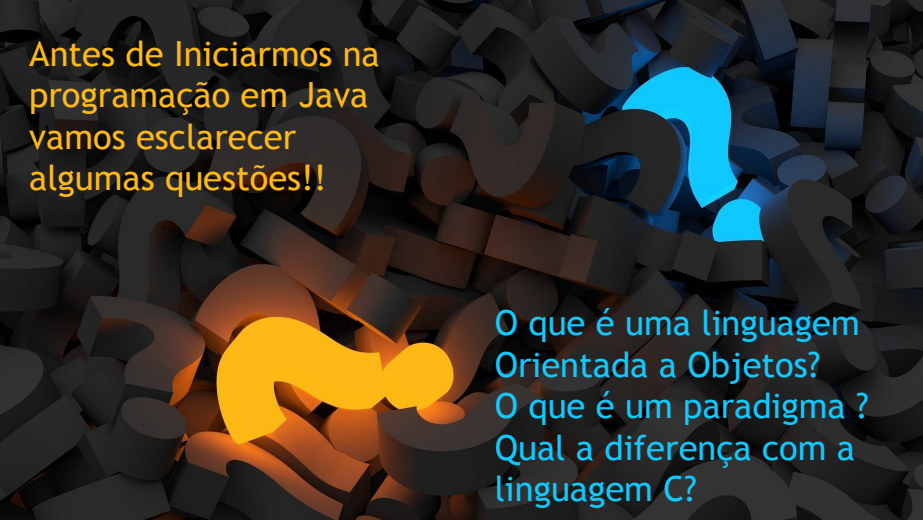


# **Programação Orientada a Objetos**

## **Introdução ao Java**

Renato Moraes Silva  
renato.silva@facens.br

Antes de Iniciarmos na  
programação em Java  
vamos esclarecer  
algumas questões!!



O que é uma linguagem  
Orientada a Objetos?  
O que é um paradigma ?  
Qual a diferença com a  
linguagem C?

 Uma das principais linguagens de programação atualmente

 Baseada nas seguintes linguagens

 C e C++

 Suporte a programação

 “Estruturada”

 Baseada em Componentes

 Orientada a Objetos

🐼 Como Java é uma linguagem orientada a objeto, todo o código de um programa deve estar definido dentro de uma classe

🐼 Especificamente deve estar definido dentro de um método especial da classe chamado de **main**

```
public class PrimeiroPrograma
{
    public static void main(String[] args)
    {
        // Aqui serão digitados os passos do algoritmo
        /*
        instrução1;
        instrução2;
        instrução3;
        instrução4;
        */
    }
}
```

 A “Estrutura”:

```
public class PrimeiroPrograma
{
    ...
}
```

representa o programa propriamente dito. É através desta estrutura que são definidas as variáveis e as instruções do programa

O par { } marca sempre **início** e **fim**. Neste caso **marcam** onde **inicia** e **termina** o **programa**.



O método **Main** contém as instruções que serão executadas

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



Como Fazer?

1. Obter salário
2. Aumentar salario (sal)  
     $Sal = sal + (sal * 30/100)$
3. Apresentar sal

 Utilizada para escrever informações no monitor do computador

 O comando possui a seguinte sintaxe

`System.out.println(<Texto ou expressão>;`

Comando de Escrita

Informação a ser escrita  
(sempre entre parênteses)

Todo o  
comando em  
Java termina  
em com um ;

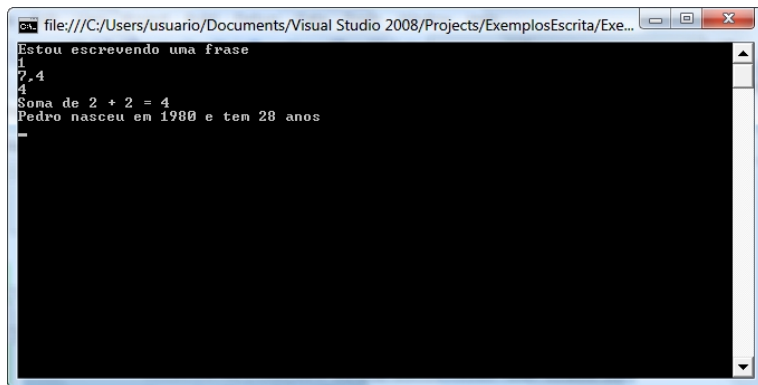
Objeto Responsável pela  
Interação com os dispositivos de Entrada e Saída



```
1 System.out.println("Estou escrevendo uma frase");
2 System.out.println(1);
3 System.out.println(7.4);
4 System.out.println(2+2);
5 System.out.println ("Soma de 2 + 2 =" + (2+2));
6 System.out.println ("Pedro nasceu em" + 1980 + " e tem
    " + (2008-1980) + "anos" );
```

