



Curso de **Java8** para **Web**

Professor
Antonio Benedito Coimbra Sampaio Jr

abc  | Treinamentos

www.abctreinamentos.com.br

Primeira Disciplina

JAVA 8 - Fundamentos Teóricos e Orientação a Objetos

- **UNIDADE 1:** Introdução à Tecnologia Java
- **UNIDADE 2: Introdução à Sintaxe Java**
- **UNIDADE 3:** Programação Orientada a Objetos em Java (Parte I)
- **UNIDADE 4:** Programação Orientada a Objetos em Java (Parte II)

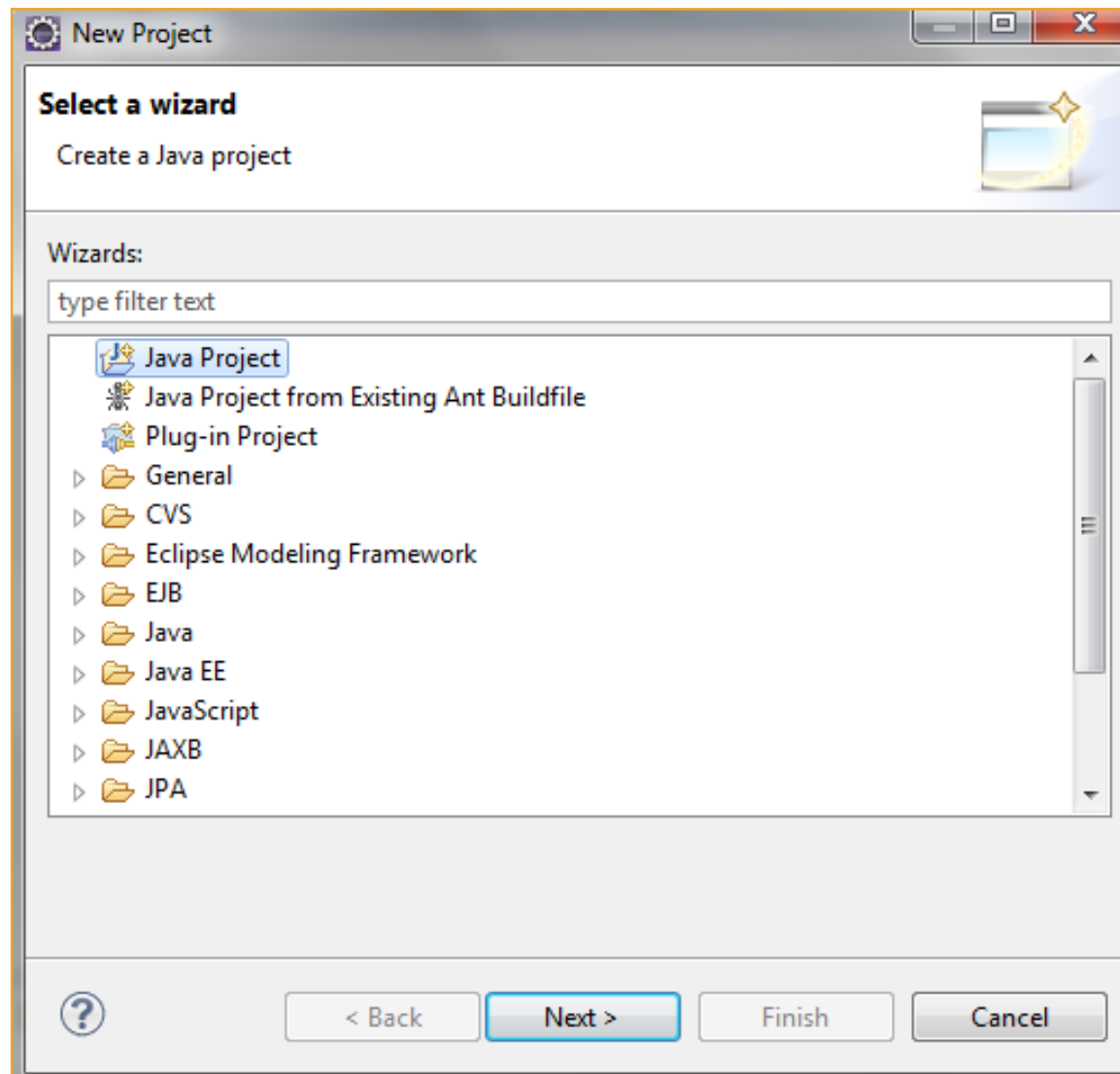
UNIDADE 2

INTRODUÇÃO À SINTAXE J³AVA

Análise do Primeiro Código Java

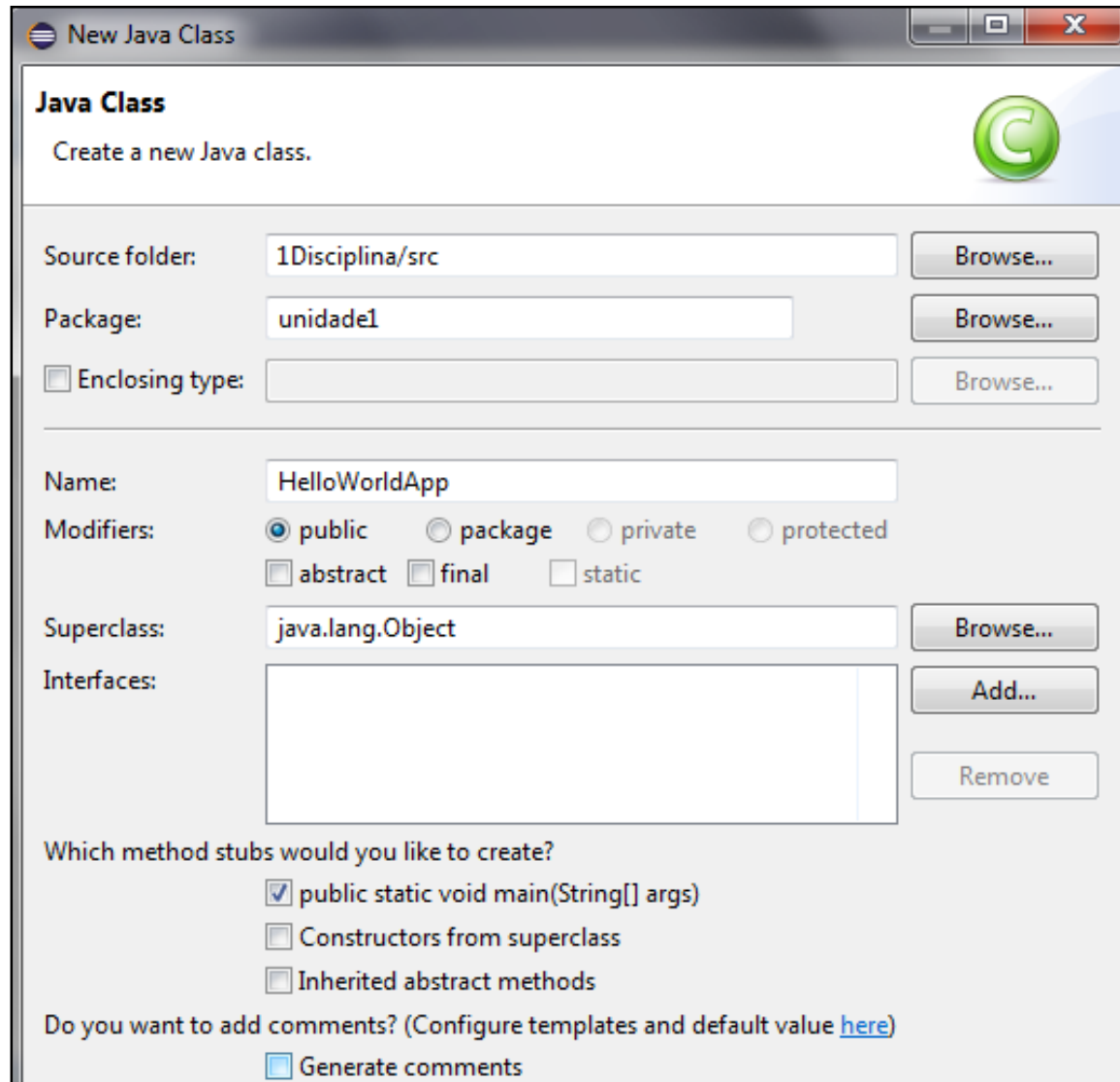
Primeiro Programa JAVA

- Inicialmente, é necessário criar um Projeto Java no Eclipse.



Primeiro Programa JAVA

- Depois, deve-se criar a Classe Java.

