



Prof. Dr. Fernando Almeida  
[proffernando.almeida@fiap.com.br](mailto:proffernando.almeida@fiap.com.br)





# DDD (Domain Driven Design) Java SE e Java EE

# O QUE VAMOS APRENDER HOJE?

## Introdução a Linguagem Java

1

Características da Linguagem Java

2

Java Virtual Machine - JVM

3

Boas práticas e convenções

4

Variáveis e Tipos de dados

5

Conversões de tipos

6

Classe Scanner



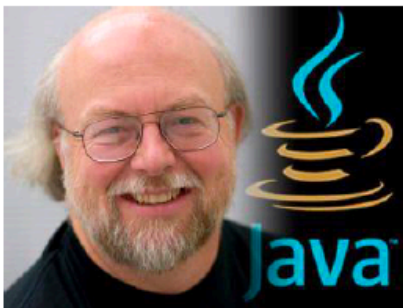
# Introdução a Linguagem Java

## História do Java



# História da Linguagem JAVA

Criada por James Gosling da Sun Microsystems e lançada em 1996













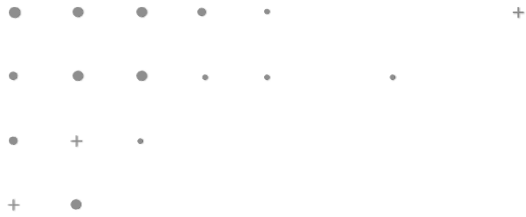
Nome inicial era OAK (Carvalho) e depois para Java

Criada inicialmente para execução em televisões, eletro-domésticos, telefones, etc

Utilizou como ponto de partida o mesmo estilo de sintaxe do C/C++

Utiliza o conceito de WORA (Write Once, Run Anywhere)

	Feb 2022	Feb 2021	Change	Programming Language		Ratings	Change
1		3	▲		Python	15.33%	+4.47%
2		1	▼		C	14.08%	-2.26%
3		2	▼		Java	12.13%	+0.84%
4		4			C++	8.01%	+1.13%
5		5			C#	5.37%	+0.93%
6		6			Visual Basic	5.23%	+0.90%
7		7			JavaScript	1.83%	-0.45%
8		8			PHP	1.79%	+0.04%
9		10	▲		Assembly language	1.60%	-0.06%
10		9	▼		SQL	1.55%	-0.18%



# Características da Linguagem Java



# Características da Linguagem **JAVA**



---

Maior facilidade na programação

---

Escreva uma vez, execute em qualquer lugar

---

Portabilidade do código

---

Possibilidade de programas executarem mais de uma tarefa (*multithreading*);

---

Programação centrada na rede

---



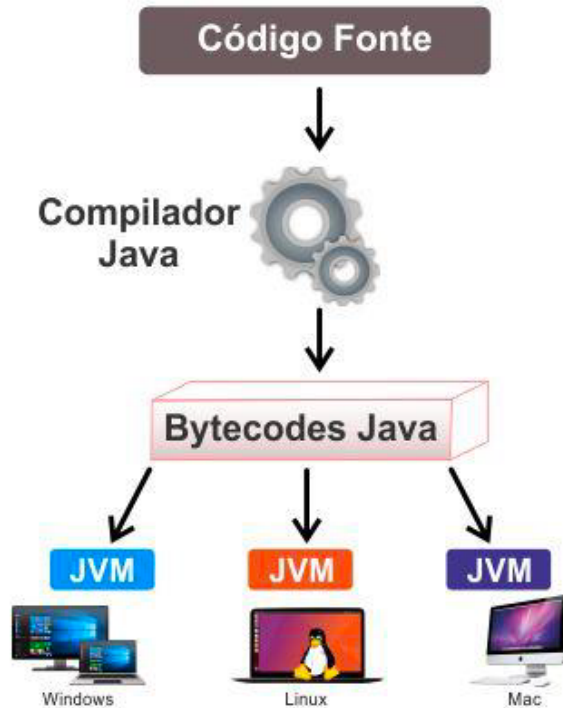


"Java não é apenas uma linguagem de programação...  
é uma **completa plataforma de desenvolvimento e  
execução**, composta por três pilares:

