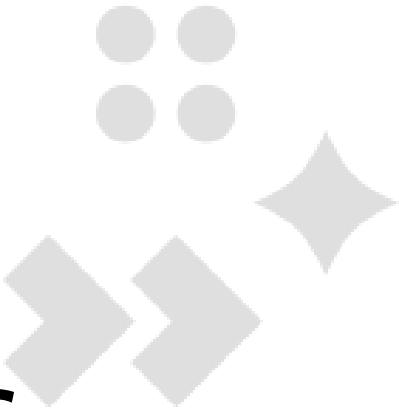


Programação Orientada a Objetos

Turma 3

Carlos Henrique Bughi, MSc



Onde estamos?
(e para onde vamos)



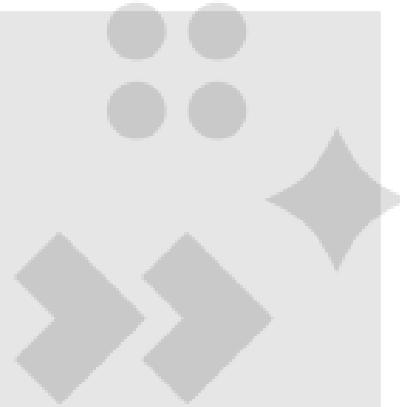
- Avaliação M1
- Introdução Java

Programar, programar e programar



Aula 7

Introdução a linguagem Java e a
Programação Orientada a Objetos



História da linguagem Java

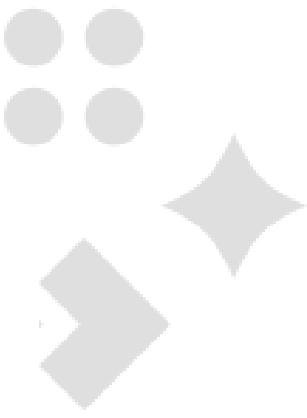
- Desenvolvida e mantida pela ~~Sun Microsystems~~ Oracle ;
- Planejada inicialmente para ser usada no desenvolvimento de programas executados por processadores de eletrodomésticos, como geladeiras e torradeiras;
- No início de 1990, Patrick Naughton, Sun Fellow e James Gosling, começaram a definir as bases para o projeto de uma nova linguagem de programação, apropriada para eletrodomésticos, sem os problemas já tão conhecidos de linguagens tradicionais como C e C++.



Java é ao mesmo tempo uma **linguagem de programação** e uma **plataforma**. Você entenderá essa diferença mais à frente, estudando as próximas unidades.



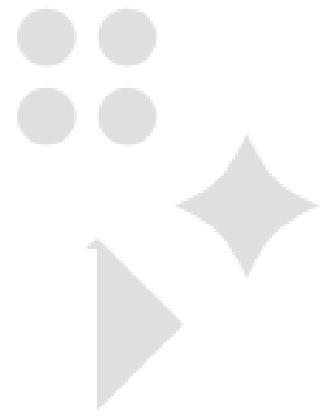
História da linguagem Java



- A especificação da linguagem terminou em agosto de 1991 e recebeu o nome de “*Oak*” (Carvalho);
- Por problemas de *copyright*, o nome foi mudado em 1995 para **Java**, em homenagem à ilha de Java, de onde vinha o café consumido pela equipe da Sun;
- Em 1992, Oak foi utilizada pela primeira vez em um projeto chamado Projeto Green, que tinha por propósito desenvolver uma nova interface de usuário para controlar os aparelhos de uma casa;



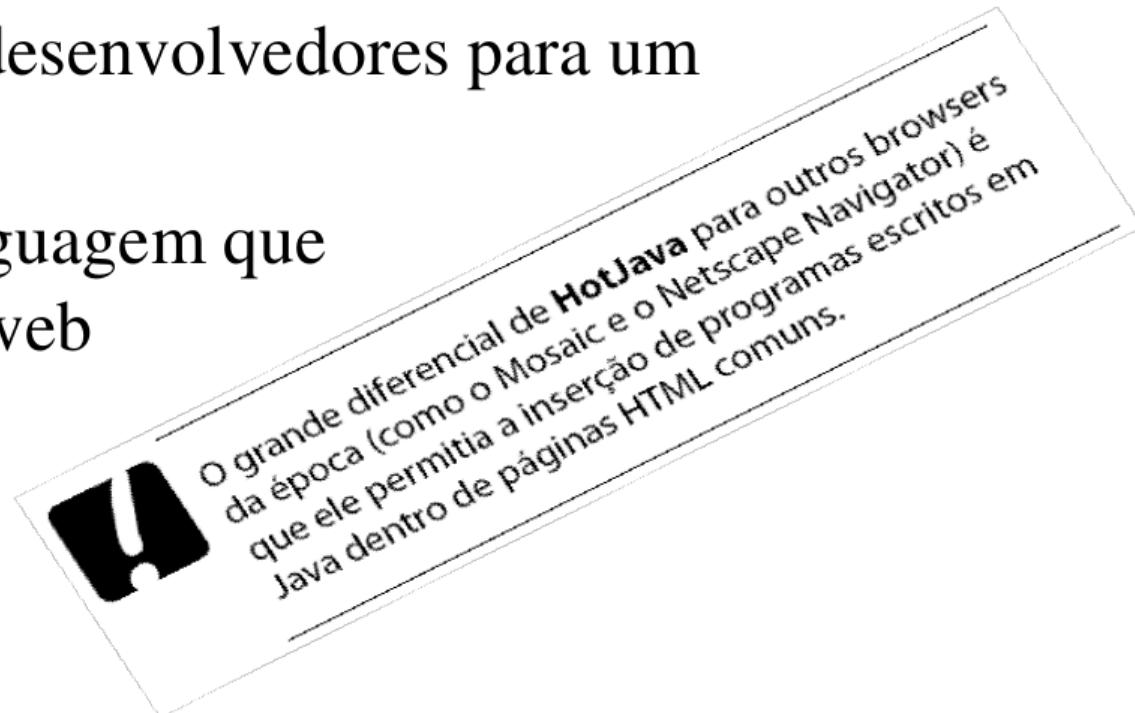
História da linguagem Java



- Essa interface foi totalmente escrita em Oak e evoluiu de um projeto de interface para redes de televisão *pay-per-view*;
- Contudo, o padrão proposto por esses dois projetos não vingou, sendo economicamente inviáveis e sem grandes futuro no desenvolvimento de aparelhos que suportassem essa nova linguagem;
- Nesta mesma época, surgia a World Wide Web, trazendo um novo horizonte para a Internet;