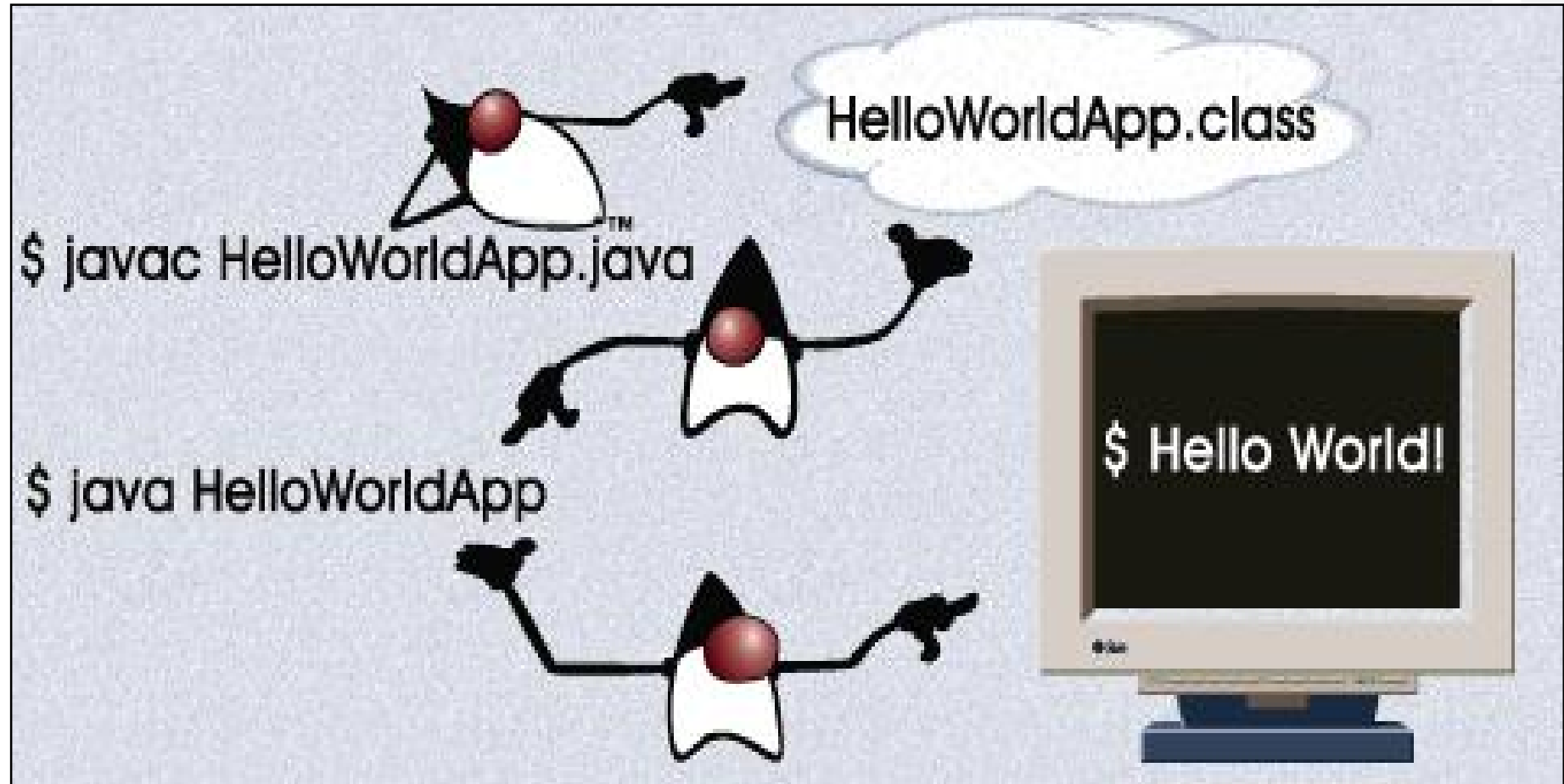


Primeiro Programa JAVA

- Digite o programa “HelloWorldApp.java” apresentado abaixo no editor do Eclipse.

```
/** Primeiro Programa Java */  
package unidade1;  
  
class HelloWorldApp  
{  
    public static void main(String arg[])  
    {  
        System.out.println("Hello World!");  
    }  
}
```

Primeiro Programa JAVA



Primeiro Programa JAVA

Comentário de bloco

```
/** Aplicação Hello World */
```

Nome da classe

```
public class HelloWorld {
```

Nome do método

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

Declaração de
argumento

variável local: args
tipo: String[]

```
System.out.println("Hello, world!");
```

Ponto-e-vírgula
é obrigatório no
final de toda
instrução

Definição de método main()

Atribuição de argumento
para o método println()

Definição de classe
HelloWorld

Chamada de método println()
via objeto out acessível
através da classe System

© Helder da Rocha

Primeiro Programa JAVA

COMENTÁRIOS

- Os comentários em Java seguem a mesma sintaxe da linguagem C;
- O compilador ignora essas linhas.

```
/* texto */  
// texto  
/** Primeiro Programa Java **/
```

CLASSE

- É a unidade básica para uma linguagem O.O como Java;
- **class** é a palavra reservada que marca o início da declaração de uma classe.

```
class Nome  
{  
...}
