

# Programação de Computadores 1

## Programação Orientada a Objetos em Java

Laércio Silva

[Indsilva@hotmail.com](mailto:Indsilva@hotmail.com)

# A linguagem Java

## O início:

A Sun Microsystems, em 1991, deu início ao Green Project chefiado por James Gosling. Projeto que apostava na convergência dos computadores com outros equipamentos e eletrodomésticos

Foi lançado o \*7 (StarSeven), um controle remoto com uma interface gráfica *touchscreen* com ***aplicativos desenvolvidos na linguagem Oak.***

# A linguagem Java



**\*7 - StarSeven**



**Duke**



# A linguagem Java

- \* Em 1995, graças ao estouro da internet, a linguagem Oak foi adaptada para o desenvolvimento de aplicações para web (conhecidos hoje como applets) e foi rebatizada como Java.
- \* Hoje, mais de 5 milhões de desenvolvedores usam Java diariamente e cerca de 3 bilhões de dispositivos usam Java embutido.
- \* Em 2009 a Oracle compra a Sun.



# Principais características da linguagem

## **Portabilidade**

Uma mesma aplicação pode ser executada em diferentes plataformas (hardware e software) sem a necessidade de adaptação de código.

## **Multithreading**

Possibilidade de execução de diferentes processos simultaneamente.

## **Suporte a comunicação**

Oferece um conjunto de classes para desenvolvimentos de aplicações rodando em rede.

# Principais características da linguagem

- \* **Orientação a objetos:**

- \* Técnica de programação que modela componentes de softwares em termos de objetos do mundo real.

- \* **Vantagens:**

- \* Modularidade;
  - \* Reusabilidade;
  - \* Produtividade;
  - \* Facilidade de manutenção e expansão.

# Ambientes de desenvolvimento Java

- \* **JSE (Java Standard Edition)**

- \* Seu uso é voltado a PCs e servidores.
- \* Contem todo o ambiente necessário para a criação e execução de aplicações desktop e web de pequeno e médio porte.
- \* Pode-se dizer que essa é a plataforma principal, já que, o JEE e o JME tem sua base aqui.

# Ambientes de desenvolvimento Java

- \* **JEE (Java Enterprise Edition)**

- \* Voltada para o desenvolvimento de softwares corporativos.
- \* Baseados em componentes que são executados em um servidor de aplicação.

- \* **JME (Java Micro Edition)**

- \* Ambiente de desenvolvimento para dispositivos móveis ou portáteis, como telefones celulares e palmtops.



# Ambientes de desenvolvimento Java

- **JRE (Java Runtime Environment)**

É composto pela JVM e pela biblioteca de classes Java utilizadas para execução de aplicações Java, estas bibliotecas são chamadas de APIs Java.

Portanto para rodarmos uma aplicação Java é necessário instalarmos uma JRE no computador onde o software foi instalado.

# Ambientes de desenvolvimento Java

- \* **API - *Application Programming Interface*** (ou **Interface de Programação de Aplicativos**) é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.
- \* **API** é composta por uma série de funções acessíveis somente por programação, e que permitem utilizar características do software menos evidentes ao utilizador tradicional.

## Pacotes e Instalação do Java


- \* **JDK (Java Development Kit) ou SDK (Software Development Kit)**
  - \* Significa Kit de Desenvolvimento Java
  - \* Conjunto de ferramentas para a compilação, documentação e debug de aplicativos Java.
  - \* Composto pela JRE somada as ferramentas de desenvolvimento.

[http://www.java.com/pt\\_BR/download/](http://www.java.com/pt_BR/download/)


- Para baixar a IDE Netbeans do Java:

<http://netbeans.org/downloads/index.html>

# Instalação da Ferramenta NetBeans IDE

 **NetBeans**

NetBeans IDE | NetBeans Platform | Plugins | Docs & Support | Community | Partners

Search 

HOME / Download

## Download o NetBeans IDE 8.2

8.1 | 8.2 | Desenvolvimento | Arquivo

Endereço de email (opcional):

Inscrever-se na newsletter: ☒ Mensal ☐ Semanal

☒ Permito me contatar neste email

Idioma do IDE: 













Português (Bras ▾)

Plataforma: 

Windows ▾

Nota: Tecnologias em cinza não são suportadas para esta plataforma.