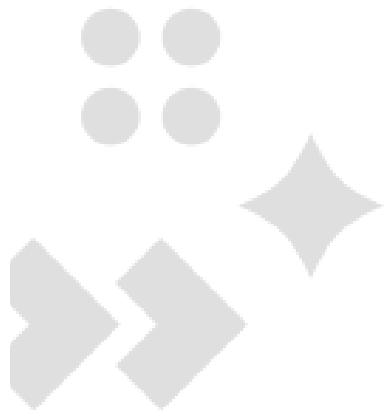
História da linguagem Java

- Em 1995, aproveitando as características “portáveis” da linguagem, a equipe da Sun desenvolveu um navegador totalmente escrito em java, denominado HotJava;
- HotJava como browser não teve sucesso comercial, mas abriu os olhos dos desenvolvedores para um fato muito importante:
- Necessidade de uma linguagem que tornasse o conteúdo da web dinâmico;





História da linguagem Java



- O grande avanço de Java veio logo a seguir, quando a Netscape anunciou que sua próxima versão do browser Navigator iria dar suporte a aplicativos Java embutidos em documentos HTML;
- Em seguida, a Microsoft anunciou o mesmo para o seu Internet Explorer. Após isso, Java estourou no mundo e começou a ser utilizada também fora da internet no desenvolvimento de softwares stand-alone;
- E como dito anteriormente Java é ao mesmo tempo uma linguagem de programação e uma plataforma, conforme você verá a seguir.



Linguagem de programação Java



- Enquanto linguagem de programação, Java é considerada uma linguagem de alto nível que possui características como:
 - Simplicidade;
 - Interpretada;
 - Orientada a objetos;
 - Multiplataforma;
 - Segura.

Linguagem de programação Java

- Simplicidade

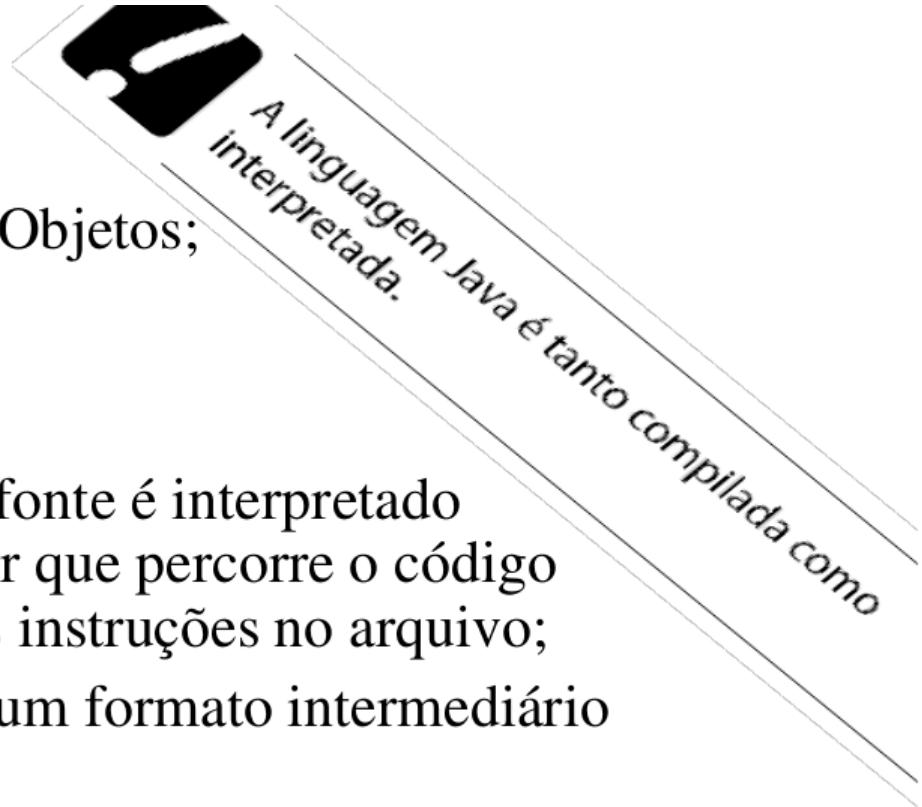
- Java é uma linguagem simples e de fácil aprendizado ou migração, pois possui um reduzido número de construções e sua sintaxe é baseada na linguagem C;
- Contém uma biblioteca (API Java) de programas (conhecidos em Java como classes) que fornecem grande parte da funcionalidade básica da linguagem, incluindo rotinas de acesso à rede e criação de interface gráfica.



API: *Application Programming Interface* ou Interface de Programação de Aplicativos.