**Máquina Virtual Java (JVM)**  
**Conjunto de APIs (bibliotecas)**  
**Linguagem Java**

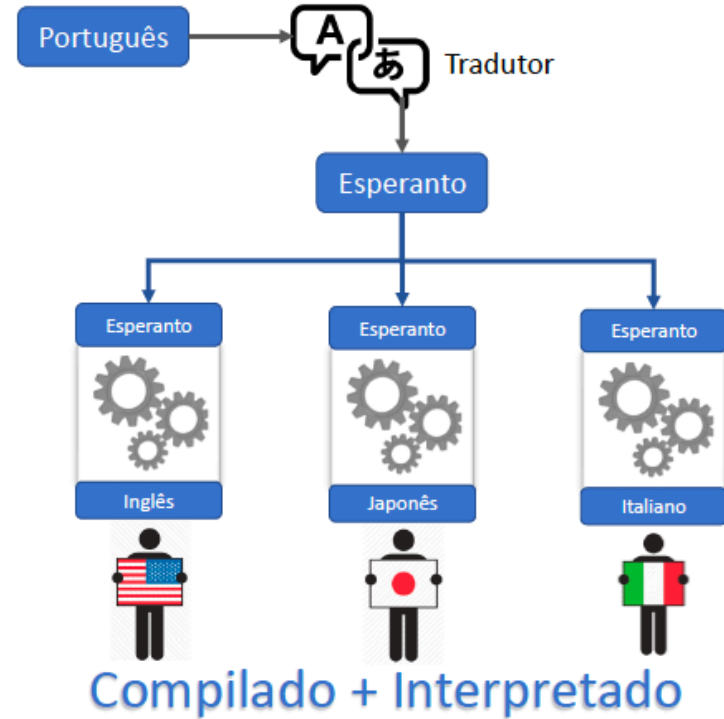
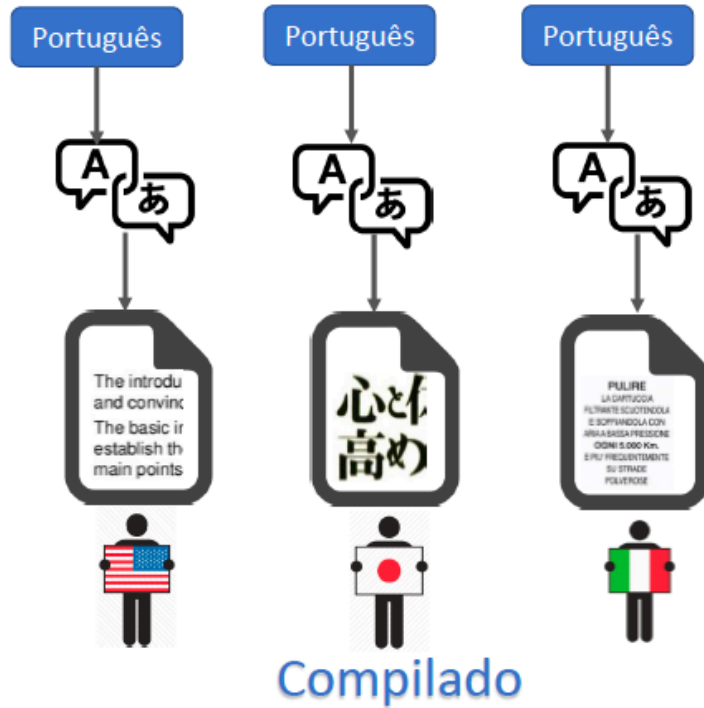


# Java Virtual Machine - JVM



- Simula uma máquina real para o *bytecode*
- Interpreta *bytecodes* (independentes de hardware)
- Pode ser implementada tanto na forma de software como de hardware
- Possui código compacto
- Torna a Linguagem Java Portável para diversas plataformas
- Qualquer interpretador Java (seja para desenvolvimento de tecnologia Java ou *browser* que rode *applets*) tem sua máquina virtual