### Distribuições para baixar do NetBeans IDE

Tecnologias suportadas *	Java SE	Java EE	HTML5/JavaScript	PHP	C/C++	Tudo
 SDK da plataforma NetBeans	•	•				•
 Java SE	•	•				•
 Java FX	•	•				•
 Java EE		•				•
 Java ME						•
 HTML5/JavaScript		•	•	•		•
 PHP			•	•		•
 C/C++					•	•
 Groovy						•
 Java Card(tm) 3 Connected						•
Servidores embutidos						
 GlassFish Server Open Source Edition 4.1.1		•				•
 Apache Tomcat 8.0.27		•				•

Download

Download

Download x86

Download x86

Download x86

Download x64

Download x64

Download x64

Download

# Visão Geral da Plataforma Java

## Java™ SE Platform at a Glance

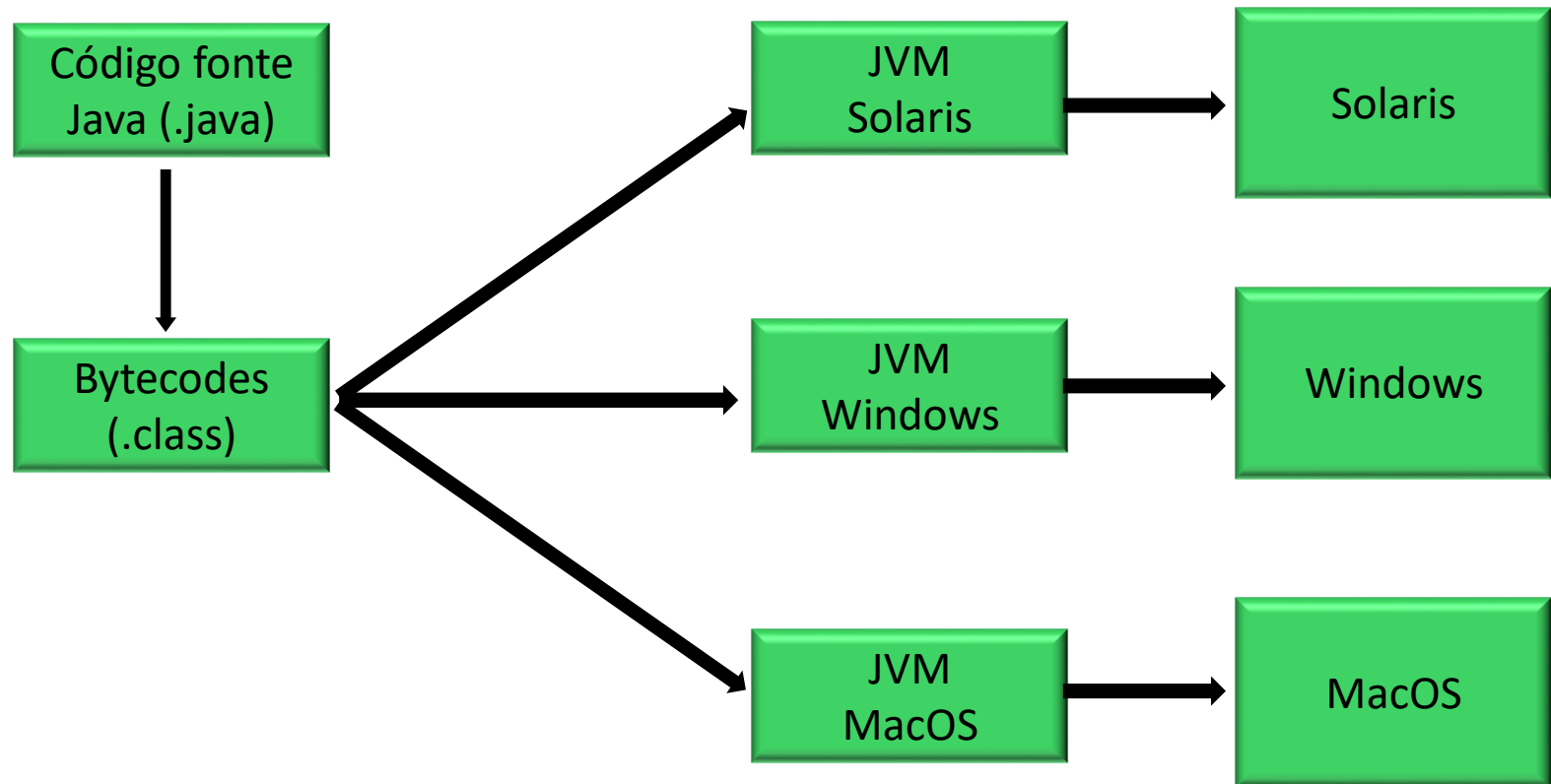
JDK	Java Language		Java Language								Java SE API	
	Tools & Tool APIs	java	javac	javadoc	apt	jar	javap	JPDA	JConsole	Java VisualVM		
		Security	Int'l	RMI	IDL	Deploy	Monitoring	Troubleshoot	Scripting	JVM TI		
	Deployment Technologies	Deployment			Java Web Start			Java Plug-in				
		AWT				Swing			Java 2D			
	User Interface Toolkits	Accessibility		Drag n Drop		Input Methods		Image I/O	Print Service	Sound		
		IDL	JDBC		JNDI		RMI	RMI-IIOP				
	Integration Libraries	Beans	Intl Support		Input/Output		JMX	JNI		Math		
		Networking	Override Mechanism		Security		Serialization	Extension Mechanism		XML JAXP		
	Other Base Libraries	lang and util	Collections		Concurrency Utilities		JAR		Logging	Management		
		Preferences API	Ref Objects		Reflection		Regular Expressions	Versioning	Zip	Instrumentation		
	Java Virtual Machine	Java Hotspot Client VM					Java Hotspot Server VM					
		Solaris			Linux		Windows			Other		