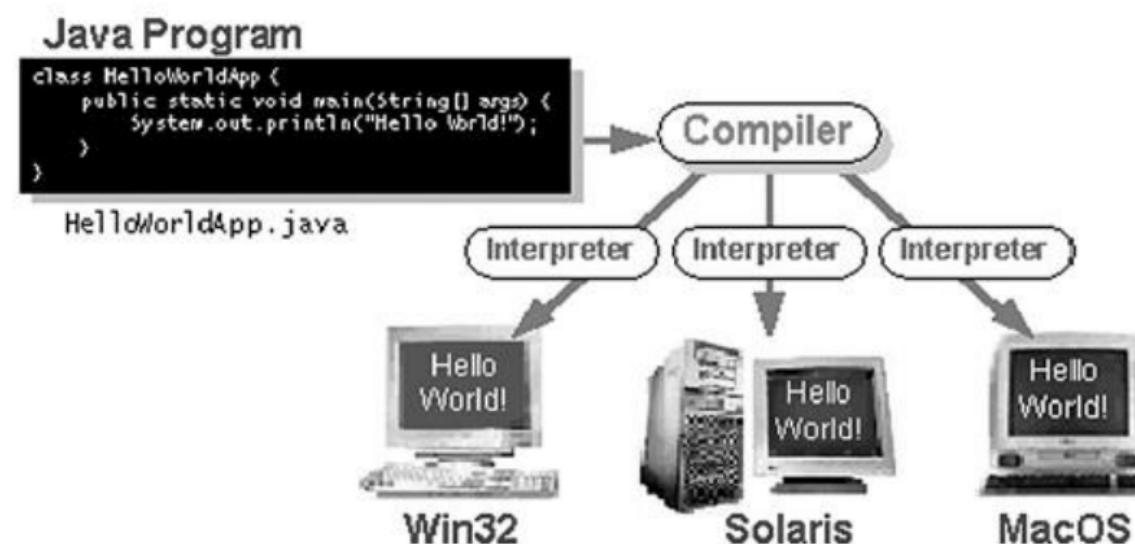
Linguagem de programação Java

- Orientada a Objetos
 - Baseada no paradigma da Orientação a Objetos;
- Interpretada
 - Nas linguagens interpretadas, o código fonte é interpretado por um programa chamado interpretador que percorre o código fonte e executa as ações indicadas pelas instruções no arquivo;
 - Os programas Java são compilados em um formato intermediário chamado bytecode.
 - Esse bytecode só pode ser executado em qualquer plataforma ou sistema operacional através de um interpretador Java (máquina virtual) específico de uma plataforma ou sistema operacional.



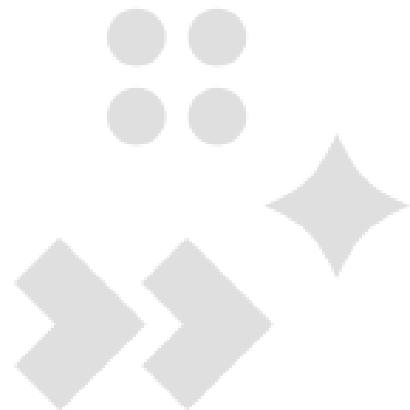
Linguagem de programação Java

- Multiplataforma
 - A característica de multiplataforma da linguagem Java indica que um programa desenvolvido nela pode ser executado em plataformas de hardware e sistemas operacionais diferentes;





Linguagem de programação Java



- Segurança
 - O processo de compilação - geração de bytecodes - é projetado para a detecção prévia dos possíveis erros, evitando que os erros se manifestem em tempo de execução;
 - O uso de código para tratamento de exceções - *exception handling* - permite manter a consistência da aplicação no caso de erros;

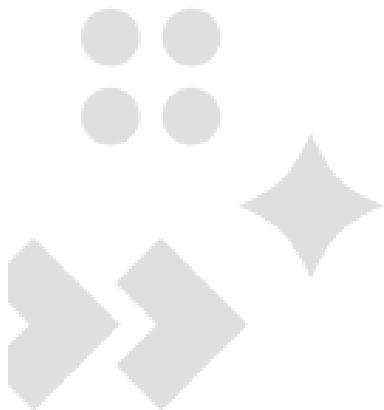
Plataforma Java

- Exemplos de plataformas populares são: Windows, Linux, Solaris, e MacOS.
- A maioria das plataformas podem ser descritas como uma combinação de sistema operacional e hardware.
- A plataforma Java difere das outras plataformas no sentido de que é uma plataforma baseada em software que roda em cima de outras plataformas baseadas em hardware.





Plataforma Java



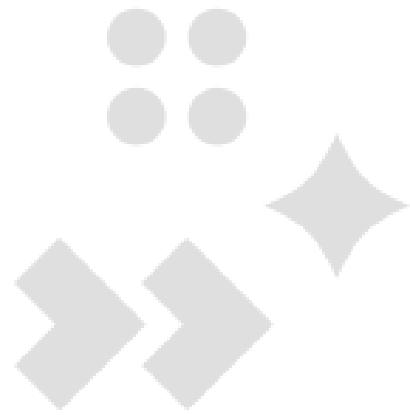
- Dois componentes principais
 - **Maquina Virtual** – Java Virtual Machine – JVM
 - **API Java** – Java Application Programming Interface

A **máquina virtual** nada mais é do que um *software* que é instalado sobre uma plataforma de *hardware*-sistema operacional e é necessária para interpretar os *bytecodes* gerados pela compilação de um código fonte em Java.

A **API Java** é uma grande coleção de componentes de *software* (programas-classes prontas) que podem ser utilizados no desenvolvimento de programas na linguagem Java, como, por exemplo, componentes que permitem a construção de interfaces gráficas (telas) de programas. Esses componentes de software, na realidade, são conjuntos de **classes** e **interfaces** organizadas em pacotes (*package*).