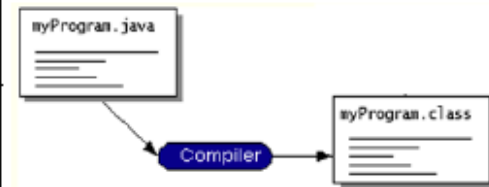
# Analogia entre Compilado e Interpretado



# Sêquência de execução Java

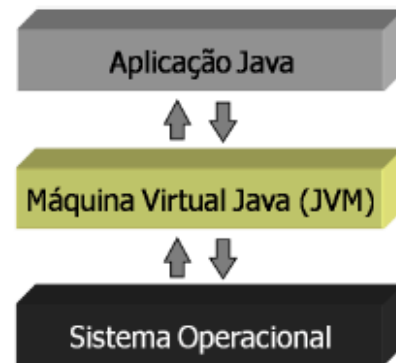
## Criação do programa

- O programa é criado no editor e armazenado em disco
- O compilador cria bytecodes e os armazena em disco.



## Execução do programa

- O carregador (loader) de classe coloca bytecodes na memória.
- O verificador de bytecodes confirma que todos os bytecodes são válidos e não violam restrições de segurança do Java
- O interpretador lê os bytecodes e os traduz para uma linguagem que o computador pode entender, possivelmente armazenando valores dos dados enquanto executa o programa



# Sêquência de execução Java



- O Java é uma linguagem de alto nível (compreensível por humanos)
- A JVM **não** entende código Java! ...ela entende um código de máquina específico (*bytecodes*)
- Esse código de máquina é gerado por um compilador java, como o **javac**, e é conhecido por **bytecode (binários)**.
- A tradução de Java para *bytecodes* é feita na compilação do código Java



O que baixar?  
JVM, JRE, JDK...



# Siglas comuns

- **JVM - Java Virtual Machine**, apenas a virtual machine (não existe).
- **JRE - Java Runtime Environment**, ambiente de execução Java, formado pela JVM e bibliotecas.
- **JDK - Java Development Kit**: Ele é formado pela JRE somado a ferramentas, como o compilador.
- Tanto o **JRE** e o **JDK** podem ser baixados do site:
- <http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- **Java EE** - Java Enterprise Edition
- **Java SE** - Java Standard Edition
- **Java ME** - Java Micro Edition
- dentre outras...

# Estrutura básica de um programa em Java

```
package meupacote;
```

**Package.** Utilizado quando o código do programa deverá fazer parte de um pacote.

```
import java.lang.*;
```

**Import.** Seção de importação de bibliotecas.

```
/** Primeiro programa Java  
Conhecendo a estrutura de um  
programa Java */
```

**Comentários.** Com sintaxe `/** ...` para comentários simples ou `/* ... */` e a mais recente `/** .. */` que permite geração de documentação automática (ferramenta javadoc)

```
public class MinhaClassePublica {  
.....  
/** Comentário sobre o método */  
    public (private/protected) tipoRet  
    nomeMetodo(<parametros>) {  
        // código do método  
    } // fim da definição do método  
} // fim da classe
```

**Classes.** Declaração de classes, atributos e métodos do programa Java. A declaração e a definição dos métodos ocorre obrigatoriamente dentro do limite de declaração da classe.

**Método main().** Indica que a classe Java é um aplicativo que será interpretado pela máquina virtual.



# Elementos da Programação Estruturada

```
// Nosso primeiro programa Java
// Conhecendo a estrutura de um programa Java
public class HelloWorld {
    public static void main (String arg[]) {
        System.out.println("Hello, World!");
    } // fim do método main
} // fim da classe MeuPrimeiroPrograma
```

**Função Principal.** Programas em Linguagem C e C++ e Java buscam seu início pela função principal (main()).

**Parâmetros.** Parâmetros em funções permitem que essas iniciem com valores recebidos externamente, para variáveis que utilizarão internamente.