```

Primeiro Programa JAVA

BLOCOS

- um bloco está sempre entre chaves { }
- Em Java as instruções terminam em ponto-e-vírgula (;)

MÉTODO **MAIN**

- Toda aplicação Java SE deve possuir o método **main**.
- O método **main** indica o início de execução de qualquer programa Java.

```
public static void main (String args[]){  
    System.out.println("Hello World!");  
}
```

MÉTODO **PRINTLN**

- Escreve na tela o conteúdo nos parêntesis.
- Chamada do método println para o atributo out da classe System.
- O argumento (“...”) é do tipo String.

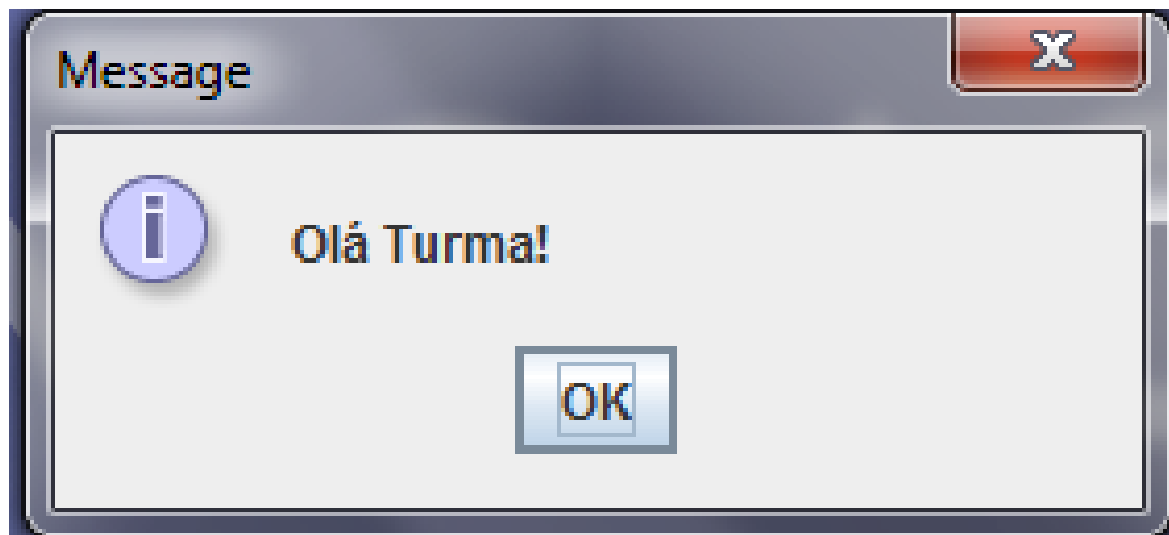
Detalhes Importantes

BLOCOS

- Um programa fonte Java deve sempre ter a extensão `‘.java’`;
- Um programa Java compilado deve sempre ter a extensão `‘.class’`;
- A linguagem é “case-sensitive”, letras maiúsculas são diferentes de letras minúsculas;
- Cada programa é uma classe;
- Nome da classe no programa tem que ser igual ao nome do arquivo físico `‘.java’`;
- Sintaxe similar a linguagem C/C++;
- As classes podem estar agrupadas em package;
- Package é um subdiretório. Exemplo: `java.awt`, `java.math`;
- As classes podem ser compactadas (zip ou jar);

Exercícios

- 1) Escreva um programa Java que imprima na tela várias mensagens de texto.
- 2) Adapte o mesmo programa para mostrar as mensagens como uma janela gráfica.



