
  
}
```

- Estrutura Condicional Composta

```
if ( condicao )  
{  
  
}  
else  
{  
  
}
```

# Estruturas Condicionais e de Repetição

## ▪ Estrutura Condicional Aninhada

```
if ( condicao1 )
{
    if ( condicao2 )
    {
        ...
    }
}
else
{
    if ( condicao3 )
    {
        ...
    }
    else
    {
    }
}
```



# Estruturas Condicionais e de Repetição

- Estrutura Condicional Mutuamente Exclusiva

```
if ( condicao1 )
{
    ...
}
else if (condicao 2)
{
    ...
}
else if (condicao 3)
{
    ...
}
else
{
    ...
}
```

# Estruturas Condicionais e de Repetição

- Estrutura Condicional Mutuamente Exclusiva


```
switch (variavel)
{
    case 1:
        ...
        break;
    case 2:
        ...
        break;

    default:
        ...
}
```

# Estruturas Condicionais e de Repetição

- Estrutura Condicional Mutuamente Exclusiva

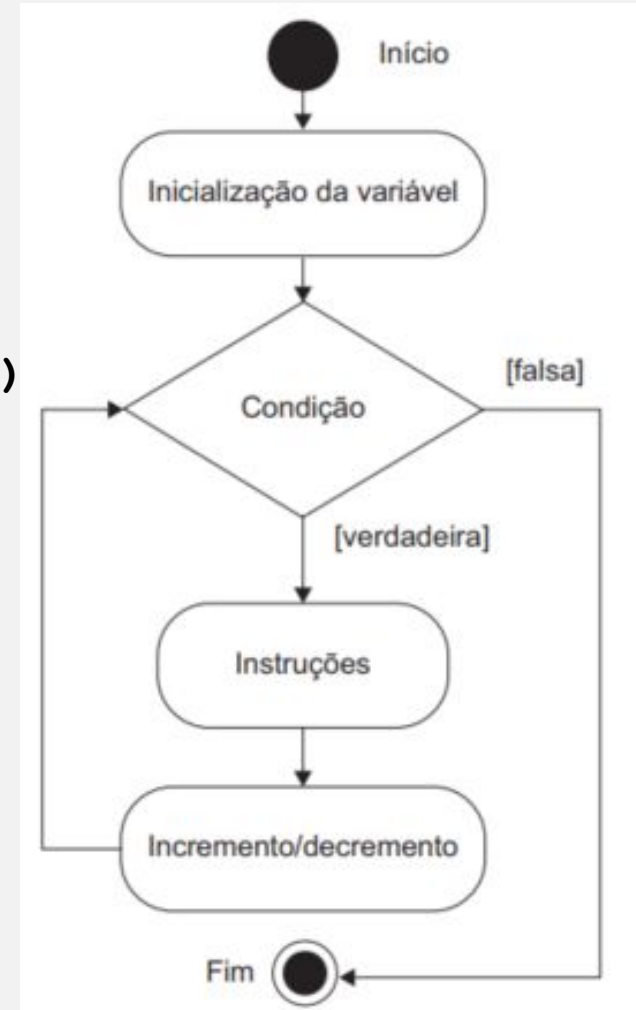
```
switch(variável do tipo String)  
{  
    case "valor1":  
        ...  
        break;  
    case "valor2":  
        ...  
        break;  
    default:  
        ...  
}
```



# Estruturas Condicionais e de Repetição

## ▪ Estrutura Repetição *for*

```
for ( condicao_inicial; condicao_parada;  
      incremento ou decremento )  
{  
    trecho do programa  
}
```



# Estruturas Condicionais e de Repetição

```
public static void main(String[] args) {  
  
    Scanner entrada = new Scanner( System.in );  
    int num;  
  
    System.out.println("Qual o número que você deseja  
                        calcular a Tabuada? ");  
    num = entrada.nextInt();  
  
    for (int i = 1; i <= 10; i++)  
        System.out.println(num + " x " + i + " = " + (num*i));  
}
```

# Estruturas Condicionais e de Repetição

```
public static void main(String[] args) {  
  
    System.out.println(" Números pares entre 1 e 10 ");  
    for (int i = 0; i <= 10; i = i+2)  
        System.out.println( i );  
}
```

# Estruturas Condicionais e de Repetição