# Elementos da Programação Orientada a Objeto

```
// Nosso primeiro programa Java
// Conhecendo a estrutura de um programa Java
public class HelloWorld {
    public static void main (String arg[]) {
        System.out.println("Hello, World!");
    } // fim do método main
} // fim da classe MeuPrimeiroPrograma
```

**Método.** A impressão da mensagem "Hello, World!" se deu pela execução do método "println" da classe "System".

**Classe.** Como qualquer programa JAVA, ele exige uma classe (palavra reservada "class"). Por ser Publica (palavra "public"), isso garante visibilidade em qualquer contexto de sua utilização

**Objeto.** Para imprimir a mensagem de saída do programa é necessário o objeto "out" da classe "System" da biblioteca padrão java.lang

**Biblioteca.** A organização das classes JAVA se dá na forma de bibliotecas. Nesse programa utilizamos a biblioteca padrão da linguagem JAVA (biblioteca java.lang)

# Resumo

Comentário de bloco

Nome da classe

Nome do método

**Declaração** de argumento

variável local: args  
tipo: String[]

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

**Atribuição** de argumento  
para o método println()

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

**Definição** de método main()

**Definição** de classe  
HelloWorld

```
/** Aplicação Hello World */
```

```
public class HelloWorld {
```

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

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

```
    }
```

```
}
```

# Palabras Reservadas

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	enum				



# Como começar um programa em Java?

```
1 class MinhaClasse {  
2  
3     //variáveis, métodos  
4  
5 }
```

- **Classe** - abstração do mundo real
- **Estado** (variáveis e atributos) - define as características
- **Ações** (métodos) - define o comportamento da classe

## Exemplo: criação de um cadastro de alunos

```
1  class Aluno {  
2  
3      String nome;  
4      String matricula;  
5  
6      public void realizarMatricula() {}  
7  
8  }
```



# Tipos de Dados em Java





# Tipos de dados<sup>+</sup>

- Tipos primitivos:

Tipo	Descrição
<code>boolean</code>	Pode assumir os valores <code>true</code> ou <code>false</code>
<code>char</code>	Representa um caractere Unicode de 16 bits para armazenar dados alfanuméricos
<code>byte</code>	Inteiro de 8 bits. Pode assumir valores entre $-2^7$ e $2^7-1$ (de -128 a 127)
<code>short</code>	Inteiro de 16 bits. Pode assumir valores entre $-2^{15}$ e $2^{15}-1$ (de -32.768 a 32.767)
<code>int</code>	Inteiro de 32 bits. Pode assumir valores entre $-2^{31}$ e $2^{31}-1$ (de -2.147.483.648 a 2.147.483.647)
<code>long</code>	Inteiro de 64 bits. Pode assumir valores entre $-2^{63}$ e $2^{63}-1$
<code>float</code>	Número de ponto flutuante de 32 bits. Pode assumir valores entre $\pm 1.40129846432481707e-45$ e $\pm 3.40282346638528860e+38$
<code>double</code>	Número de ponto flutuante de 64 bits. Pode assumir valores entre $\pm 4.94065645841246544e-324$ e $\pm 1.79769313486231570e+308$

- String:

Tipo	Descrição
<code>String</code>	Cadeia de caracteres que usam 2 bytes por caractere. Strings podem ser vazias (zero caractere) e conter qualquer tipo de caractere.

# Declaração de variáveis

tipo do atributo

**int**

**pesoVeiculo;**

identificador  
(nome do atributo)

## Exemplos:

```
float precoProduto;  
int idadeAluno;  
char conceito;  
String nome_Aluno;  
boolean maiorIdade;
```

# Exemplo de identificadores

O identificador (nome da variável) é formado por caracteres `Unicode`. Eles podem ser formados por letras, cifrão (\$), *underline*(\_) e números (**não pode ser iniciado com número**)

```
1 public class Variaveis {
2
3     public static void main(String args[]){
4         String nome;    // válido
5         int $idade;     // válido
6         double 1preco;   // inválido
7         double preco1;  // válido
8         int ___$;       // válido
9         String :nome;   // inválido
10    }
11
12 }
```