- Dica: Pesquise na Java SE DOC API a classe JOptionPane.

Estruturas de Programação

Estruturas de Programação

IDENTIFICADORES

- Nomeiam variáveis, funções, classes e objetos;
- Podem conter letras e/ou dígitos, “_” e “\$”;
- Não podem ser iniciados por dígito;
- Não podem ser palavras reservadas;
- Não tem tamanho máximo.

VARIÁVEIS

- Variáveis são usadas em linguagens em geral para armazenar valores
- Valores são passados para variáveis através de operações de atribuição
- Sintaxe Padrão no Java:

```
<TIPO_VARIAVEL> nomeVariavel;
```

Exemplo:

```
int anoNascimento;
```

Nomes das Variáveis

VÁLIDOS

Nome	NumDepen
total_geral	NOME

INVÁLIDOS

```
1prova  
total geral  
class // Palavra-chave
```


Palavras-Chaves

Palavras-chave Java

<code>abstract</code>	<code>do</code>	<code>implements</code>	<code>private</code>	<code>throw</code>
<code>boolean</code>	<code>double</code>	<code>import</code>	<code>protected</code>	<code>throws</code>
<code>break</code>	<code>else</code>	<code>instanceof</code>	<code>public</code>	<code>transient</code>
<code>byte</code>	<code>extends</code>	<code>int</code>	<code>return</code>	<code>true</code>
<code>case</code>	<code>false</code>	<code>interface</code>	<code>short</code>	<code>try</code>
<code>catch</code>	<code>final</code>	<code>long</code>	<code>static</code>	<code>void</code>
<code>char</code>	<code>finally</code>	<code>native</code>	<code>super</code>	<code>volatile</code>
<code>class</code>	<code>float</code>	<code>new</code>	<code>switch</code>	<code>while</code>
<code>continue</code>	<code>for</code>	<code>null</code>	<code>synchronized</code>	<code>const</code> *
<code>default</code>	<code>if</code>	<code>package</code>	<code>this</code>	<code>goto</code> *

* `goto` e `const` são palavras-chave, mas não são usadas em Java.

Tipos das Variáveis

	TIPO	MEMÓRIA	FAIXA
Lógico	boolean	1 bit	true ou false
Texto	char	2 bytes	\u0000 a \uFFFF
	String	variável	\u0000 a \uFFFF em cada localização
Integral	byte	1 byte	-128 a 127
	short	2 bytes	-32,768 a 32,767
	int	4 bytes	-2,147,483,648 a 2,147,483,647
	long	8 bytes	-9,223,372,036,854,775,808L a 9,223,372,036,854,775,807L
Ponto flutuante	float	4 bytes	aproximadamente +/-3.40282347E+38F (7 dígitos decimais significantes)
	double	8 bytes	aproximadamente +/-1.79769313486231570E+308 (15 dígitos decimais significantes)

Atribuição às Variáveis

A ATRIBUIÇÃO É REALIZADA COM O OPERADOR '='

- '=' serve apenas para atribuição – não pode ser usado em comparações (que usa '==')!
- Copia o valor da variável ou constante do lado direito para a variável do lado esquerdo.
- **EX: `y = 13;` // copia a constante inteira 13 para y**

INICIALIZAÇÃO PADRÃO JAVA

- variáveis numéricas com **0**;
- variáveis booleanas com **false**;
- outras variáveis com **null**.

Declarações das Variáveis

DECLARAÇÕES E ATRIBUIÇÕES DAS VARIÁVEIS

- As declarações podem ser exibidas em qualquer posição do código-fonte.

```
int x, y;  
float z = 3.144f;  
double w = 3.1415;  
boolean verdade = true;  
char c, d;  
c = 'A';  
d = '\u0013';  
x = 6;  
y = 1000;
```

Declarações das Variáveis

LÓGICO

```
boolean terminou;  
terminou = true;  
terminou = false;
```

TEXTO

- **char:** representa um caracter Unicode de 16 bits (exemplos: 'a', 'M', '\t', '\u02B1')
- **String:** representa uma seqüência de caracteres.

```
char opcao;  
opcao = 's';  
opcao = 'n';  
  
String frase;  
frase = "Ordem e Progresso";
```

Declarações das Variáveis

TIPO INTEGRAL (INTEIRO)

byte	8 bits	$2^7 \dots 2^7-1$
short	16 bits	$2^{15} \dots 2^{15}-1$
int	32 bits	$2^{31} \dots 2^{31}-1$
long	64 bits	$2^{63} \dots 2^{63}-1$

- Representações: 2 (decimal) / 077 (octal) / 0xBA (hexadecimal)

INTEIRO

```
byte index = 50;  
short soma = 2000;  
int num_carros = 5;  
long valor = 0XDADAL;
```

Declarações das Variáveis

TIPOS PONTO FLUTUANTE (REAL)