```
public static void main(String[] args) {  
  
    System.out.println(" Números ímpares entre 1 e 10 ");  
    for (int i = 1; i <= 10; i = i+2)  
        System.out.println( i );  
}
```

# Estruturas Condicionais e de Repetição

- Exemplos de utilização da Estrutura **FOR**
  - Números entre 1 e 100 com incremento de 1
    - ***for ( int i = 1; i <= 100; i++ )***
  - Números entre 100 e 1 com decremento de 1
    - ***for ( int i = 100; i >= 1 ; i-- )***
  - Números entre 7 e 77 com incremento de 7
    - ***for ( int i = 7; i <= 77; i += 7 )***
  - Números entre 20 e 2 com decremento de 2
    - ***for ( int i = 20; i >= 2; i -= 2 )***



# Estruturas Condicionais e de Repetição

- Exemplos de utilização da Estrutura **FOR**
  - Geração da sequência: 2, 5, 8, 11, 14, 17, 20.
    - ***for ( int i = 2 ; i <= 20; i += 3 )***
  - Geração da sequência: 99, 88, 77, 66, 55, 44, 33, 22, 11, 0.
    - ***for ( int i = 99; i >= 0; i -= 11 )***

# Estruturas Condicionais e de Repetição

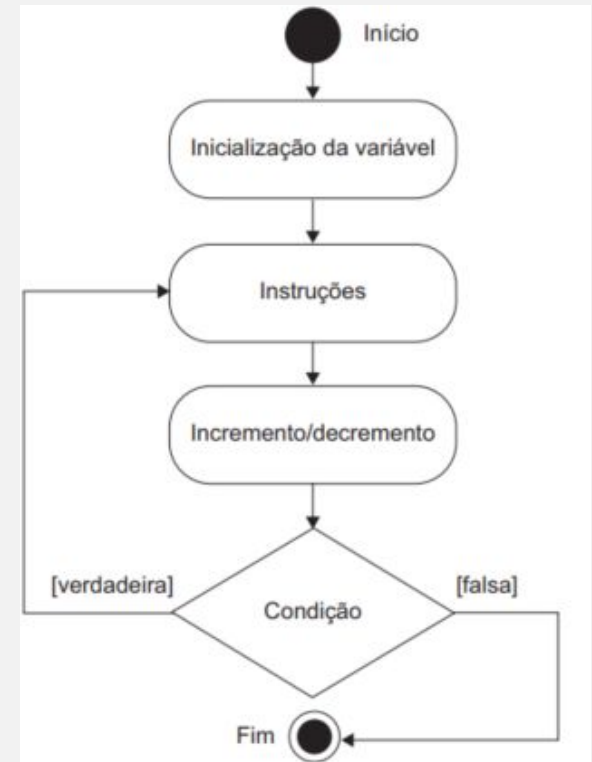
## ■ Somatório

```
public static void main(String[] args) {  
  
    int soma = 0;  
  
    System.out.println("Soma dos números entre 1 e 5 ");  
    for (int i = 1; i <= 5; i++)  
        soma = soma + i;  
  
    System.out.println("Resultado = " + soma );  
}
```

# Estruturas Condicionais e de Repetição

## ▪ Estrutura de Repetição *while*

```
public static void main(String[] args) {  
  
    int cont = 1;  
  
    while ( cont <= 10 )  
    {  
        System.out.println(cont);  
        cont++;  
    }  
}
```



# Estruturas Condicionais e de Repetição

```
public static void main(String[] args) {  
  
    Scanner entrada = new Scanner (System.in);  
  
    int soma = 0;  
  
    System.out.println("Entre com valores e pressione <enter>");  
    System.out.println("Para sair pressione <ctrl>+Z");  
  
    while ( entrada.hasNext() )  
    {  
        soma += entrada.nextInt();  
    }  
  
    System.out.println("Soma = " + soma);  
}
```

# Estruturas Condicionais e de Repetição

- Estrutura de Repetição ***do..while***

```
public static void main(String[] args) {  
  
    int cont = 1;  
  
    do{  
        System.out.println(cont);  
        cont++;  
    }while(cont <= 10);  
  