# Atribuição

pesoVeiculo = 1500

identificador  
(nome do atributo)

valor do atributo

## Exemplos:

```
float precoProduto = 3.5;  
int idadeAluno = 18;  
char conceito = 'A';  
String nome_Aluno = "João";  
boolean maiorIdade = false;
```

# Exemplo de atribuição

- + .
- + • Uma vez declarada, a variável deve ser inicializado, e após isso, ser modificada e utilizada. Elas podem ser declaradas e inicializadas em uma mesma linha de código.

```
1 String nome;           // declara uma variável do tipo String
2 nome = "Frederico Maia"; // inicializa com um valor
3 int idade = 21;         // declara e inicializa na mesma linha
4 System.out.println(nome+" "+idade); // imprime o valor das variáveis na tela
```

# Operadores Aritméticos

• + •  
+ •

| +  
media = somaNotas / 2;  
Expressão

operador

## Exemplos:

Operação	Operador	Expressão Algébrica	Expressão Java
Adição	+	$X + 1$	$X + 1$
Subtração	-	$Y - 2$	$Y - 2$
Multiplicação	*	$K \cdot X$	$K * X$
Divisão	/	$C / 2$	$C / 2$
Resto	%	$X \text{ mod } Y$	$X \% Y$

+  
•  
•  
•  
• + •  
•  
• • •

# Precedência de Operadores Aritméticos

• + •

+ •

|

+

Operador	Operação	Expressão Algébrica
* / %	Multiplicação, Divisão e Resto	É o primeiro a ser avaliado. A ordem de avaliação é da esquerda para a direita.
- +	Subtração e soma	É avaliado posteriormente. A ordem de avaliação também é da esquerda para a direita.

•

• +

■ □ •

• •

•

• + • •

• • •

# Operadores Relacionais

operador

$\perp$

(media  $\geq$  6)

Respostas  
**true** ou **false**

## Exemplos:

Operação	Operador Matemático	Operador Java	Exemplo	Significado
Igual	=	==	X == Y	X é igual a Y
Diferente	$\neq$	!=	X != Y	X é diferente de Y
Maior	>	>	X > Y	X é Maior que Y
Menor	<	<	X < Y	X é menor que Y
Maior ou Igual	$\geq$	>=	X >= Y	X é maior ou igual a Y
Menor ou Igual	$\leq$	<=	X <= Y	X é menor ou igual a Y

Permite saber a relação existente  
entre seus dois operandos



# Operadores Lógicos

resulta em **true**

$(2 > 1) \text{ || } (3 < 7)$

resulta em **false**

$(3 > 2) \text{ \&\& } (2 == 2)$

resulta em **true**

$(5 != 0) \text{ || } (1 < 2)$

resulta em **false**

**!true**

## Exemplos:

Operação	Operador Matemático	Operador Java	Exemplo
OU	$\vee$	<code>  </code>	<code>(notaEnem &gt; 6)    (notaRedacao == 10)</code>
E	$\wedge$	<code>&amp;&amp;</code>	<code>(mediaFinal &gt;= 6) &amp;&amp; (totalFaltas &lt; 25%)</code>
Negação	$\sim$	<code>!</code>	<code>!pendenciaDocumento</code>

conectam duas ou mais expressões relacionais



# Convenções de Código Java





# Java é Case Sensitive!

significa que Java diferencia letras maiúsculas e minúsculas



# Convenções de Código em Java

- “80% do tempo e dos custos gastos com software estão relacionados com atividades de manutenção” (*Sun Microsystem*)
- Boa prática de programação
- Possibilidade de desenvolver códigos mais legíveis
- Maior qualidade
- Melhor entendimento do código
- Aplicáveis em todo desenvolvimento: classes, interfaces, métodos, variáveis e constantes...