- float (32 bits)
- double (64 bits)
- Representações: 3.14 / 6.02E23 / 2.718F / 123.4E+306D

```
float pi = 3.1415f;  
double pi = 3.1415;
```

TIPO NUMÉRICO COM UNDERSCORE

```
public static void main(String... args) {  
  
    // THE OLD WAY  
    int oldBillion = 1000000000;  
  
    // THE NEW WAY  
    int newBillion = 1_000_000_000;  
  
}
```

NOVIDADE
JAVA 7

Exercícios

- 1) Escreva um programa que calcule o faturamento trimestral de uma empresa de software. Sabendo que, em Janeiro, as vendas foram de R\$15.000, em Fevereiro, R\$23.000 e em Março, R\$17.000. O valor final deverá ser impresso na tela.
 - 2) Adapte o programa acima para que a leitura das vendas mensais seja informada pelo usuário final .
- Dica: Pesquise na Java SE DOC API o método `showInputDialog(...)` da classe `JOptionPane`.

Operadores Matemáticos, Relacionais e Lógicos

Operadores

- Um operador produz um novo valor a partir de um ou mais argumentos
- Os operadores em Java são praticamente os mesmos encontrados em outras linguagens

`+, -, /, *, =, ==, <, >, >=, &&, etc.`

- A maior parte dos operadores só trabalha com valores de tipos primitivos.
- Exceções:

`+ e +=` são usados na concatenação de strings

`!=, = e ==` são usados também com objetos

Lista de Operadores JAVA

OPERADOR	FUNÇÃO	OPERADOR	FUNÇÃO
+	Adição	~	Complemento
-	Subtração	<<	Deslocamento à esquerda
*	Multiplicação	>>	Deslocamento à direita
/	Divisão	>>>	Desloc. a direita com zeros
%	Resto	=	Atribuição
++	Incremento	+=	Atribuição com adição
--	Decremento	-=	Atribuição com subtração
>	Maior que	*=	Atribuição com multiplicação
>=	Maior ou igual	/=	Atribuição com divisão
<	Menor que	%=	Atribuição com resto
<=	Menor ou igual	&=	Atribuição com AND
==	Igual	=	Atribuição com OR
!=	Não igual	^=	Atribuição com XOR
!	NÃO lógico	<<=	Atribuição com desl. esquerdo
&&	E lógico	>>=	Atribuição com desloc. direito
	OU lógico	>>>=	Atrib. C/ desloc. a dir. c/ zeros
&	AND	? :	Operador ternário
^	XOR	(tipo)	Conversão de tipos (cast)
	OR	instanceof	Comparação de tipos

Operadores Matemáticos

- (+) Soma
- (-) Subtração
- (*) Multiplicação
- (/) Divisão
- (%) Resto

ATENÇÃO (PROGRAMADORES C/C++):

- O operador + não é apenas aritmético (por exemplo, pode ser utilizado para inicialização e concatenação de strings);

```
String s = 1 + 2 + 3 + "=" + 4 + 5 + 6;
```

■ *Resultado: s contém a String "6=456"*

- Existe o tipo booleano, logo os operadores relacionais e lógicos NÃO geram inteiros.

Operadores Relacionais

- **(==) igual**
- **(!=) diferente**
- **(<) menor**
- **(<=) menor ou igual**
- **(>) maior**
- **(>=) maior ou igual**
- Sempre produzem um resultado booleano (*true* ou *false*).

Operadores Lógicos

- **(`&&`) E**
- **(`||`) OU**
- **(`!`) Negação**
- Sempre produzem um resultado booleano (*true* ou *false*).

Exercícios

- 1) Escreva um programa Java que leia o valor de dois números inteiros e calcule as operações aritméticas abaixo.
 - (+) Adição
 - (-) Subtração
 - (/) Divisão
 - (*) Multiplicação
- 2) Adapte o programa acima para fazer uso dos Operadores Relacionais e Lógicos listados abaixo:
 - (<) menor
 - (>) maior
 - (>=) maior ou igual
 - (&&) E (||) OU

Estruturas de Seleção

Controle de Execução

- O controle do fluxo da execução em Java utiliza os mesmos comandos existentes em outras linguagens;

Seleção: *if-else, switch-case*

Repetição: *for, while, do-while*

Desvios (somente em estruturas de repetição): *continue e break*

- Não existe o comando **goto**.

Seleção

IF-ELSE

```
if (expressão booleana)  
    instrução_simples;
```

```
if (expressão booleana) {  
    instruções  
}
```

```
if (expressão booleana) {  
    instruções  
} else if (expressão booleana) {  
    instruções  
} else {  
    instruções  
}
```

Seleção

IF-ELSE

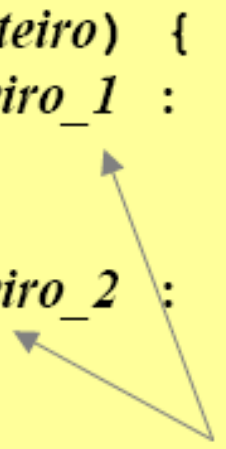
```
if (cont == 0)
{
    System.out.Println("Cont = 0");
}
else if (cont == 1)
{
    System.out.println("Cont = 1");
}
else
{
    System.out.println("Cont = Erro");
}
```

Seleção


SWITCH - CASE

```
switch(seletor_inteiro) {  
    case valor_inteiro_1 :  
        instruções;  
        break;  
    case valor_inteiro_2 :  
        instruções;  
        break;  
    ...  
    default:  
        instruções;  
}
```

uma constante inteira (inclui char)



```
switch(letra) {  
    case 'A' :  
        System.out.println("A") ;  
        break;  
    case 'B' :  
        System.out.println("B") ;  
        break;  
    ...  
    default:  
        System.out.println("?") ;  
}
```



Seleção

SWITCH – CASE

```
char cor = ' ';  
switch (cor){  
    case 0:  
        setBackground(Color.black);  
        break;  
    case 2:  
        setBackground(Color.red);  
        break;  
    default:  
        setBackground(Color.white);  
        break;  
}
```

BREAK E DEFAULT

- A instrução **break** (opcional) impede que o fluxo de execução continue pelas opções seguintes;
- A instrução **default** (opcional) é chamada quando nenhuma cláusula 'case' for executada.