}
```

# **ATIVIDADE PRÁTICA**

# Atividade Prática

## ■ **Exercício 1**

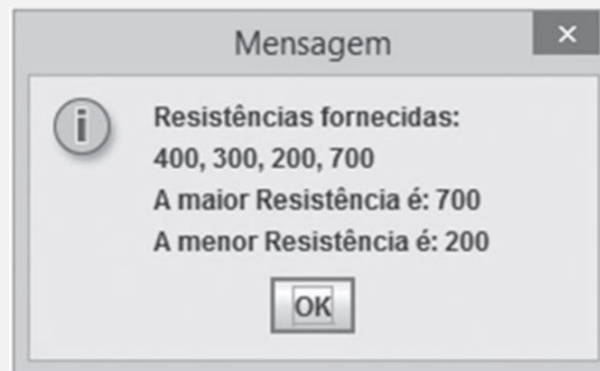
- Elabore uma classe que receba o nome de um produto e o seu valor. O desconto deve ser calculado de acordo com o valor fornecido conforme a Tabela. Apresente em tela o nome do produto, valor original do produto e o novo valor depois de ser realizado o desconto. Caso o valor digitado seja menor que zero, deve ser emitida uma mensagem de aviso.

Valor (R\$)	Desconto (%)
$\geq 50$ e $< 200$	5
$\geq 200$ e $< 500$	6
$\geq 500$ e $< 1000$	7
$\geq 1000$	8

# Atividade Prática

## ■ **Exercício 2**

- Na área da eletrônica, em um circuito em série temos que a resistência equivalente (total) desse circuito é obtida mediante a soma de todas as resistências existentes ( $RE = r1 + r2 + ... + rn$ ). Faça uma classe que receba o valor de quatro resistências ligadas em série, calcule e mostre a resistência equivalente, a maior e a menor resistência.

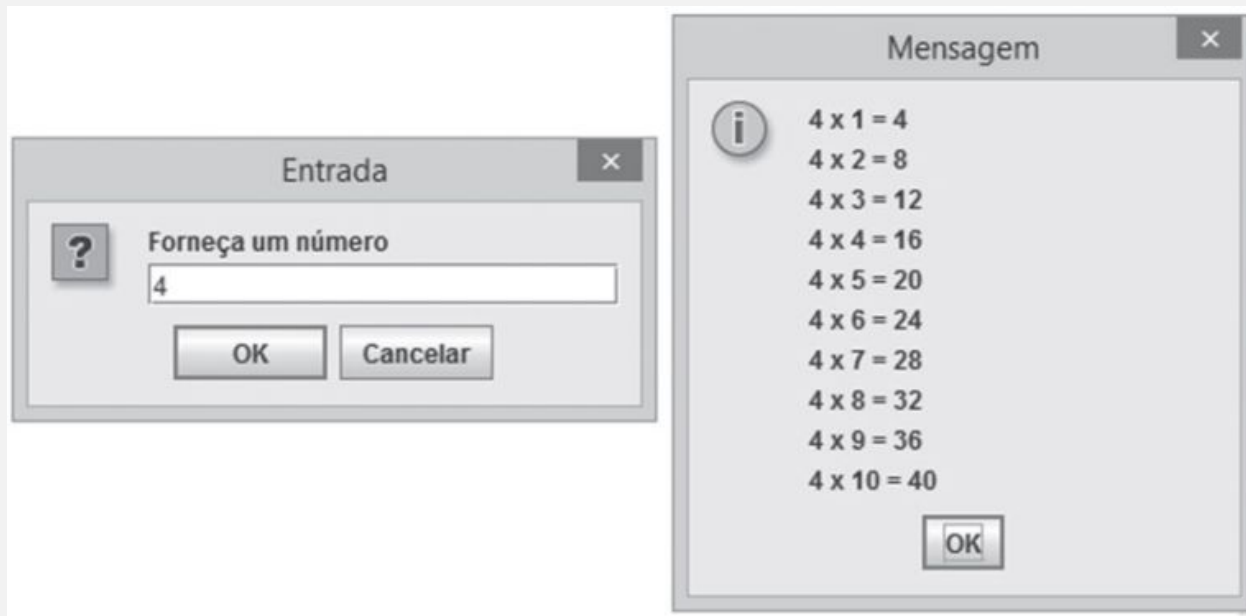




# Atividade Prática

## ■ *Exercício 3*

- Faça uma classe que apresente em tela a tabuada de qualquer número. O usuário fornece o número desejado e a classe apresenta a relação de todos os cálculos de 1 a 10.



# **OPERAÇÕES MATEMÁTICAS COM A CLASSE *MATH***

# Operações com a classe Math

- A linguagem Java possui uma classe chamada ***Math*** que contém diversos métodos especializados em realizar cálculos matemáticos. Observe a seguinte sintaxe:

```
Math.<nome do método>(<argumentos ou lista de argumentos>)
```

# Operações com a classe Math

- Arredondamento de números
  - ***Math.ceil***
    - Arredondamento para cima
    - Exemplo:  $1.85 \square 2$
  - ***Math.floor***
    - Arredondamento para baixo
    - Exemplo:  $1.85 \square 1$

# Operações com a classe Math

Método	Sintaxe	Descrição
round	Math.round(<valor>)	Recebe um valor numérico e retorna esse valor arredondado. Para valores decimais <0.5 arredonda para baixo, para valores >=0.5 arredonda para cima. Exemplos: Math.round(2.35) → 2, Math.round(2.59) → 3
max	Math.max(<valor1>, <valor2>)	Recebe dois valores numéricos e retorna o maior deles. Exemplo: Math.max(10,20) → 20
min	Math.min(<valor1>, <valor2>)	Recebe dois valores numéricos e retorna o menor deles. Exemplo: Math.max(10,20) → 10
sqrt	Math.sqrt(<valor>)	Recebe um valor numérico e retorna sua raiz quadrada. Exemplo: Math.max(25) → 25
pow	Math.pow(<valor1>, <valor2>)	Recebe dois valores numéricos (o operando e o expoente) e eleva o primeiro valor ao segundo. Exemplo: Math.max(10,2) → 100
abs	Math.abs(<valor>)	Recebe um valor numérico e retorna seu valor absoluto, desconsiderando o sinal. Exemplo: Math.max(-2) → 2

# Operações com a classe Math

- Método *random*

- Permite a geração de números aleatórios

```
String senha = "";
for (int i = 1; i <= 10; i++) {
    int num = (int) (Math.random() * 10);
    senha += num;
}
JOptionPane.showMessageDialog(null, "Senha gerada: " + senha);

for (int cartao = 1; cartao <= 4; cartao++) { // número de cartões
    String numerosCartao = "";
    for (int numCartao = 1; numCartao <= 6; numCartao++) { // qtde de números por cartão
        int num = (int) (Math.random() * 100);
        numerosCartao += num + "  ";
    }
    JOptionPane.showMessageDialog(null, "Números do cartão: " + cartao
        + "\n" + numerosCartao);
}
```

# Operações com a classe Math

- Formatação de números com ***DecimalFormat***
  - Cálculos matemáticos podem gerar resultados com muitas casas decimais.
  - Apresentar um resultado com muitas casas decimais não é muito agradável nem legível à maioria dos usuários.
  - Considere duas variáveis do tipo double:  $x=1$  e  $y=6$ . Ao realizar a divisão de  $x$  por  $y$ , aparece na tela o resultado 0.16666666666666666666, que não é o mais adequado para se apresentar na tela.

# *DecimalFormat*