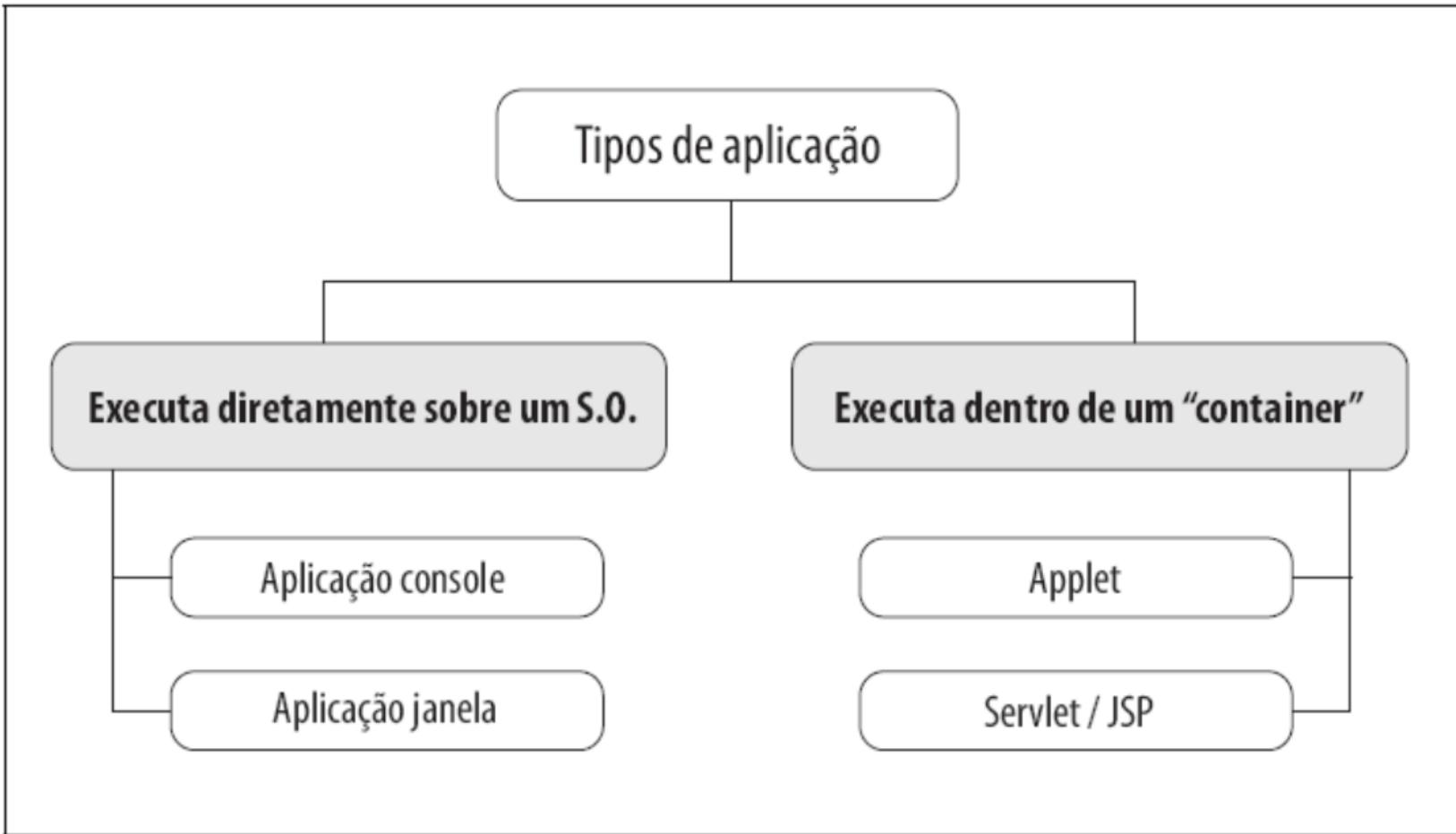


Tipos de aplicações Java



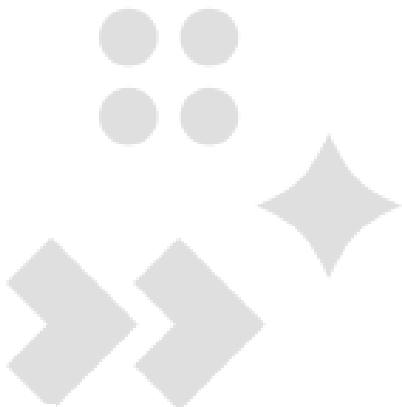
- Os programas ou aplicações escritos em Java podem ser de dois tipos:
 - Aqueles que são executados diretamente pela máquina virtual no sistema operacional.
 - Aqueles que são executados através de outro programa chamado genericamente de “container”.

Tipos de aplicações Java





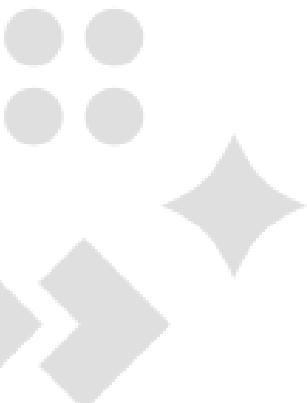
Tipos de aplicações Java



- **Aplicação console**
 - são aplicações Java que são executadas a partir da JVM instalada no sistema operacional e o seu resultado é mostrado na tela do shell (por exemplo, janela do MS-DOS), sem recursos gráficos.
- **Aplicação janela**
 - são aplicações Java que são executadas a partir da JVM instalada no sistema operacional, porém, utiliza recursos gráficos como janelas, etc.



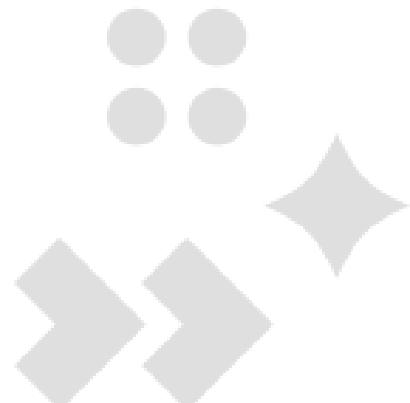
Tipos de aplicações Java



- Applet
 - são aplicações Java que precisam de um web browser (navegador web, por exemplo, Internet Explorer) para serem executados. Isso quer dizer que applets são aplicações voltadas para o ambiente Internet/Intranet e que são transportadas pela rede junto com hiperdocumentos HTML.
- Servlet/JSP
 - tipo de programa Java que pode, por exemplo, receber dados de um formulário HTML por meio de uma requisição HTTP, processar os dados, atualizar a base de dados de uma empresa e gerar alguma resposta dinamicamente para o cliente que fez a requisição.
 - Ou seja, é através do desenvolvimento de servlets que podemos desenvolver uma APLICAÇÃO WEB DINÂMICA que interaja com o usuário.



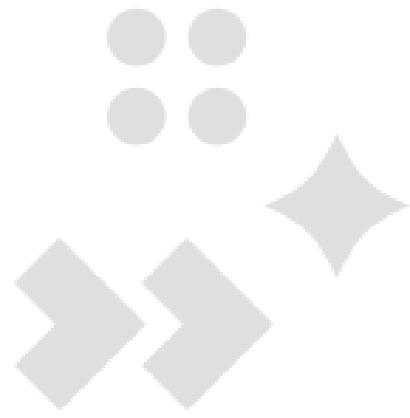
Preparando o ambiente



- 1º. Passo: escolha da plataforma desejada
 - **J2SE (Java 2 Platform, Standard Edition)** é a base para as demais plataformas, voltada para a criação de sistemas para pcs e servidores;
 - **J2EE (Java 2 Platform, Enterprise Edition)** é aplicada principalmente no desenvolvimento de sistemas para redes e Internet, com acesso a servidores, a sistemas de e-mail e banco de dados;
 - **J2ME (Java 2 Platform, Micro Edition)**, plataforma de desenvolvimento de sistemas para dispositivos móveis ou portáteis, como telefones celulares e palmtops.