Seleção

SWITCH – CASE COM STRING

NOVIDADE JAVA 7

```
String cor = "";  
switch (cor)  
{  
    case "azul":  
        setBackground(Color.black);  
        break;  
    case "vermelho":  
        setBackground(Color.red);  
        break;  
    default:  
        setBackground(Color.white);  
        break;  
}
```

Exercício

- 1) Escreva um programa que leia uma nota (0 a 100) e escreva o conceito associado.

[90,100] "Excelente"

[70,90["Bom"

[50, 70["Regular"

[0,50["Insuficiente"

Estruturas de Repetição

Repetição

WHILE & DO-WHILE

```
while (expressão booleana )  
{  
    instruções;  
}
```

```
do  
{  
    instruções;  
} while (expressão booleana ) ;
```

```
int cont = 0;  
while (cont < 100){  
    System.out.println("contando "+ cont);  
    cont++;  
}
```

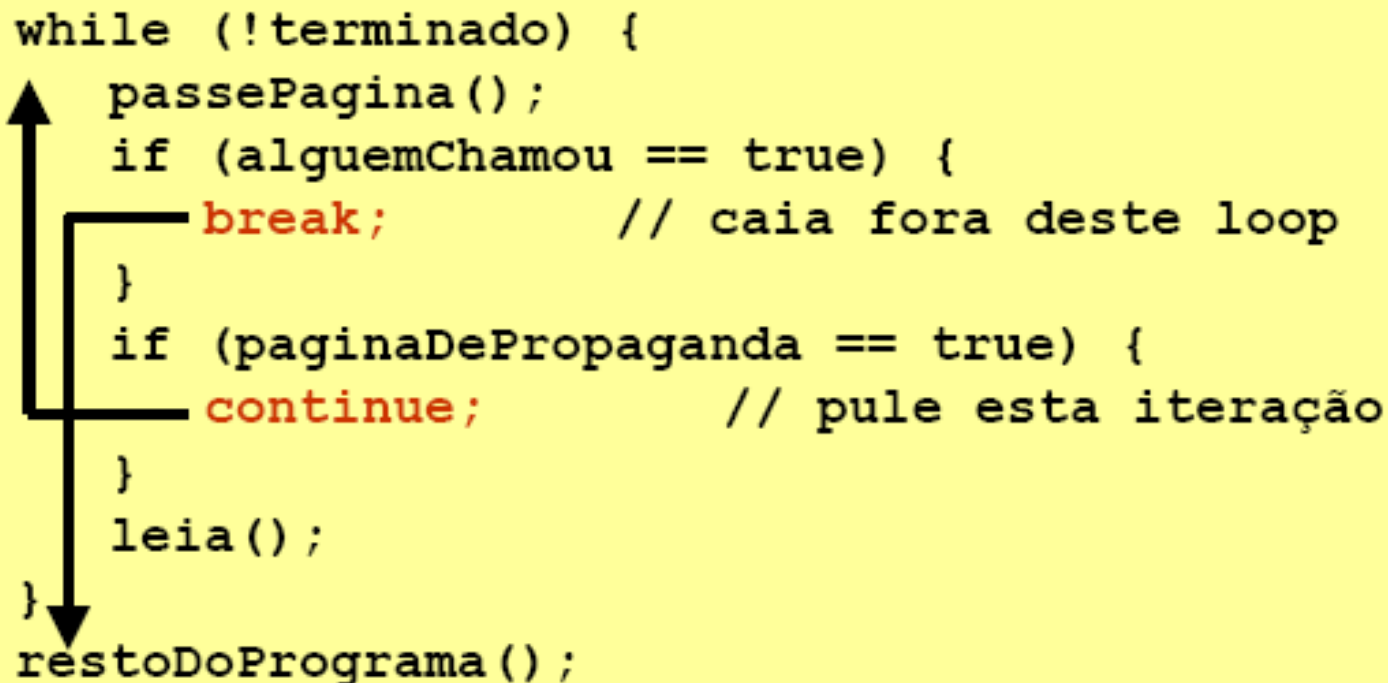
```
int cont = 0;  
do {  
    System.out.println("contando "+ cont);  
    cont++;  
} while (cont <100);
```

Repetição

CONTINUE

- A instrução **continue** (opcional) interrompe o fluxo de execução de um bloco de repetição.

```
while (!terminado) {  
    passePagina();  
    if (alguemChamou == true) {  
        break;           // caia fora deste loop  
    }  
    if (paginaDePropaganda == true) {  
        continue;       // pule esta iteração  
    }  
    leia();  
}  
restoDoPrograma();
```



Repetição

FOR

```
for ( inicialização ;  
      expressões booleanas;  
      passo da repetição )  
{  
    instruções;  
}
```

```
for ( inicialização ;  
      expressões booleanas;  
      passo da repetição )  
  
    instrução_simples;
```

```
for (int x=0; x<10; x++)  
{  
    System.out.println("Valor do X : " + x);  
}
```

Repetição

FOR-EACH

for (var: Collection framework)
{bloco de comandos}

```
int nums[]={1,2,3,4,5,6,7,8,9,10};  
int sum = 0;  
for (int x:nums)  
{  
    sum = sum + x;  
}
```

Exercícios

- 1) Escreva um programa para calcular a média aritmética, maior e menor valores de um conjunto de valores inteiros positivos.