```
DecimalFormat df = new DecimalFormat();
short idade = 38;
df.applyPattern("000");
System.out.println(df.format(idade));
int quantidade = 9750;
df.applyPattern("#0,000");
System.out.println(df.format(quantidade));
long estoque = 198564;
df.applyPattern("#,##0,000");
System.out.println(df.format(estoque));
float altura = 1.74f;
df.applyPattern("#0.00");
System.out.println(df.format(altura));
double peso = 70.25;
df.applyPattern("#0.00");
System.out.println(df.format(peso));
String valorEmReais = "2583.75";
df.applyPattern("R$ #,##0.00");
System.out.println(df.format(Double.parseDouble(valorEmReais)));
```

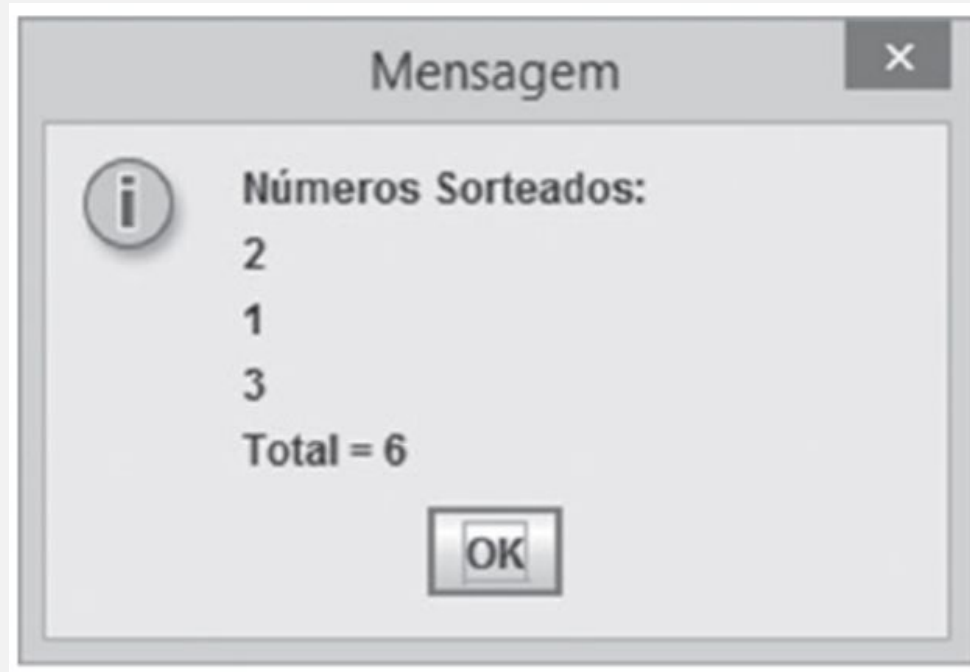


# **ATIVIDADE PRÁTICA**

# Atividade Prática

## ■ **Exercício 1**

- Crie uma classe que simule a jogada de um dado de seis lados (números de 1 a 6) por três vezes. Ao final some seus valores e apresente o resultado das três jogadas.



# Atividade Prática

## ■ **Exercício 2**

- Uma farmácia precisa ajustar os preços de seus produtos em 12%. Elabore uma classe que receba o valor do produto e aplique o percentual de acréscimo.
- O novo valor a ser calculado deve ser arredondado para mais ou para menos usando o método round.
- A classe deve também conter um laço de repetição que encerre o programa quando o usuário fornecer o valor zero(0) para o valor do produto

# Atividade Prática

## ■ **Exercício 3**

- Pesquise na referência da classe ***Math*** um método para calcular a raiz quadrada de um número informado pelo usuário.
- Apresente o resultado na tela com a formatação (*DecimalFormat*) de suas casas decimais.

# Atividade Prática

## ■ **Exercício 4**

- Pesquise na referência da classe ***Math*** um método para calcular um número elevado a outro.
- Por exemplo, 2 elevado a 3 é igual a 8.
- Apresente o resultado na tela com a formatação (*DecimalFormat*) de suas casas decimais.

# **ARRAYS**

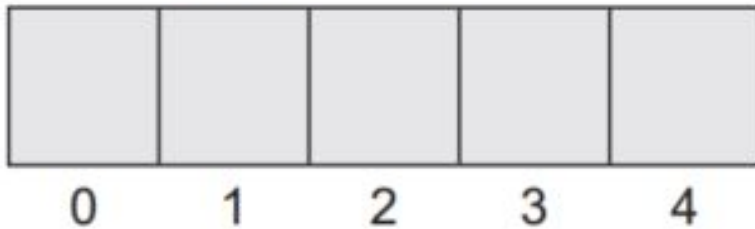
## **VETORES E MATRIZES**

# Arrays

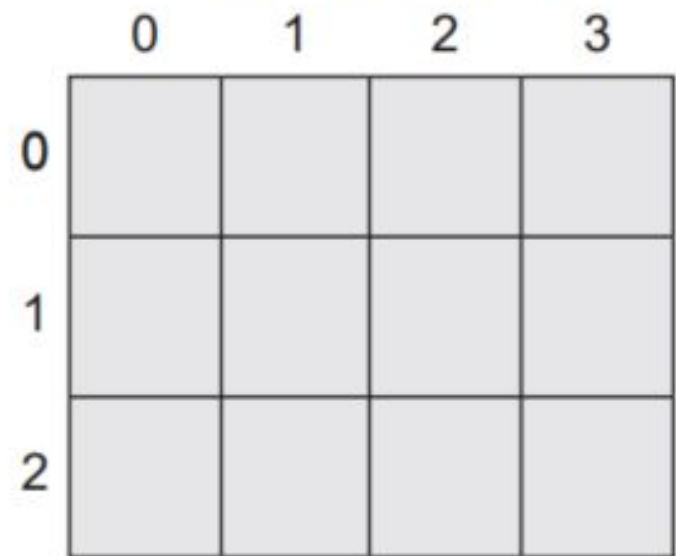
- Definição
  - Estruturas de dados que contém itens do mesmo tipo
  - Uma vez criadas mantém o mesmo tamanho
- Conjunto de dados localizado continuamente na memória
  - Cada localização da memória tem o mesmo nome
  - Cada localização da memória tem o mesmo tipo

# Arrays

Array unidimensional



Array bidimensional





# Arrays

## ■ Índices

- Em Java o índice dos Arrays começa em 0 (zero)
  - $v[0], v[1], \dots, v[n]$
- ***v.length***
  - retorna o tamanho (número de elementos) do array ***v***

# Arrays

## ■ Vetores

- `int v[] = new int[10]`
- `String c[] = new String[100];`

# Arrays