# Colocando as instruções em um programa

```
1 public class ExemplosEscrita
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Estou escrevendo uma frase");
6         System.out.println(1);
7         System.out.println(7.4);
8         System.out.println(2+2);
9         System.out.println ("Soma de 2 + 2 =" + (2+2));
10        System.out.println ("Pedro nasceu em" + 1980 + " e
            tem" + (2008-1980) + "anos" );
11    }
12 }
```



A screenshot of a Windows command prompt window. The title bar shows the file path: file:///C:/Users/usuario/Documents/Visual Studio 2008/Projects/ExemplosEscrita/Exe... The window has standard Windows window controls (minimize, maximize, close). The command prompt area is black with white text. The output of the program is as follows:

```
Estou escrevendo uma frase
1
7,4
4
Soma de 2 + 2 = 4
Pedro nasceu em 1980 e tem 28 anos
-
```

Tipo	Tamanho em bytes	Descrição
int	4	Valores inteiros entre -2147483648 and 2147483647
float	4	Números reais (ponto flutuante). Valores entre $\pm 1.5 \times 10^{-45}$ até aproximadamente $\pm 3.4 \times 10^{38}$ com 7 dígitos significativos
double	8	Números reais com dupla precisão. Armazena valores no intervalo $\pm 5.0 \times 10^{-324}$ até aproximadamente $\pm 1.8 \times 10^{308}$ com até 16 dígitos significativos
char	2	Caracteres
String	Objeto	Armazena uma sequência de caracteres e símbolos especiais (\n e \t)
boolean	1	Armazena valores true ou false apenas

 Em Java cria-se uma variável (declaração) da seguinte forma:

```
tipo nomeVariavel;
```


 Onde

 **nomeVariavel** é um nome significativo que representa uma posição de memória

 **tipo** identifica o tipo da informação armazenada naquela posição de memória.

 Ex: int, double, float, char ou String

Declaração	Significado
<b>String nome;</b>	Cria uma variável chamada nome que é capaz de armazenar uma sequência de caracteres
<b>int idade;</b>	Cria uma variável chamada idade que é capaz de armazenar números inteiros
<b>double salario;</b>	Cria uma variável chamada salário que é capaz de armazenar números reais
<b>float altura;</b>	Idêntica a anterior só que com a capacidade de representar uma menor quantidade de números
<b>char opcao;</b>	Cria uma variável chamada opcao que é capaz de armazenar um caractere
<b>boolean resultado;</b>	Cria uma variável lógica que é capaz de armazenar um dos seguintes valores: <b>true</b> ou <b>false</b>

-  Em nossa disciplina recomenda-se que a variável seja declarada dentro do método **main**

```
1 class ExemploDeclVariaveis
2 {
3     static void main(String[] args)
4     {
5         //declaracao de variaveis
6         String nome;
7         int idade;
8         double salario;
9         float altura;
10        char opcao;
11        boolean resultado;
12    }
13 }
```

- 👤 Somente criar uma variável não acrescenta mais funcionalidades ao nossos programas
- 👤 Para que estas tenham alguma funcionalidade é necessário permitir que o usuário digite valores e estes sejam armazenados nas variáveis
- 👤 Fazemos isto através da seguinte instrução

```
Scanner entrada = new Scanner(System.in);  
nomeVariavel = entrada.nextTipo();
```



Lê um valor digitado pelo usuário no teclado e o armazena na variável indicada



```
1 class ExemploDeclVariaveis{
2
3     public static void main(String[] args)
4     {
5         //declaracao de variaveis
6         string nome;
7         int idade;
8         double salario;
9         float altura;
10        char opcao;
11        bool resultado;
12        Scanner entrada = new Scanner(System.in)
13
14        System.out.println("Digite seu nome");
15        nome = entrada.next();
```

## Exemplo 01

```
16     System.out.println("Nome Lido =" + nome);  
17 }  
18 }
```

🐼 Quando desejamos ler um valor do teclado que seja diferente de uma string devemos

🔥 Converter a string no tipo de dado desejado

📺 para isto utilizamos as variações do método next

📺 Exemplo:



- `Idade = entrada.nextInt(); // lê um inteiro`
- `salario = entrada.nextFloat(); // lê um número real`


```
1 class ExemploDeclVariaveis
2 {
3     static void Main(string[] args)
4     {
5         //declaração de variaveis
6         string nome;
7         int idade;
8         double salario;
9         float altura;
10        char opcao;
11        boolean resultado;
12
13        Scanner entrada = new Scanner(System.in);
14
15        System.out.println("Digite seu nome");
```

```
16     nome = entrada.next();
17     System.out.println("Nome Lido" + nome);
18
19     System.out.println("Digite sua idade");
20     idade = entrada.nextInt();
21     System.out.println("Idade Lida =" + idade);
22 }
23 }
```



Instrução	Significado	Exemplo
<code>nextInt()</code>	Lê um dado do teclado e o converte para um número inteiro	<code>idade = entrada.nextInt();</code>
<code>nextFloat()</code> <code>nextDouble()</code>	Lê um dado do teclado e o converte para um número real	<code>salario = entrada.nextDouble();</code> <code>altura = entrada.nextFloat();</code>
<code>nextLine();</code>	Lê uma string do teclado	<code>texto = entrada.nextLine();</code>
<code>nextBoolean();</code>	Lê um dado do teclado e o converte para um valor lógico	<code>resultado = entrada.nextBoolean();</code>

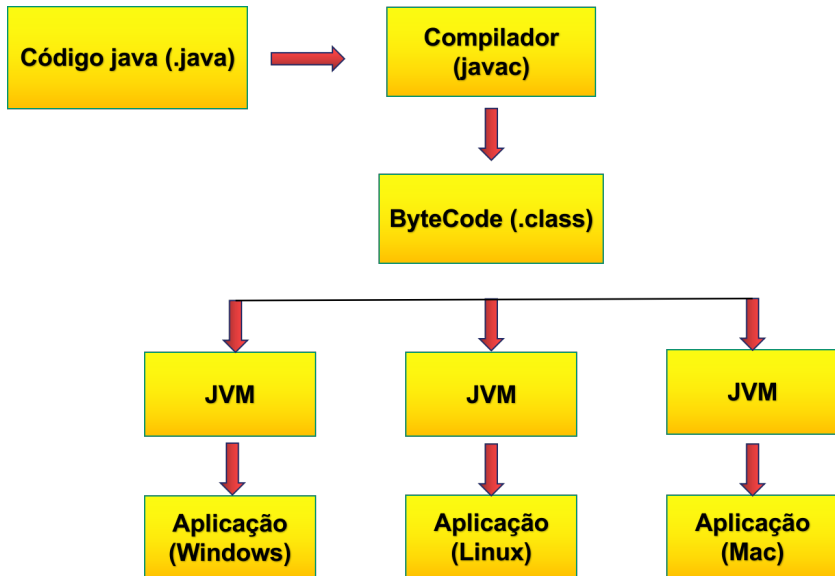
## Diferença entre next e nextLine:

-  next: lê a String até o primeiro espaço; no fim, posiciona o cursor na mesma linha
-  nextLine: lê a String até encontrar a primeira quebra de linha; no fim, posiciona o cursor na próxima linha

 Quando o método nextLine é usado logo após nextInt ou nextFloat, pode acontecer de ele não ler o texto que for digitado

### Soluções

-  use dois objetos da classe Scanner
-  use um método nextLine imediatamente após o nextInt ou nextFloat para consumir o '/n' que sobra







**JDK:** Kit de Desenvolvimento Java



responsável por compilar o código-fonte (.java) em ByteCode (.class)



**JRE:** ambiente de execução do Java



fornece as bibliotecas padrões do Java para o JDK compilar o seu código e para a JVM executar o seu programa.

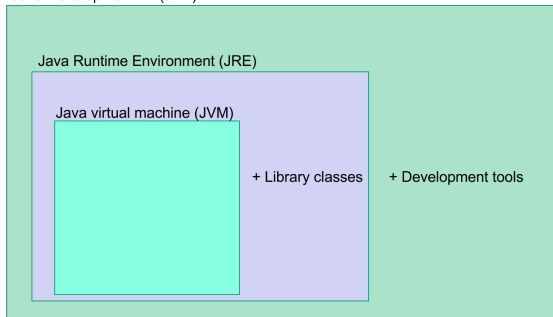


**JVM:** máquina virtual do Java.






responsável por executar o bytecode (.class)








Java Development Kit (JDK)