# Meu primeiro programa em Java

## Hello World!

# HelloWorld.java

- O nome do arquivo deve ser o mesmo nome da classe pública e ter a extensão *.java*
- Exemplo: **HelloWorld.java**

```
1 public class HelloApp {  
2     public static void main(String args[]) {  
3         System.out.println("Hello World!");  
4     }  
5 }
```

- Compilando e executando em linha de comando

```
1 cd c:/java           // entra no diretório c:/java  
2 javac HelloWorld.java // compila a classe HelloWorld.java e gera o arquivo HelloWorld.class  
3 java HelloWorld      // executa o arquivo HelloWorld.class e imprime na tela: "Hello World"
```

# Método `main()`

Quando uma classe é executada, a máquina virtual procura pelo método `main()`. Este método é o ponto de partida de todas as aplicações Java, e deve ser assinado desta forma:

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



# Comentários no código

## única linha, várias linhas, documentação





# Comentários em Java

- São textos ignorados pelo compilador, mas que podem ser útil para nós humanos
- Utilizados para explicar funcionalidade, melhorar a compreensão do que foi implementado

```
1 | // texto ignorado pelo compilador
```

Comentários de apenas uma linha (//)

```
1 | /* texto que pode conter
2 |    várias linhas */
```

Comentários de várias linhas (/\* \*/)

```
1 | /** comentários de documentação,
2 |    começam com barra e dois asteriscos */
```

Comentários de documentação (/\*\* \*/)



# Conversões de Variáveis implícitas ou explícitas



# Conversões de variáveis

Conversões

## IMPLICITAS

Nenhuma sintaxe especial é necessária porque a conversão é de tipo seguro e nenhum dado será perdido. Exemplo:

- byte para int ou long para float → `byte b = 10; int i = b;`

Conversões

## EXPLICITAS

As conversões explícitas exigem um operador cast. A conversão é necessária quando as informações podem ser perdidas na conversão ou quando a conversão pode não funcionar por outros motivos. Exemplo:

- Float para long ou int para byte → `int i = 10; byte b = (byte)i;`

Conversões

## OUTRAS

O restante dos tipos de conversão disponíveis ou criados no programa. Exemplo:

- String para int → `String s = "123"; int i = Integer.parseInt(s);`

# Conversões de variáveis

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



# Entrada de Dados (console)

## Classe Scanner



# Recebendo dados do teclado

- A forma mais comum de recebermos dados informados pelo usuário é através do teclado
- Java possui em sua API uma classe para nos auxiliar - **Scanner**
- A classe `Scanner` possui métodos (funcionalidades) que permitem ao usuário informar valores do tipo `int`, `long`, `float`, `double` e `String`
- A classe `Scanner` pode ser utilizada para entrada de dados através de um arquivo de texto ou através do teclado

# Entrada de dados (console) em JAVA

• + •

- Captura a entrada de dados via console
- Possui diversos métodos para facilitar a leitura de dados pelo teclado

```
1 package Aula1;
2
3 import java.util.Scanner;
4
5 /**
6  *
7  * @author Fernando
8  */
9 public class IntroducaoScanner {
10     public static void main(String[] args) {
11         Scanner input = new Scanner(System.in);
12         String nome;
13         System.out.println("Nome: ");
14         nome = input.next();
15         System.out.println("Olá, " + nome);
16     }
17 }
```

# Métodos e tipos de retorno da classe Scanner

Método	Tipo de Retorno
int nextInt()	Retorna o valor lido como um int. Se o valor não for um inteiro, ou estiver fora de sua faixa, lança uma exceção.
long nextLong()	Retorna o valor lido como um long. Se o valor não for um long, ou estiver fora de sua faixa, lança uma exceção.
float nextFloat()	Retorna o valor lido como um float. Se não for um float ou estiver fora de sua faixa, lança uma exceção.
double nextDouble()	Retorna o valor lido como um double. Se não for um double ou estiver fora de sua faixa, lança uma exceção.
String next()	Retorna o valor lido como uma <b>String</b> . A função termina ao encontrar um espaço em branco. Caso entre com um valor como “ <i>Sou Java</i> ”, apenas a palavra “ <i>Sou</i> ” será capturada pelo método e retornada.
String nextLine()	Retorna a String informada, mesmo com espaços. Neste caso, se entrar com um valor como “ <i>Sou Java</i> ”, este mesmo texto será retornado.
void close()	Fecha o Scanner.