```
public static void main(String[] args) {  
  
    int vet[] = new int[5];  
  
    vet[0] = 10;  
    vet[1] = 20;  
    vet[2] = 30;  
    vet[3] = 40;  
    vet[4] = 50;  
  
    System.out.println("Tamanho do vetor = " + vet.length);  
    System.out.println("Elementos do vetor");  
    for (int i = 0; i < vet.length; i++)  
        System.out.println("  Posição " + i + " = " + vet[i]);  
  
}
```

# Arrays

```
public static void main(String[] args) {  
  
    int vet[] = {10,20,30,40,50};  
  
    System.out.println("Tamanho do vetor = " + vet.length);  
  
    System.out.println("Elementos do vetor");  
    for (int i = 0; i < vet.length; i++)  
        System.out.println("  Posição " + i + " = " + vet[i]);  
  
}
```

# Arrays

## ■ Matrizes

- Arranjos multidimensionais organizados através de linhas e colunas

- Exemplo: Matrizes 2D

  - `matriz[linha][coluna]`

- Declaração

  - `int mat[][] = new int[3][4];` □ 3 linhas 4 colunas

  - `int mat[][] = {{1,2},{3,4}};` □ 2 linhas 2 colunas

	0	1	2
0	120	232	201
1	129	111	187
2	128	247	123

# Arrays

```
int mat[][] = new int[3][3];

for (int lin = 0; lin < mat.length; lin++) {
    for (int col = 0; col < mat[lin].length; col++) {
        // Numeros aleatorios entre 0 e 99
        mat[lin][col] = (int) (Math.random() * 100);
    }
}