## Compilação






-  `javac <arquivoFonte.java>`
  -  adicionar a extensão `.java`
  -  Exemplo: `java HelloWorld.java`

## Execução

-  `java <arquivoFonte>`
  -  não adicionar a extensão `.java`
  -  Exemplo: `java HelloWorld`
-  Para passar parâmetros use: `java <arquivoFonte> <param1> <param2>`
  -  Exemplo: `java HelloWorld Fulano`
  -  os parâmetros são recebidos pela variável **args** do método main:  
**`public static void main(String[] args)`**
  -  Essa variável é um vetor. Portanto, o parâmetro deve ser acessado através de seu respectivo índice: **`args[i]`**

- 👤 String é uma classe
- 👤 Valores tipo strings são instâncias desta classe
- 👤 Tipo de classe especial onde instância pode ser declarada como tipos simples:
  - 🔥 Ex: String nome = "Fulano";
- 👤 Comparação
  - 🔥 <string1>.equals(<string2>)
  - 🔥 <string1>.equalsIgnoreCase(<string2>)


Úteis quando a conversão não é automática

-  `Short.parseShort(<argumento>)`
-  `Integer.parseInt(<argumento>)`
-  `Long.parseLong(<argumento>)`
-  `Float.parseFloat(<argumento>)`
-  `Double.parseDouble(<argumento>)`

```
1 double x = 50;  
2 float y;  
3 y = (float)x;
```

```
1 if(<condição>)  
2   <bloco>
```

```
1 if (<condição>)  
2   <bloco>  
3 else  
4   <bloco>
```

 (<condição>)


 Parênteses são obrigatórios

 <bloco>

 Apenas uma instrução


 terminada por ponto-e-vírgula

 Mais de uma instrução

 delimitada por chaves { }

 cada instrução dentro das chaves é encerrada por ponto-e-vírgula

```
1 switch ( <expressão> )  
2 {  
3     case <constante> : <instruções>  
4                         break;  
5     ...  
6     case <constante> : <instruções>  
7                         break;  
8  
9     default : <instruções>  
10 }
```

-  Desvia para o “case” cujo valor da <constante> é igual ao valor da <expressão>; senão desvia para o default.



```
1 int day = 4;
2 switch (day) {
3     case 1:
4         System.out.println("Monday");
5         break;
6     case 2:
7         System.out.println("Tuesday");
8         break;
9     case 3:
10        System.out.println("Wednesday");
11        break;
12    case 4:
13        System.out.println("Thursday");
14        break;
15    case 5:
```

```
16     System.out.println("Friday");
17     break;
18 case 6:
19     System.out.println("Saturday");
20     break;
21 case 7:
22     System.out.println("Sunday");
23     break;
24 default:
25     System.out.println("Invalid day");
26 }
```

```
1 while (<condição>)  
2     <bloco>
```

 testa condição no início

```
1 do  
2     <bloco>  
3 while (<condição>);
```

 testa condição no final

```
1 int i = 0;
2 while (i < 5) {
3     System.out.println(i);
4     i++;
5 }
```

```
1 int count = 1;
2 do {
3     System.out.println("Count is: " + count);
4     count++;
5 } while (count < 11);
```

```
1 for(<inicialização>;<condição>;<incremento>)  
2   <bloco>
```



<inicialização>



executada antes de entrar no for



usualmente inicializa variável de controle



<condição>



testada na entrada e a cada ciclo completo



se verdadeira prossegue a repetição



<incremento>




executada a cada ciclo completo




usualmente incrementa variável de controle


```
1 for(int i=1;i<=10;i++){  
2     System.out.println(i);  
3 }
```

## Declaração

 `<tipo>[] <declaração_1 >, ..., <declaração_n>;`


 `<tipo> <declaração_1>[], ..., <declaração_n>[];`


 `<declaração>`

 Sintaxe: `<nome> = <inicialização>`


 Chaves são usadas para inicializar cada dimensão


 Ex.: `int primos[] = {1, 2, 3, 5, 7};`

 Quando a inicialização não é inline, o vetor ou matriz precisa ser instanciado


 `<nome> = new <tipo>[<tamanho>]`


 Ex.:


 `int primos[];`

 `primos = new int[5];`

 Alguns slides dessa apresentação foram criados pelos professores André Santanchè e Luciano Freire

 Links para aprender a instalar o JDK e o Eclipse:

 <https://www.youtube.com/watch?v=LG5cdQSSgKc>

 <https://www.youtube.com/watch?v=rAaxlyKs78U>