Java SE API

# Java como Plataforma de Programação

“Write Once, Run Anywhere”

(Escreva uma vez, execute em qualquer lugar)



# Java como Plataforma de Programação

A criação e utilização de um programa Java obedecem a cinco fases distintas.

- **Edição:** Armazenamento;
- **Compilação:** Armazenamento;
- **Carga:** Memória volátil;
- **Verificação dos Bytecodes:** Confirma a validade dos bytecodes e da segurança de Java;
- **Execução:** o interpretador lê o arquivo de bytecodes e transforma numa linguagem de 1 e 0 que o computador entende.

## Anatomia de uma Classe Java

```
public class Exemplo{
```

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

```
        System.out.println("Exemplo de Classe em
```

```
Java");
```

```
    }
```

```
}
```



## Anatomia de uma Classe Java

```
public class Exemplo{ public static void main ( String[] arg){  
System.out.println(“Exemplo de Classe em Java”)}}}
```



Não tem nenhum erro...

# Anatomia de uma Classe Java

```
public class Exemplo{ public static void main ( String[] arg){  
System.out.println("Exemplo de Classe em Java");}}
```

```
public class Exemplo2_6
```

```
{
```

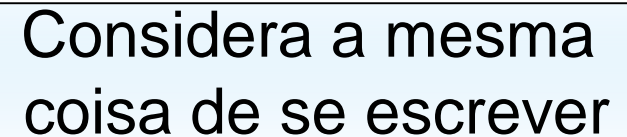
```
    public static void main ( String[] arg)
```

```
    {
```

```
        System.out.println("Exemplo de Classe em Java);  
        System.exit(0);
```

```
    }
```

```
}
```



Considera a mesma  
coisa de se escrever

# Anatomia de uma Classe Java

O compilador é *CASE-SENSITIVE*

- *Exemplo2\_1.java*

-*exemplo2\_1.java*

**são classes distintas para o compilador Java**

# Primeiras Aplicações em Java

```
public class NomeDaClasse
{
    public static void main ( String[] arg)
    {
        comandos...
    }
}
```

-Abre o arquivo de classe;

-Indica o início da classe;

-Abre o início do método “main”;

-Inicia a escrita do corpo do método;

-Corpo do método “main”;

-(conjunto de instruções que compõem o método)

-Inicia a escrita do corpo do método;

-Indica o final da classe;

## O método *main*

É o método principal de um programa, este método transforma uma classe em um programa executável.

```
public static void main ( String[] arg)
{
    comandos...
}
}
```

**É A MESMA COISA**

```
public static void main ( String arg[])
{
    comandos...
}
}
```

# O método *main*

## O COMANDO

```
System.out.println(" Uma frase")
```

Gera uma saída de texto para o vídeo do computador. E somente pode exercer a sua função se estiver definido dentro do corpo do método *main*.