# Exemplo

```
1 import java.util.Scanner;
2
3 public class Media {
4
5     public static void main(String[] args) {
6         String nome = "";
7         double nota1 = 0.0;
8         double nota2 = 0.0;
9         double nota3 = 0.0;
10        double media = 0.0;
11
12        Scanner entrada = new Scanner(System.in);
13
14        System.out.println("Informe seu nome: ");
15        nome = entrada.nextLine();
16        System.out.println("Informe o valor da nota 1: ");
17        nota1 = entrada.nextInt();
18        System.out.println("Informe o valor da nota 2: ");
19        nota2 = entrada.nextInt();
20        System.out.println("Informe o valor da nota 3: ");
21        nota3 = entrada.nextInt();
22
23        media = (nota1+nota2+nota3)/3;
24
25        System.out.println("Olá "+nome+" sua média é: "+media);
26    }
27
28 }
```

## Métodos da classe Scanner

Método	Finalidade
next()	Aguarda uma entrada em formato String
nextInt()	Aguarda uma entrada em formato Inteiro
nextByte()	Aguarda uma entrada em formato Inteiro
nextLong()	Aguarda uma entrada em formato Inteiro Longo
nextFloat()	Aguarda uma entrada em formato Número Fracionário
nextDouble()	Aguarda uma entrada em formato Número Fracionário

```
package Aula1;
```

```
import java.util.Scanner;
```

```
/**
```

```
 * @author Fernando
```

```
 */
```

```
public class Exemplo1ClasseScanner {
```

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

```
        Scanner input = new Scanner(System.in);
```

```
        int valorA, valorB, valorC;
```

```
        double media;
```

```
        System.out.println("Valor A: ");
```

```
        valorA = input.nextInt();
```

```
        System.out.println("Valor B: ");
```

```
        valorB = input.nextInt();
```

```
        System.out.println("Valor C: ");
```

```
        valorC = input.nextInt();
```

```
        media = (valorA + valorB + valorC) / 3;
```

```
        System.out.println("Média: " + media);
```

```
    }
```

```
}
```



# Atividades

## Entrada, Processamento e Saída



# Atividades

1) Escreva uma classe para ler o nome de um aluno e calcular a média final considerando que a mesma será composta por 4 notas: n1, n2, n3 e n4. Ao final, o programa deve imprimir o nome do aluno e a média final calculada.

- 1) Identifique os dados de entrada
- 2) Qual será o processamento realizado
- 3) Quais dados são de saída

# Atividades

2) Na empresa onde trabalhamos há uma tabela com o quanto foi gasto em cada mês. Para fechar o balanço do primeiro trimestre, precisamos somar o gasto total, sabendo que em Janeiro, foram gastos R\$ 15000, em Fevereiro, R\$ 23000 e em Março, R\$ 17000. Faça um programa que calcule e imprima o gasto total no trimestre.

Siga os passos abaixo:

- 1) Crie uma classe chamada `BalancoTrimestral` com um bloco main (exemplo anterior)
- 2) Dentro do main, declare uma variável inteira chamada `gastosJaneiro` e inicialize-a com R\$ 15000
- 3) Crie também as variáveis `gastosFevereiro` e `gastosMarco`, inicializando-as com R\$ 23000 e R\$ 17000, respectivamente (utilize uma linha para cada declaração)
- 4) Crie uma variável chamada `gastosTrimestre` e inicialize-a com a soma das outras 3 variáveis:  

```
int gastosTrimestre = gastosJaneiro + gastosFevereiro + gastosMarco
```
- 5) Imprima a variável `gastosTrimestre`
- 6) Adicione um código (sem alterar as linhas que já existem) para imprimir a média mensal de gastos, criando uma variável `mediaMensal` junto com uma mensagem. Para isso, concatene a String com valor utilizando a expressão **“Valor da média mensal = “ + media mensal**

• • • • • +

• • • • •

• + •

+ •

|

+

# Obrigado e até a próxima aula!

FIAP

Copyright © 2022 | Professor Fernando Luiz de Almeida

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento, é expressamente proibido sem consentimento formal, por escrito, do professor/autor.

• • •

• + • •