for (int lin = 0; lin < mat.length; lin++) {
    for (int col = 0; col < mat[lin].length; col++) {
        System.out.print( mat[lin][col] + "\t");
    }
    System.out.print("\n");
}
```

# **ATIVIDADE PRÁTICA**

# Atividade Prática

## ■ ***Exercício 1***

- Elabore uma classe que receba 5 notas de alunos por meio de `showInputDialog`, armazene essas notas em um array de cinco elementos, apresente em tela as cinco notas em ordem decrescente (da maior para a menor) e a média aritmética das notas.



# Atividade Prática

## ▪ **Exercício 2**

- Uma imagem é formada por pixels. Considere uma imagem com dimensão de 10 x 10 e faça uma classe que contenha um array bidimensional com essas dimensões.
- A seguir, para cada posição desse array bidimensional armazene um valor aleatório entre 0 e 255 (esses valores correspondem às tonalidades aplicadas sobre a imagem).
- Apresente em tela os valores gerados.

# **MÉTODOS PROCEDIMENTOS E FUNÇÕES**

# Métodos

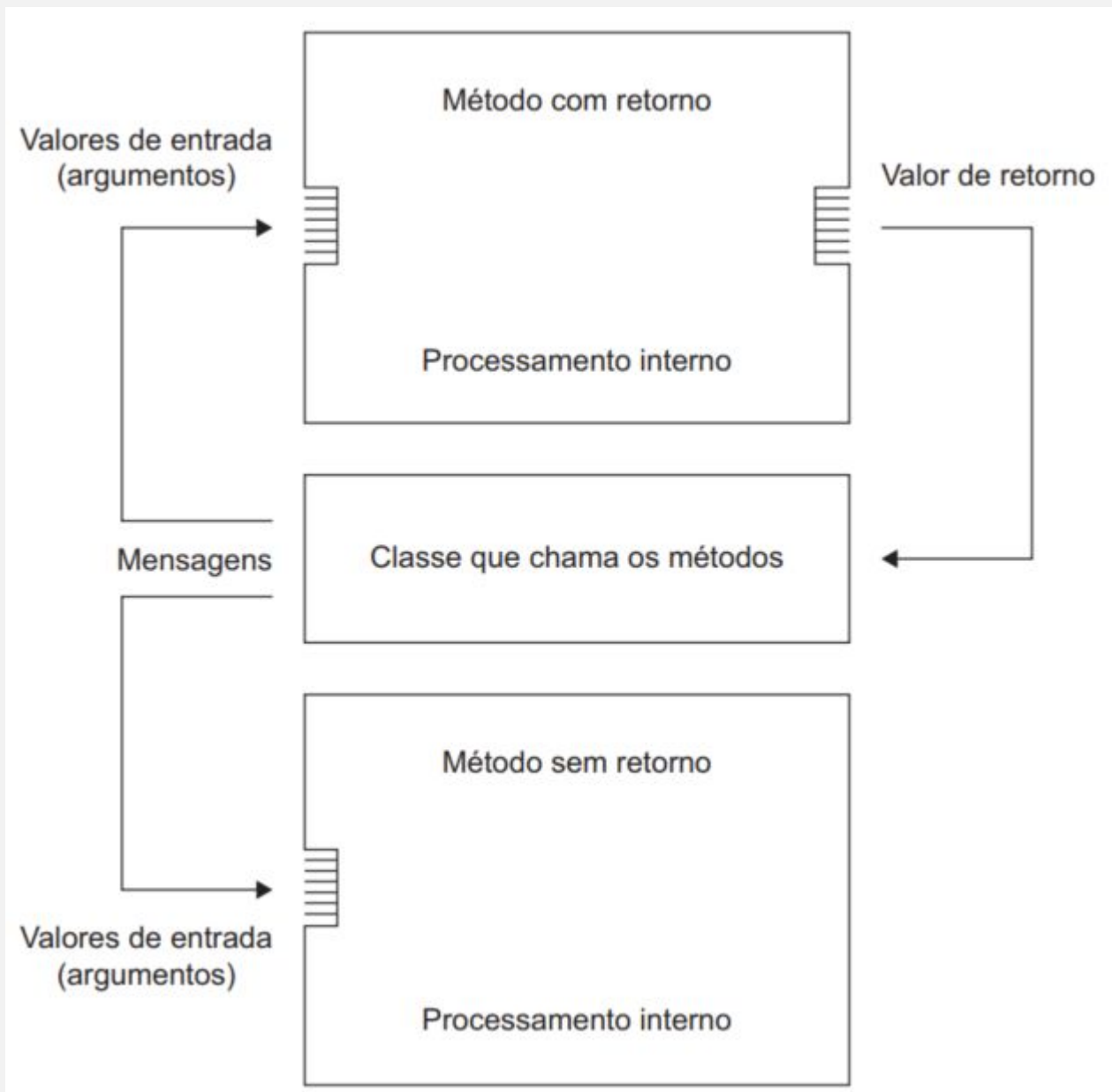
- Na linguagem Java a modularização de código fonte é realizada por meio de ***métodos***.
- Os métodos tem as mesmas características das ***funções*** da linguagem de programação C.

# Métodos

## ■ Sintaxe