- Observação: considere o **valor (-1)** como finalizador.

- 2) Escreva um programa para gerar a seguinte série abaixo para os 50 primeiros termos.

$$e^x = \frac{x^1}{1} + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \frac{x^5}{5} + \dots$$

- 3) Escreva um programa para gerar os 20 primeiro termos da série de Fibonacci , sendo que $F(n) = F(n-1) + F(n-2)$ e $F(1) = F(2) = 1$. Também calcule a sua média.

$$F(n) = 1, 1, 2, 3, 5, \dots$$

Vetores e Matrizes

Vetores (Arrays)

DECLARAÇÃO

- Podem ser declarados *arrays* de quaisquer dos tipos através dos símbolos “[” e “]”.
- A declaração não cria o *array*, isto é, não aloca memória. Isso é feito pela instrução **new** (*arrays* são objetos em Java).

```
char s[ ];          // declaração
s = new char[3];    // Criação
s[0] = 'A'; s[1] = 'B';
s[2] = 'C';         // atribuição
```

- Os colchetes podem ser usados antes ou depois da variável.
- Exemplo: **char s[]** ou **char []s**;

Vetores (Arrays)

DECLARANDO, CRIANDO E INICIALIZANDO

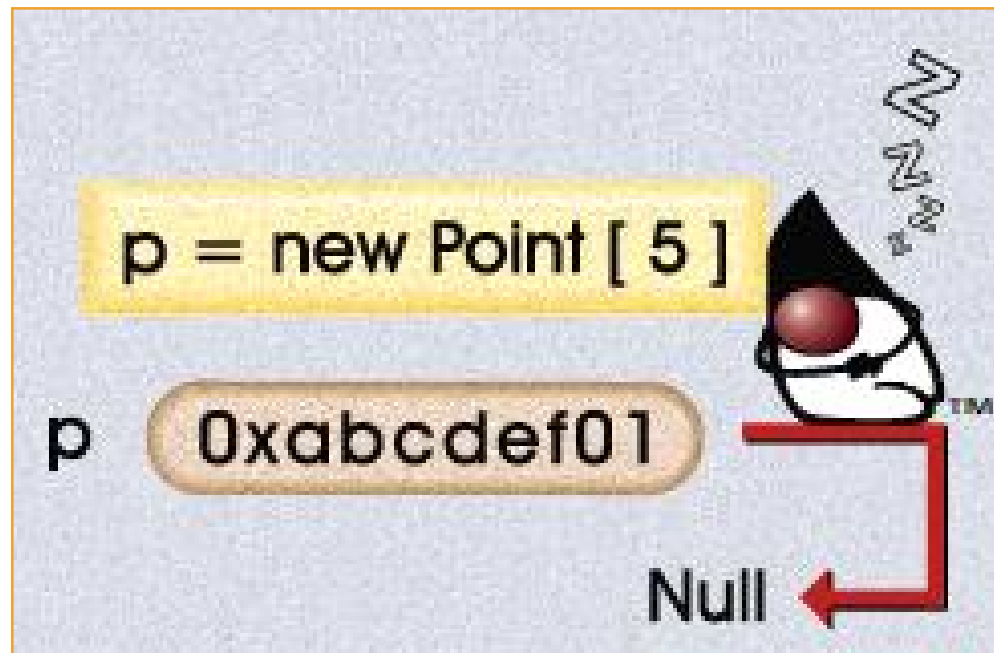
```
char s[ ] = {'A','B','C'};  
// declaração, criação e inicialização
```

- Em Java, *array* é um objeto, mesmo quando é composto por tipos primitivos. Apenas a declaração não cria o objeto, sendo necessário o uso da palavra reservada **new**.
- Quando um *array* é criado, todos os seus elementos são inicializados:
 - **null**, para objetos;
 - **0**, para int, long, short, byte, float, double;
 - **Unicode 0**, para char;
 - **false**, para boolean.

Vetores (Arrays)

DECLARANDO, CRIANDO E INICIALIZANDO

```
Point p[ ] = new Point[5];  
p[0] = new Point();  
p[1] = new Point();
```



Vetores (Arrays)

- Arrays não podem ser dimensionados na definição:

```
int vector[5]; //ERRADO!
```

- Arrays não podem ser utilizados sem a criação:

```
int vector[];  
vector[0] = 4; //ERRADO!
```

LENGTH

- Todo vetor em Java possui a propriedade **length** que informa o número de elementos que possui.
 - length** é uma propriedade *read-only* e já foi extremamente útil em blocos de repetição antes do JAVA 5.