Preparando o ambiente



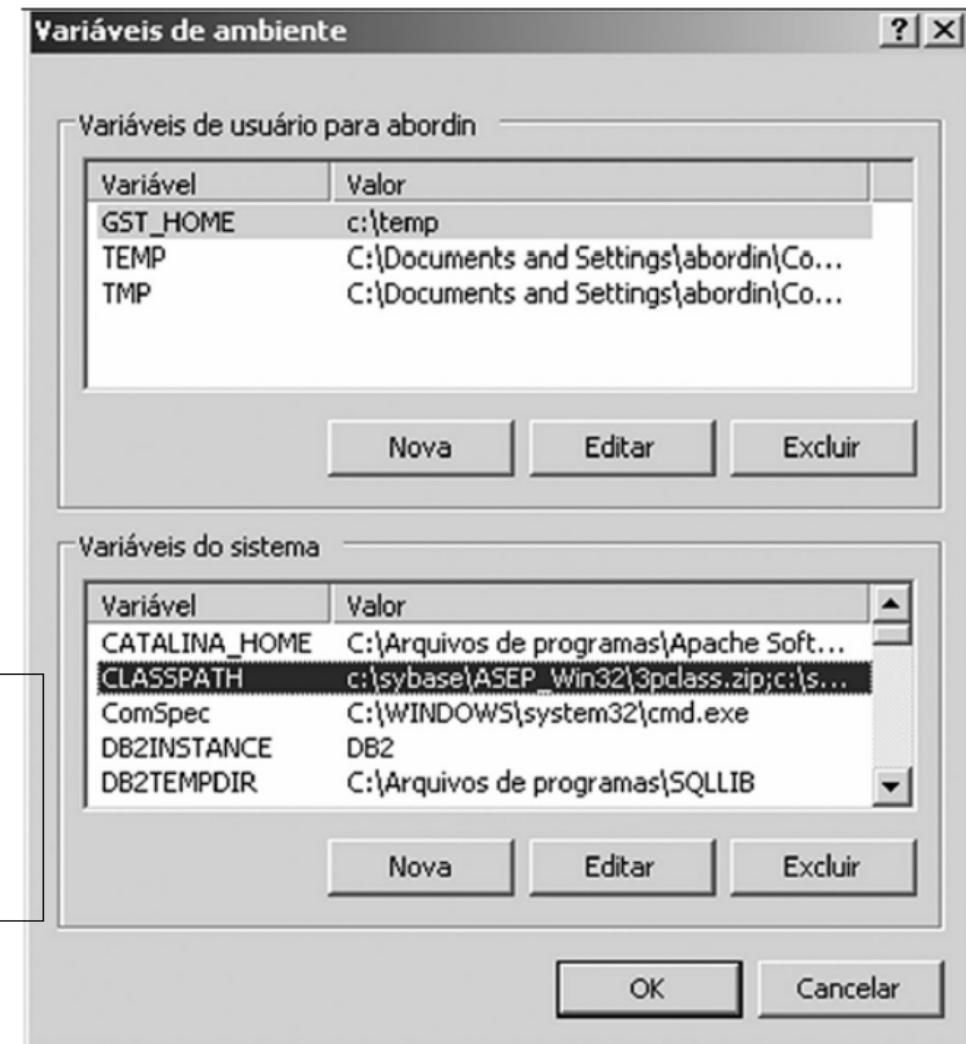
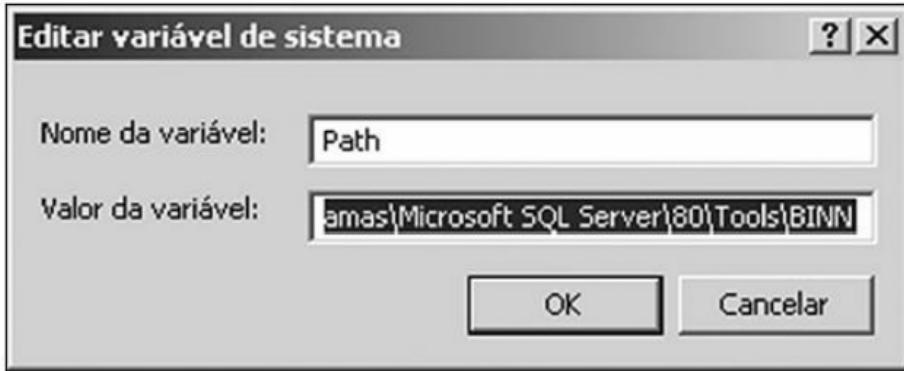
- 2º. Passo: instalação do JDK
 - O Java 2 Development Kit (JDK) consiste de programas como o compilador, a JVM (Java Virtual Machine) e a API Java necessários para o desenvolvimento de aplicações;
 - O JDK pode ser baixado em:
www.oracle.com/technetwork/java/javase/downloads
 - O procedimento de instalação no Windows é muito simples, basta você executar o arquivo, seguir os passos e instalar no diretório desejado.



IMPORTANTE !!

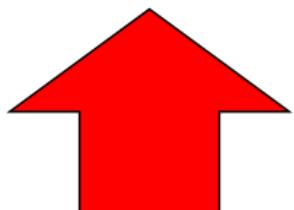
Após a instalação, é necessário configurar duas **variáveis de ambiente** do sistema operacional. Essas variáveis se chamam **PATH** e **CLASSPATH**.