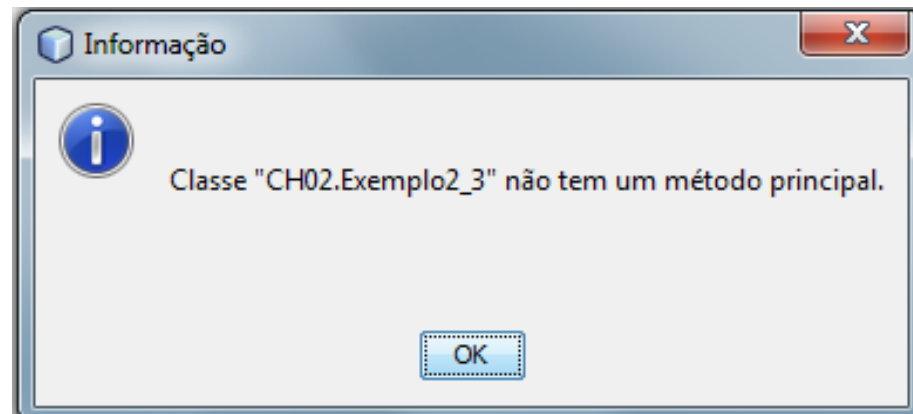
*Uma classe pode ter muitos métodos escritos e ser definida por uma grande quantidade de valores constantes e variáveis de diversos tipos.*

## O método *main*

**VAMOS EXECUTAR O EXEMPLO ABAIXO:**

```
public class Exemplo2
{
    public static void escrever()
    {
        int x = 10 ;
        String nome = "Valor =";
        System.out.println(nome+" "+x);
    }
}
```

O método anterior não  
funcionou porque?





### FALTA O MÉTODO PRINCIPAL

### O “*main*”

```
public class Exemplo2_4
{
    public static void main ( String arg[])
    {
        Exemplo2_3.escrever();
    }
}
```

Vamos implementar e executar agora....

## O método *main*

```
Import javax.swing.*; // importa o pacote de classes visuais de Java
```

```
public class Exemplo3
```

```
{
```

```
    public static void main (String[] arg)
```

```
    {
```

```
        // apresenta uma caixa de entrada de dado
```

```
        String Nome = JOptionPane.showInputDialog(null,"Digite um nome");
```

```
                String Resp= "O meu nome é = "+Nome ; // Cria a String de saída
```

```
        // apresenta uma caixa de mensagem com o resultado do programa
```

```
                JOptionPane.showMessageDialog(null,Resp,"Mensagem",1);
```

```
    }
```

```
}
```

## O método *main*

```
package CH02;

import javax.swing.*;

public class Exemplo2_6
{
    public static void main ( String[] arg)
    {
        JOptionPane.showMessageDialog(null,"O meu
nome é Juca","Mensagem",1);
        System.exit(0);
    }
}
```

## Recursos básicos da linguagem

São necessários para armazenar na memória do computador valores específicos para serem utilizados posteriormente pelo processo

```
// Declaração de variáveis  
int num1 = 0, op = 0;  
double valor;  
String usuario;
```

```
// Declaração de constantes  
final double pi = 3.1416;
```

- Tipos primitivos são escritos sempre com letras minúsculas.
- As variáveis podem (e em alguns casos devem) ser inicializadas na declaração
- O Java possui classes que são utilizadas como tipos comuns de dados, normalmente existentes em outras linguagens, baseados em tipos primitivos (String, um array de char).

# Funcionamento da Memória

Apresentação dos dados na memória.