Vetores (Arrays)

LENGTH

//ANTES DO JAVA 5

```
int lista [] = new int [10];  
for (int j = 0; j < lista.length; j++)  
{  
    System.out.println(lista[j]);  
}
```

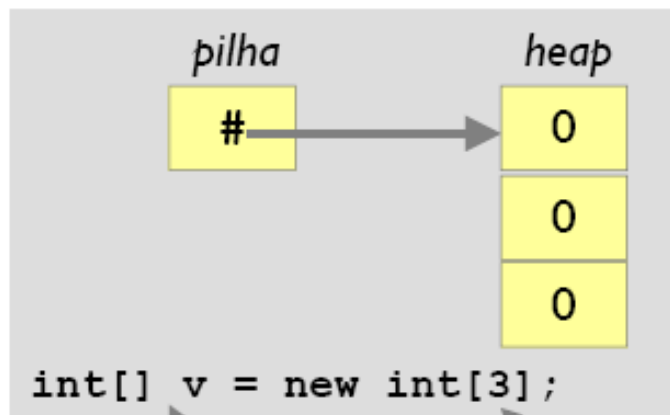
//APÓS O JAVA 5

```
int lista [] = new int [10];  
for (int j : lista)  
    System.out.println(j);
```

Vetores (Arrays)

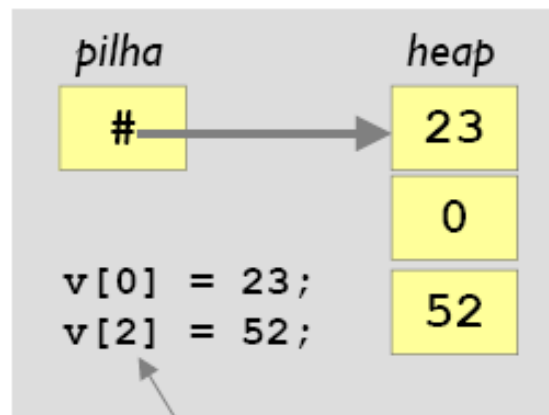
TIPOS PRIMITIVOS X OBJETOS

- De tipos primitivos



v é objeto do tipo (int[])

cria um vetor



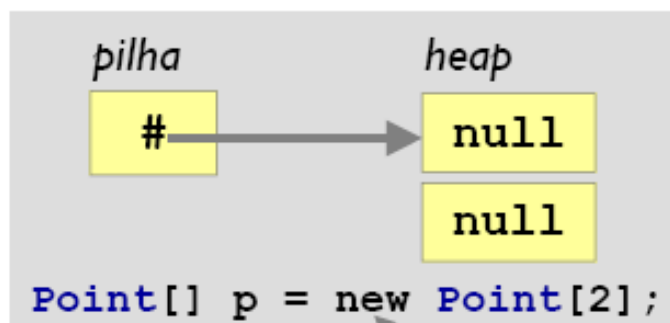
inicialização dos elementos

```
class Point {
    public int x;
    public int y;
}
```

Point

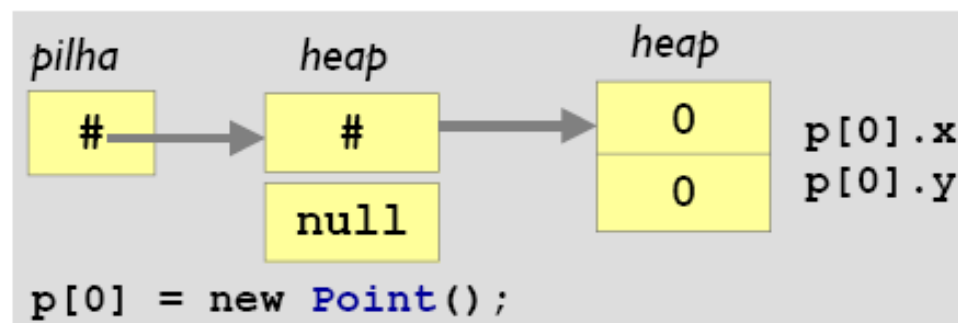
+**x**: int
+**y**: int

- De objetos (***Point** é uma classe, com membros `x` e `y`, inteiros*)



p é objeto do tipo (Point[])

cria um vetor



cria um objeto Point

© Helder da Rocha

Matrizes

VETORES MULTIDIMENSIONAIS

```
int matriz [][] = new int [4][4];  
matriz [0][0] = 300;  
matriz [1][3] = 600;
```

300			
			600

```
//MATRIZ NULA  
for(x=0; x <4; x++)  
    for (y=0; y <4; y++)  
        matriz[x][y] = 0;
```

Exercícios

- 1) Escreva um programa que leia um vetor A de 50 elementos, construa e imprima outro vetor B da seguinte forma:
 - Os elementos de ordem par são correspondentes a $(2*A)$;
 - Os elementos de ordem ímpar são correspondentes a $(A/2)$.
- 2) Construa uma matriz Identidade $I=3$ (apenas os valores diagonais são = 1). Todos os outros são 0.

$$I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

RESUMO

TÓPICOS APRESENTADOS

- Nesta aula nós estudamos:
 - **Análise do Primeiro Código Java**
 - **Estruturas de Programação**
 - **Operadores Matemáticos, Relacionais e Lógicos**
 - **Estruturas de Seleção**
 - **Estruturas de Repetição**
 - **Vetores e Matrizes**

ATIVIDADES PARA SE APROFUNDAR

- 1) Escrever uma classe Java para calcular o fatorial de um dado número N.
- 2) Escreva uma classe Java para criar uma matriz 4x4 de números inteiros. Em seguida, inicialize esta matriz e informe todos os elementos presentes e quantas vezes o mesmo aparece na matriz.
- 3) Escreva uma classe Java para calcular a soma de uma série harmônica (n).
$$\text{Harmonic}(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$
- 4) Dados dois números inteiros A e B, escreva uma classe Java para calcular a soma de todos os inteiros existentes entre A e B.

ATIVIDADES PARA SE APROFUNDAR

- **5) Estudar o conceito de Expressões Regulares.**
- **6) Criar uma classe Java que valide um endereço de e-mail utilizando expressões regulares.**