```
qualificador tipo-do-retorno nome-do-método ([lista-de-parâmetros]) {  
    códigos do corpo  
}
```

- ***Qualificador***: define o nome e a visibilidade do método
- ***Tipo de retorno***: com retorno (int, double, etc) ou sem retorno *void*.
- ***Lista de parâmetros***: variáveis recebidas pelo método.



# *Exemplo*

## Métodos sem retorno

```
public class MetodosSemRetorno {  
    public static void main(String args[]) {  
        imprimir();  
        imprimirTexto("Ola");  
        mostrarQuadrado(10);  
        somar(10, 20);  
        mostrarMaior(10, 20, 30);  
        mostrarSexoPorExtenso('F');  
    }  
    public static void imprimir() {  
        System.out.println("Aprendendo a Linguagem Java");  
    }  
    public static void imprimirTexto(String texto) {  
        System.out.println(texto);  
    }  
    public static void somar(int a, int b) {  
        System.out.println(a + b);  
    }  
    public static void mostrarQuadrado(int valor) {  
        System.out.println(Math.pow(valor, 2));  
    }  
    public static void mostrarMaior(int a, int b, int c) {  
        System.out.println(Math.max(c, Math.max(a, b)));  
    }  
    public static void mostrarSexoPorExtenso(char sexo) {  
        if (sexo == 'F') {  
            System.out.println("Feminino");  
        } else if (sexo == 'M') {  
            System.out.println("Masculino");  
        } else {  
            System.out.println("Sexo desconhecido");  
        }  
    }  
}
```

# Exemplo

## Métodos com retorno

```
public class MetodosComRetorno {
    public static void main(String[] args) {
        System.out.println(MetodosComRetorno.somar(2, 3));
        System.out.println(MetodosComRetorno.mostrarPares(100));
        System.out.println(MetodosComRetorno.mostrarDiaPorExtenso(3));
        System.out.println(MetodosComRetorno.contarLetrasA("Banana"));
    }
    public static double somar(double a, double b) {
        return a + b;
    }
    public static String mostrarPares(int valor) {
        String retorno = "";
        for (int a = 0; a <= valor; a = a + 2) {
            retorno += a + " ";
        }
        return retorno;
    }
    public static String mostrarDiaPorExtenso(int dia) {
        String extenso = "Domingo";
        if (dia == 2) {
            extenso = "Segunda";
        } else if (dia == 3) {
            extenso = "Terça";
        } else if (dia == 4) {
            extenso = "Quarta";
        } else if (dia == 5) {
            extenso = "Quinta";
        } else if (dia == 6) {
            extenso = "Sexta";
        } else if (dia == 7) {
            extenso = "Sábado";
        } else {
            extenso = "dia não reconhecido";
        }
        return extenso;
    }
    public static int contarLetrasA(String palavra) {
        int quantidade = 0;
        palavra = palavra.toUpperCase();
        for (int a = 0; a < palavra.length(); a++) {
            if (palavra.charAt(a) == 'A') {
                quantidade++;
            }
        }
        return quantidade;
    }
}
```

# Métodos

- Método *sem retorno*

```
public static void exibir(String nome) {  
    System.out.println(nome);  
}
```

- Método *com retorno*

```
public static int multiplicar(int x, int y) {  
    return x * y;  
}
```



# **ATIVIDADE PRÁTICA**

# Atividade Prática

## ▪ ***Exercício 1***

- Faça uma classe que possua um método que recebe um número inteiro, referente a idade de uma pessoa, e retorne uma mensagem:
  - De 0 a 2 anos = bebê
  - De 3 a 11 anos = criança
  - De 12 a 19 anos = adolescente
  - De 20 a 30 anos = jovem
  - De 31 a 60 anos = adulto
  - Maior de 60 = idoso

# Atividade Prática

## ▪ **Exercício 2**

- Um sorteio de brindes será feito em um supermercado este mês. Faça uma classe que forneça dez nomes de brindes e sorteie um desses brindes por meio de um método chamado ***sorteio***.

**FIM**