	<b>num1</b>			<b>pi</b>
	0			3.1416
<b>valor</b>			<b>op</b>	
		<b>usuario</b>	0	

# Tipos Primitivos

Classificação	Tipo	Descrição
Lógico	boolean	Pode possuir os valores true (verdadeiro) ou false (falso)
Inteiro	byte	Abrange de -128 a 127 (8 bits)
	short	Abrange de -32768 a 32767 (16 bits)
	int	Abrange de -2147483648 a 2147483647 (32 bits)
	long	Abrange de $-2^{63}$ a $(2^{63})-1$ (64 bits)
Ponto Flutuante	float	Abrange de $1.40239846^{-46}$ a $3.40282347^{+38}$ com precisão simples (32 bits)
	double	Abrange de $4.94065645841246544^{-324}$ a $1.7976931348623157^{+308}$ com precisão dupla (64 bits)
Caracter	char	Pode armazenar um caracteres unicode (16 bits) ou um inteiro entre 0 e 65535

## Tamanho dos tipos

Na tabela abaixo, estão os tamanhos de cada tipo primitivo do Java.

<b><i>TIPO</i></b>	<b><i>TAMANHO</i></b>
boolean	1 bit
byte	1 byte
short	2 bytes
char	2 bytes
int	4 bytes
float	4 bytes
long	8 bytes
double	8 bytes

# Classe String

Uma string em Java é nada mais nada menos do que um conjunto de caracteres separados por aspas e que podem ser atribuídos diretamente para variáveis do tipo String.

## **Declaração de uma variável do tipo string:**

```
string nome;
```

## **Inicialização de uma variável string:**

```
nome = “Etec Irmã Agostina”;
```

## **Imprimindo na Tela do Usuário:**

```
System.out.println(“Meu nome é : ”+nome);
```

## **Pode-se declarar e inicializar uma variável:**

```
string nome = “Etec Irmã Agostina”;
```

```
System.out.println(“Meu nome é : ”+nome);
```



## Palavras reservadas em Java

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

## Declaração Atributos e Variáveis

- **Atributos e variáveis** são a mesma coisa em questão de funcionalidade. Ambos são endereços de memória que tem um espaço ou tamanho definido de acordo com o tipo de dado que será guardado,
- **Por exemplo:** caracter, número, número decimal, etc.
- **Em Java**, costumamos utilizar o termo atributo, que é nada além do que uma variável que está dentro de uma classe. Como tudo que fazemos em Java está contido dentro de uma classe, então usamos o termo atributo ao invés de variável.

# Declaração de Variáveis

**Exemplo:** package exemplo;

```
public class TiposPrimitivos {  
    public static void main(String args[ ]) {  
        char sexo = 'f';  
        byte idade = 89;  
        short codigo = 256;  
        float nota = 9.4f;  
        int alunos = 100, classes = 10;  
        long habitantes = 9050100;  
        double dolar = 2.62;  
        boolean alternativa = false;  
  
        System.out.println("\n sexo:" + sexo + " idade:" + idade + " codigo:" + codigo);  
        System.out.println("\n nota:" + nota + " alunos:" + alunos + " classes:" + classes);  
        System.out.println("\n habitantes:" + habitantes + " dolar:" + dolar + " alternativa:" +  
alternativa);  
    }  
}
```

## Declaração de Constantes

É um tipo de variável que não pode alterar seu conteúdo depois de ter sido inicializado, ou seja, o conteúdo permanece o mesmo durante toda a execução do programa.

**Exemplo:**

```
final double pi = 3.14;
```

```
final int miliSegundosPorSegundo = 1000;
```

## Declaração de Constantes - Exemplo

```
public class Constantes {  
  
    static final float PI = 3.14159265f;  
  
    public static void main(String args[]) {  
        float raio = 25f;  
        float comprimento = raio * 2 * PI;  
        float area = (raio * raio) * PI;  
        System.out.println("Dados de um círculo de " + raio +  
"cm:\n" + "comprimento: " + comprimento + "cm\n"+ "Área:  
" + area + "cm2");  
    }  
}
```

## Casting e promoções

- Alguns valores são incompatíveis se você tentar fazer uma atribuição direta.
- Enquanto um número real costuma ser representado em uma variável do tipo double, tentar atribuir ele a uma variável int não funciona porque é um código que diz:
- "i deve valer d", mas não se sabe se d realmente é um número inteiro ou não.