Preparando o ambiente



Exemplos

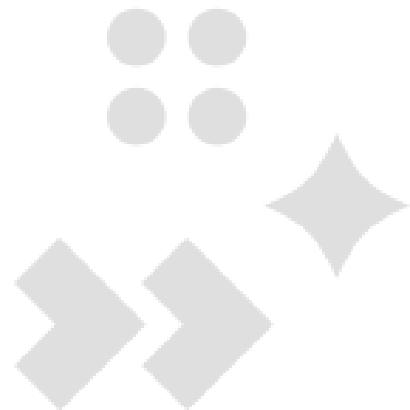
- JAVA_HOME C:\Program Files\Java\jdk1.7.0_05
- PATH C:\Program Files\Java\jdk1.7.0_05\bin
- CLASSPATH .;C:\Program Files\Java\jdk1.7.0_05\lib





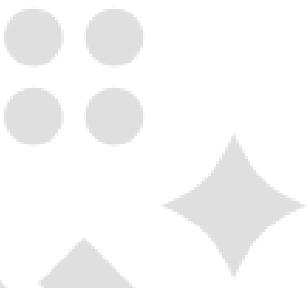
Preparando o ambiente

- 3º. Passo: testando o ambiente
 - No prompt de comando, digite:
 - java.exe <enter>
 - javac.exe <enter>





Primeiro programa em Java



- Os programas em Java consistem em partes chamadas classes.
- Essas, por sua vez, consistem em partes chamadas métodos que realizam tarefas e muitas vezes retornam informações ao completarem tarefas.
- Os programadores podem construir novas classes ou podem utilizar as classes existentes em bibliotecas de classes prontas que Java possui.
- Essas bibliotecas de classes são conhecidas como Java API (Application Programming Interface ou Interface de Programas Aplicativos).



Primeiro programa em Java



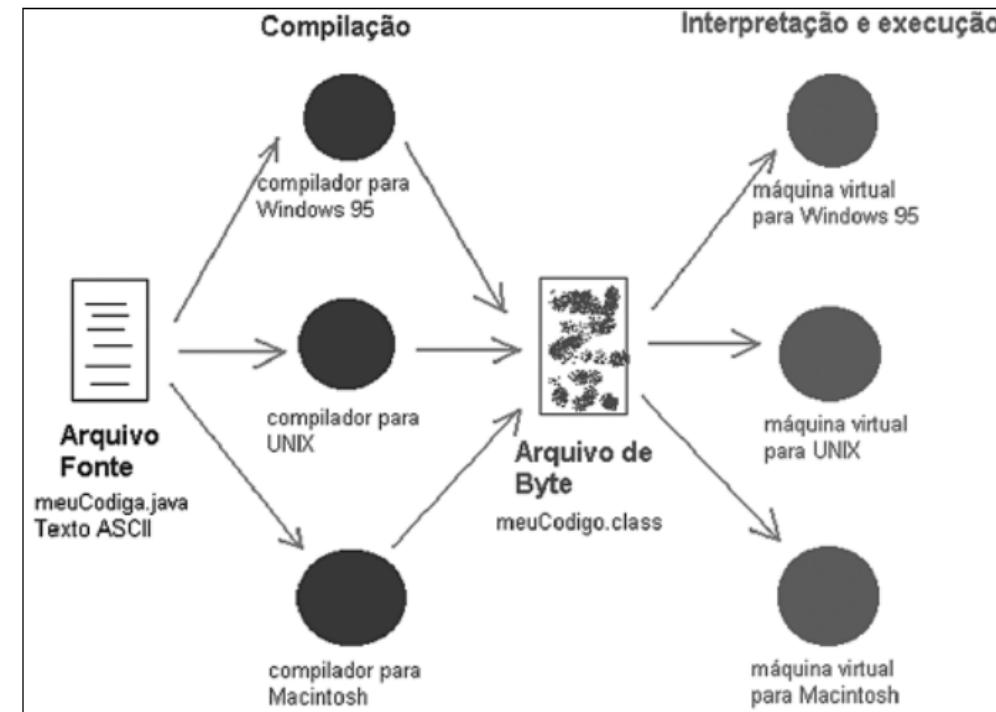
- Desenvolver um programa em Java consiste basicamente de três etapas.
 - Edição:
 - Desenvolvimento do programa de acordo com a sintaxe da linguagem, utilizando um software editor de texto como o Bloco de Notas do Windows ou uma IDE (Integrated Development Environment - ambiente de desenvolvimento integrado) como o JBuilder, NetBeans, Eclipse.
 - O arquivo contendo o programa deve ser salvo com a extensão .java
 - Compilação
 - Após a edição do programa, o mesmo deve ser compilado, ou seja, é gerado um arquivo contendo os bytecodes do programa.
 - O arquivo contendo o bytecode possui a extensão .class

Primeiro programa em Java

- Desenvolver um programa em Java consiste basicamente de três etapas.

- Interpretação

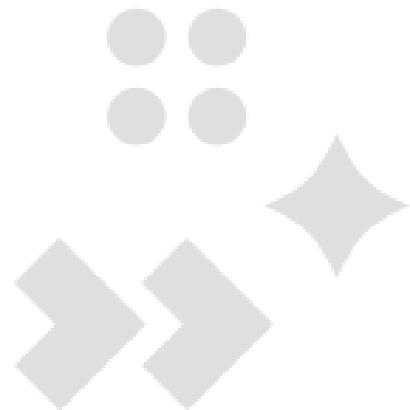
- O processo de interpretação é realizado a partir dos bytecodes gerados pela compilação, que estão no arquivo .class.
 - Realizado pela JVM instalada no computador a qual é dependente do sistema operacional do mesmo.





Primeiro programa em Java

- Como compilar
 - **javac nomedoprograma.java**
 - javac é o nome do software compilador que deve ser executado.
- Como interpretar
 - **java nomedoprograma.class**
 - java é o nome do software que representa a JVM instalada no computador.