## Casting e promoções

```
double d = 3.1415;
```

```
int i = d; // não compila
```

- O mesmo ocorre no seguinte trecho:

```
int i = 3.14;
```

- O mais interessante, é que nem mesmo o seguinte código compila:

```
double d = 5; // ok, o double pode conter um número inteiro
```

```
int i = d; // não compila
```

- Apesar de 5 ser um bom valor para um int , o compilador não tem como saber que valor estará dentro desse double no momento da execução.
- Esse valor pode ter sido digitado pelo usuário, e ninguém vai garantir que essa conversão ocorra sem perda de valores.

## Casting e promoções

```
int i = 5;
```

```
double d2 = i;
```

- Compila sem problemas, já que um double pode guardar um número com ou sem ponto flutuante.
- Todos os inteiros representados por uma variável do tipo int podem ser guardados em uma variável double, então não existem problemas no código acima.



## Casting e promoções

- Às vezes, precisamos que um número quebrado seja arredondado e armazenado num número inteiro.
- Para fazer isso sem que haja o erro de compilação, é preciso ordenar que o número quebrado seja moldado (casted) como um número inteiro.

## Casting e promoções

```
double d3 = 3.14;
```

```
int i = (int) d3;
```

- O casting foi feito para moldar a variável d3 como um int .
- O valor de i agora é 3.

# Casting possíveis

- Abaixo estão relacionados todos os casts possíveis na linguagem Java, mostrando a conversão de um valor para outro.
- A indicação \* **Impl.** quer dizer que aquele cast é implícito e automático, ou seja, você não precisa indicar o cast explicitamente (lembrando que o tipo boolean não pode ser convertido para nenhum outro tipo).

PARA:	byte	short	char	int	long	float	double
DE:							
<b>byte</b>	----	<i>Impl.</i>	(char)	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
<b>short</b>	(byte)	----	(char)	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
<b>char</b>	(byte)	(short)	----	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
<b>int</b>	(byte)	(short)	(char)	----	<i>Impl.</i>	<i>Impl.</i>	<i>Impl.</i>
<b>long</b>	(byte)	(short)	(char)	(int)	----	<i>Impl.</i>	<i>Impl.</i>
<b>float</b>	(byte)	(short)	(char)	(int)	(long)	----	<i>Impl.</i>
<b>double</b>	(byte)	(short)	(char)	(int)	(long)	(float)	----

\* Ao declarar esse tipo de conversão, o compilador identifica que ambos são de tipos compatíveis, sendo assim nenhuma intervenção na hora do build é feita e há a garantia que nenhuma informação seja perdida durante a conversão.

# Operadores

Destinados à realização de operações aritméticas, lógicas e relacionais, com a possibilidade de formar expressões de qualquer tipo.

## Operadores Aritméticos:

Operador	Significado
+	adição
-	subtração
*	multiplicação
/	divisão
%	resto da divisão (módulo)

# Operadores

Destinados à realização de operações aritméticas, lógicas e relacionais, com a possibilidade de formar expressões de qualquer tipo.

## Operadores Aritméticos:

Operador	Significado
+	adição
-	subtração
*	multiplicação
/	divisão
%	resto da divisão (módulo)

# Operadores Aritméticos:

```
package exemplo;

public class OperadoresAritmeticos{

    public static void main (String args[]){

        // declaracao e inicializacao de variaveis

        int x = 10;    int y = 3;

        // varias operacoes com as variaveis

        System.out.println("X = "+ x);

        System.out.println("Y = "+ y);

        System.out.println("-X = "+(-x));

        System.out.println("X/Y = "+(x/y));

        System.out.println("Resto de X por Y = "+ (x%y)); // resulta 1

        System.out.println("Inteiro de X por Y = "+ (int)(x/y)); // resulta 3

        System.out.println("X + 1 = "+ (++x)); // resulta 11

    }

}
```

## Operadores de Atribuição

Operador	Exemplo	Expressão equivalente
<code>+=</code>	<code>x += y</code>	<code>x = x + y</code>
<code>-=</code>	<code>x -= y</code>	<code>x = x - y</code>
<code>*=</code>	<code>x *= y</code>	<code>x = x * y</code>
<code>/=</code>	<code>x /= y</code>	<code>x = x / y</code>
<code>%=</code>	<code>x %= y</code>	<code>x = x % y</code>

# Operadores de Atribuição

```
/**
```

Classe utilizada para demonstrar o uso do operador de atribuição ( = ).

```
*/
```

```
public class OperadorAtribuicao {
```