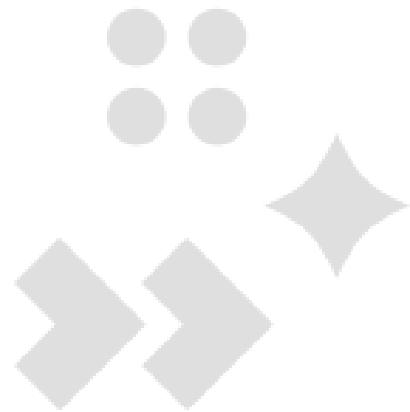


Primeiro programa em Java

- Criando o primeiro programa
 - No bloco de notas, digite o código abaixo

```
1 public class AloMundo
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Alo Mundo!");
6     }
7
8 }
```

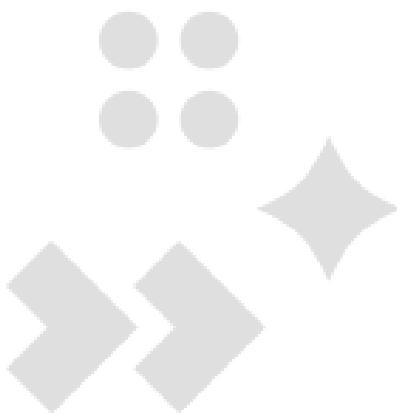
- Salve o arquivo na raiz com o nome AloMundo.java



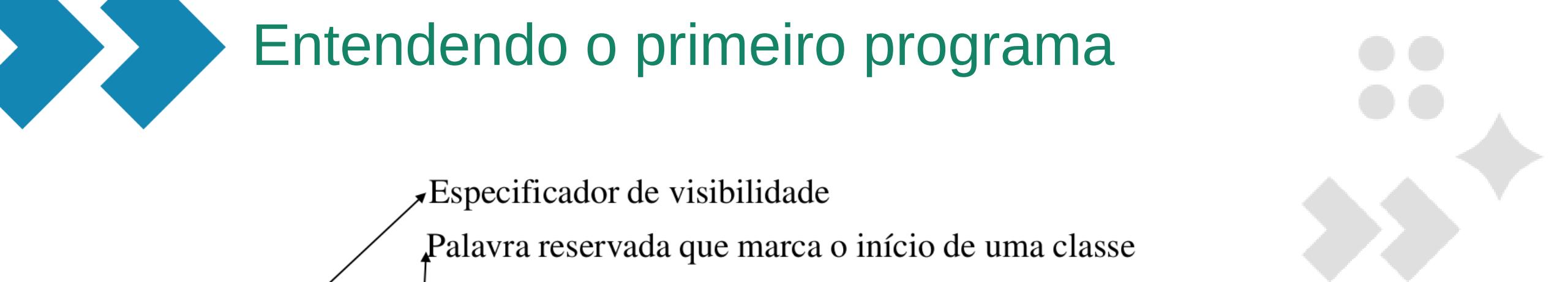


Primeiro programa em Java

- Criando o primeiro programa
 - Compilando o arquivo
 - javac AloMundo.java
 - Se tudo deu certo, foi criado um arquivo AloMundo.class
 - Executando o arquivo
 - java AloMundo
 - devemos executar os programas feitos em Java sem usar a extensão do arquivo (.class)
 - Resultado?? ☺
 - Alo Mundo!



Entendendo o primeiro programa



```
1 public class AloMundo
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Alo Mundo!");
6     }
7 }
8 }
```

1 Especificador de visibilidade
2 Palavra reservada que marca o início de uma classe
3 Nome dado a classe
4 Marca o início e fim das declarações da classe
5 Por enquanto, único método da classe AloMundo

Entendendo o primeiro programa

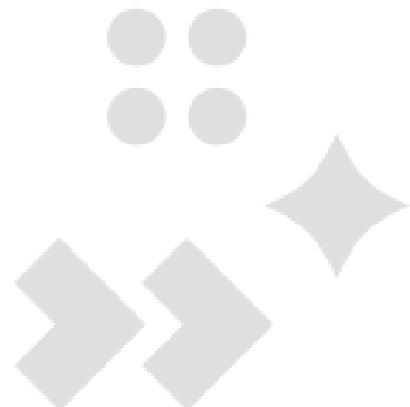
```
3 public static void main(String[] args)
4 {
5     System.out.println("Alo Mundo!");
6 }
```

- **public**: qualificador do método, neste caso indica que está acessível externamente a classe;
- **static**: outro qualificador, indica que o método deve ser compartilhado por todos os objetos criados a partir da classe;
- **void**: semelhante ao void do C++, indica que o método não possui nenhum retorno;
- **main**: nome do método, obrigatório o uso “main” para indicar para o compilador o início do programa;
- **(String[] args)**: é o argumento do método main e consequentemente do programa todo. Consiste em um array formado por strings com os valores passados durante a iniciação do programa na linha de comando;

Ex: java AloMundo argumentotexto1 argumentotexto2



Entendendo o primeiro programa



5

```
System.out.println("Alo Mundo!");
```

- Realiza uma chamada do método **println** para o atributo **out** da classe **System**;
- O argumento é uma constante do tipo String para imprimir a cadeia “Alo Mundo!” e posicionar o cursor na linha abaixo;



Exercício 1

Escreva uma classe AloMundo2.java que receba da linha de comando dois parâmetros: uma String com o nome da pessoa e um int com a idade da pessoa;

Mostre na saída padrão com System.out a frase:

“ Bom dia “+args[0]+” voce tem “+args[1]+” anos de idade!”



Identificadores

Um identificador (atributo, constante, nome do método) começa com:

Uma letra; ou

Hífen-caixa-baixa (underline) _ ; ou

Símbolo dólar (\$);

Os caracteres subsequentes podem conter dígitos;

Caracteres maiúsculo e minúsculo são diferentes e não tem tamanho máximo;

Identificadores válidos:

identifier	nomeUsuario	nome_usuario
_sys_var	\$change	

Evitar nome iniciado com \$ por causa das variáveis de ambiente do SO.

Não utilizar espaços e palavras reservadas.



Palavras reservadas

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

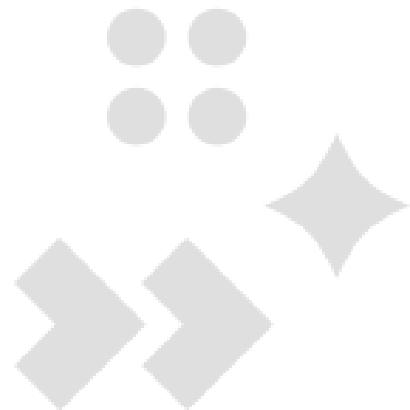
Tipos primitivos

Tipo	Tamanho	Faixa de valores	Wrapper
int	4 bytes	-2.147.483.648 até 2.147.483.647	Integer
short	2 bytes	-32.768 até 32.767	Short
byte	1 byte	-128 até 127	Byte
long	8 bytes	-9.223.372.036.854.775.808 até 9.223.373.036.854.775.807	Long
float	4 bytes	1,4E-45 a 3,4E+38	Float
double	8 bytes	4,9E-324 a 1,7E+308	Double
char	2 bytes	-	Character
<u>void</u>	-	-	Void
boolean	1 byte	true ou false	Boolean

Classes Wrapper são classes especiais para cada um dos tipos primitivos. Sua função é dotar cada tipo com métodos para resolver problemas do tipo: Converte objeto String para o tipo primitivo int.