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

```
        int x;
```

```
        x = 25;
```

```
        System.out.println(x);
```

```
    }
```

```
}
```



# Operadores de Incremento e Decremento

Operador	Exemplo	Significado
++	++a	adicionar 1 à variável <code>a</code> e depois calcular a expressão na qual <code>a</code> reside
	a++	calcular a expressão na qual <code>a</code> reside e depois adicionar 1 à variável <code>a</code>
--	--a	subtrair 1 da variável <code>a</code> e depois calcular a expressão na qual <code>a</code> reside
	a--	calcular a expressão na qual <code>a</code> reside e depois subtrair 1 da variável <code>a</code>

# Operadores de Incremento e Decremento

```
//Programa de Atribuicao
//Pacotes de extensão Java
import javax.swing.JOptionPane;

public class Incremento2 {

    // Método main início da aplicação Java
    public static void main( String args[] )
    {
        // inicializando variável
        int c;

        c = 5;

        JOptionPane.showMessageDialog( null, "c = 5 ",
            "Incremento : ",
            JOptionPane.INFORMATION_MESSAGE );

        JOptionPane.showMessageDialog( null, "++c -> " + (++c),
            "Incremento : ",
            JOptionPane.INFORMATION_MESSAGE );

        JOptionPane.showMessageDialog( null, "c -> " + c ,
            "Incremento : ",
            JOptionPane.INFORMATION_MESSAGE );

        System.exit( 0 ); // término da aplicação

    } // Fim do método main
} // Fim da classe Incremento2
```

## Operadores de Comparação

Operador	Significado
<code>==</code>	igual a
<code>!=</code>	diferente de
<code>&lt;</code>	menor que
<code>&gt;</code>	maior que
<code>&lt;=</code>	menor ou igual a
<code>&gt;=</code>	maior ou igual a

## Caixa de Diálogo para Entrada de Dados

- Trata-se da utilização de caixas de diálogo, no caso a caixa gerada a partir da **classe JOptionPane**.
- Essa classe é utilizada para facilitar a entrada de informações via teclado rapidamente. O fato de receber dados de forma gráfica parece dar mais motivação aos iniciantes em Java, pois facilita na visualização e interação do usuário com o sistema.

# Caixa de Diálogo para Entrada de Dados

```
package ProjetoJava;
import javax.swing.*;

public class EntradaComJOptionPane {
    public static void main(String args[]) {
        String aux = "";
        double nota1 = 0, nota2 = 0, trabalho = 0, media = 0;
        try {
            aux = JOptionPane.showInputDialog(null, "Entre com a nota 1");
            nota1 = Double.parseDouble(aux);

            aux = JOptionPane.showInputDialog(null, "Entre com a nota 2");
            nota2 = Double.parseDouble(aux);

            aux = JOptionPane.showInputDialog(null, "Entre com a nota do trabalho");
            trabalho = Double.parseDouble(aux);

            media = (nota1 + nota2 + trabalho) / 3;
            JOptionPane.showMessageDialog(null, "Media : " + media);
        }
        catch (NumberFormatException erro) {
            JOptionPane.showMessageDialog(null, "Houve erro na conversão, digite apenas caracteres
numéricos"+ erro.toString());
        }
        System.exit(0);
    }
}
```

# Caixa de Diálogo para Entrada de Dados

```
package cap02;
import javax.swing.*;
public class Notas {
    public static void main(String args[]) {
        String aux = "";
        double nota1 = 0, nota2 = 0, trabalho = 0, media = 0;
        aux = JOptionPane.showInputDialog(null, "Entre com a nota 1");
        nota1 = Float.parseFloat(aux);

        aux = JOptionPane.showInputDialog(null, "Entre com a nota 2");
        nota2 = Float.parseFloat(aux);

        aux = JOptionPane.showInputDialog(null, "Entre com a nota do trabalho");
        trabalho = Float.parseFloat(aux);

        media = (nota1 + nota2 + trabalho) / 3;
        JOptionPane.showMessageDialog(null, "Media : " + media);
    }
}
```

**Exercício**

**Gasto Trimestral**

# Muito Obrigado

Até a Próxima Aula

Prof. Laércio Silva  
Email: [Indsilva@hotmail.com](mailto:Indsilva@hotmail.com)