Declaração de variáveis

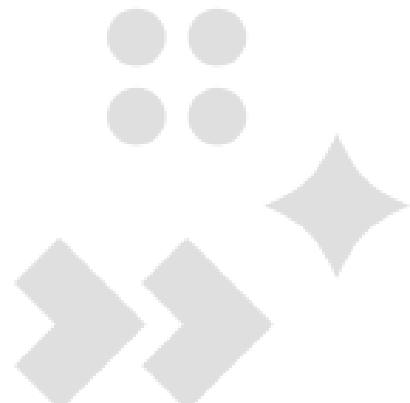


Sintaxe para definição de variáveis:

tipoPrimitivo nome = valorInicial;



Declaração de variáveis



Exemplos:

```
int x,y; //declarando variáveis inteiras
```

```
float z = 3.414; // ponto flutuante
```

```
boolean atualiza = false; // booleano
```

```
char c; // declarando variável caracter
```

```
c = 'A'; //atribuindo valor a char;
```

Operadores aritméticos

Operador aritmético	Uso	Significado
+	op1 + op2	retorna a soma de op1 e op2
-	op1 – op2	retorna a subtração de op1 e op2
*	op1 * op2	retorna a multiplicação de op1 por op2
/	op1 / op2	retorna a divisão de op1 por op2
%	op1 % op2	retorna o resto da divisão de op1 por op2
++	op++	retorna o valor de op, depois o incrementa de 1
++	++op	incrementa op de 1, depois retorna o valor
--	op--	retorna o valor de op, depois o decrementa em 1
--	--op	decrementa op de 1, depois retorna o valor

A divisão retorna resultado inteiro se os operadores forem inteiros e valores de ponto flutuante (com ponto) em caso contrário.

Operadores relacionais

Operador relacional	Uso	Significado
<code>==</code>	<code>op1 == op2</code>	retorna true se op1 é igual a op2
<code>!=</code>	<code>op1 != op2</code>	retorna true se op1 é diferente de op2
<code><</code>	<code>op1 < op2</code>	retorna true se op1 é menor do que op2
<code>></code>	<code>op1 > op2</code>	retorna true se op1 é maior do que op2
<code><=</code>	<code>op1 <= op2</code>	retorna true se op1 é menor ou igual a op2
<code>>=</code>	<code>op1 >= op2</code>	retorna true se op1 é maior ou igual a op2

Operadores lógicos

Operador relacional	Uso	Significado
&&	op1 && op2	op1 e op2 forem true. Só avalia op2 se op1 for true
	op1 op2	op1 ou op2 for true (ou ambos). Só avalia op2 se op1 for false
!	! Op	Negação
&	op1 & op2	op1 e op2 forem true. Sempre avalia op1 e op2
	op1 op2	op1 ou op2 for true (ou ambos). Sempre avalia op1 e op2
!=	op1 != op2	op1 diferente de op2



Controle de fluxo



Decisão:

if-else

Switch-case

Repetição:

for

while

do-while

Controle de fluxo

```
if  (expressão booleana)
    comando ou bloco
else
    comando ou bloco
```

```
if( <condição> ){
    <comando 1>;
    <comando 2>;
}
else{
    <comando 3>;
    <comando 4>;
}
```

```
switch (expressão1)
{
case expressão2:
    comando ou bloco
    break;
.
.
.
case expressãon:
    comando ou bloco
    break;
default:
    comando ou bloco
    break;
}
```



Controle de fluxo



```
while (expressão booleana)  
    comando ou bloco
```

```
do  
    comando ou bloco  
while(expressão booleana);
```

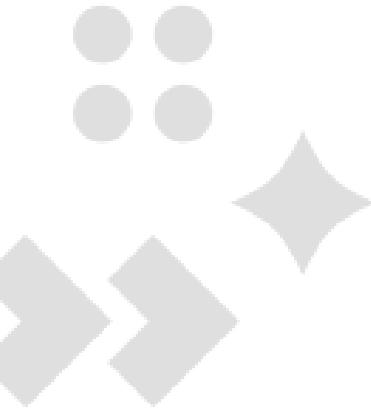
```
for (expressão inicial;  
     expressão booleana;  
     expressão de iteração)  
    comando ou bloco
```

Exercício 2

Crie uma classe “Número” que receba X números via linha de comando, mostre o menor, o maior e a média dos números informados.

Dicas:

- Para não ter perda de precisão no cálculo da média utilize **double** ou **float** para armazenar o resultado.
- Para saber quantos números foram passados através da linha de comando utilize a constante de array **args.length** que retorna o número de elementos.
- Lembre que os valores informados em **args** são **Objetos String** e para serem utilizados como **int** devem sobre uma conversão através do **Integer.parseInt(args[x])**.



Exercício 3

Escreva uma classe que receba, via linha de comando, 10 notas (em valor inteiro) e, ao final, mostre um gráfico conforme exemplo abaixo. Ao final, apresente a média dos alunos (média = soma das notas/total de alunos)

considerando os valores de entrada 4 5 7 4 3 6 10 ..., o sistema apresentará:

Aluno 01 ****

Aluno 02 *****

Aluno 03 *****

Aluno 04 ***

Aluno 05 **

Aluno 06 *****

Aluno 07 *****

(e assim até o décimo aluno)

Exercício 4

Exercícios de fixação – variáveis primitivas e controle de fluxo

<http://www.javaprogressivo.net/2012/12/Apostila-Caelum-resolvida-Capitulo-03-Variaveis-primitivas-e-Controle-de-fluxo-Tipos-primitivos-e-valores.html>

