

# Aula II - Desenvolvimento de Aplicativo I

## Continuação da Estrutura do Código Java

### Classe Main (Principal)

### Método da classe main

### Operadores aritméticos

#### Fontes:

<https://www.devmedia.com.br/metodos/7348#:~:text=Um%20m%C3%A9todo%20em%20Java%20%C3%A9,definidos%20dentro%20de%20uma%20classe.>

<https://www.devmedia.com.br/trabalhando-com-metodos-em-java/25917>

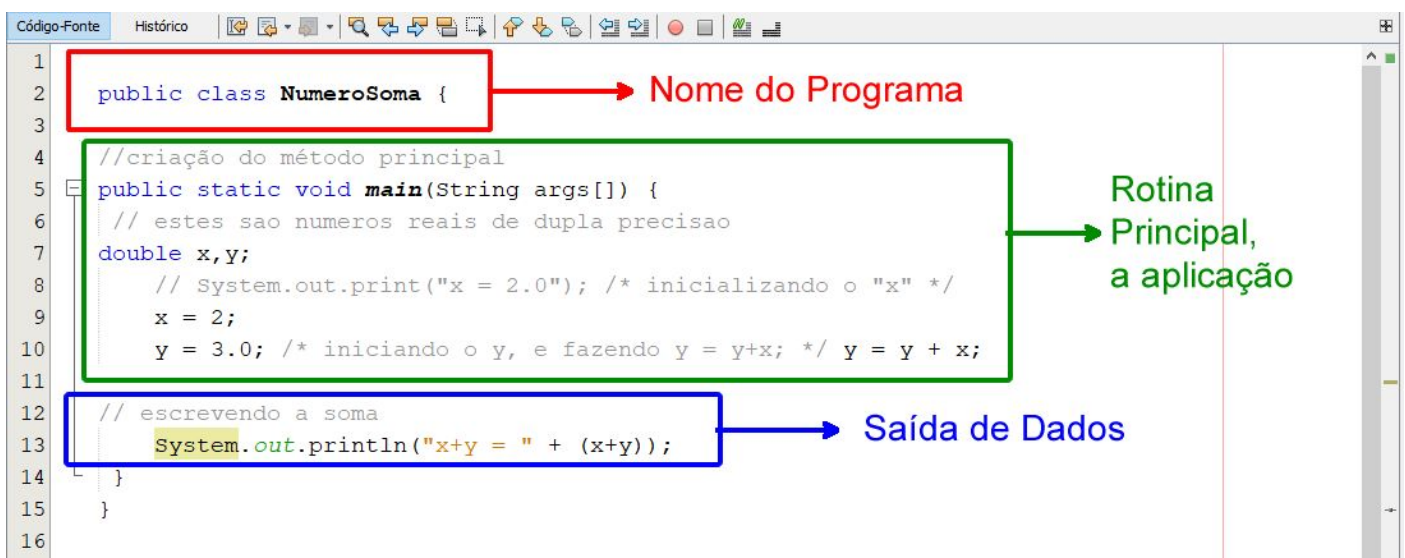
<https://www.devmedia.com.br/entendendo-a-estrutura-de-um-codigo-java/24622>

## Continuação da Estrutura do Código Java

### Estrutura do Programa Java

As primeiras coisas que devem ser abordadas para começar a desenvolver em qualquer linguagem são:

- Bloco do programa;
- Declarar variáveis;
- Sintaxes dos comandos;

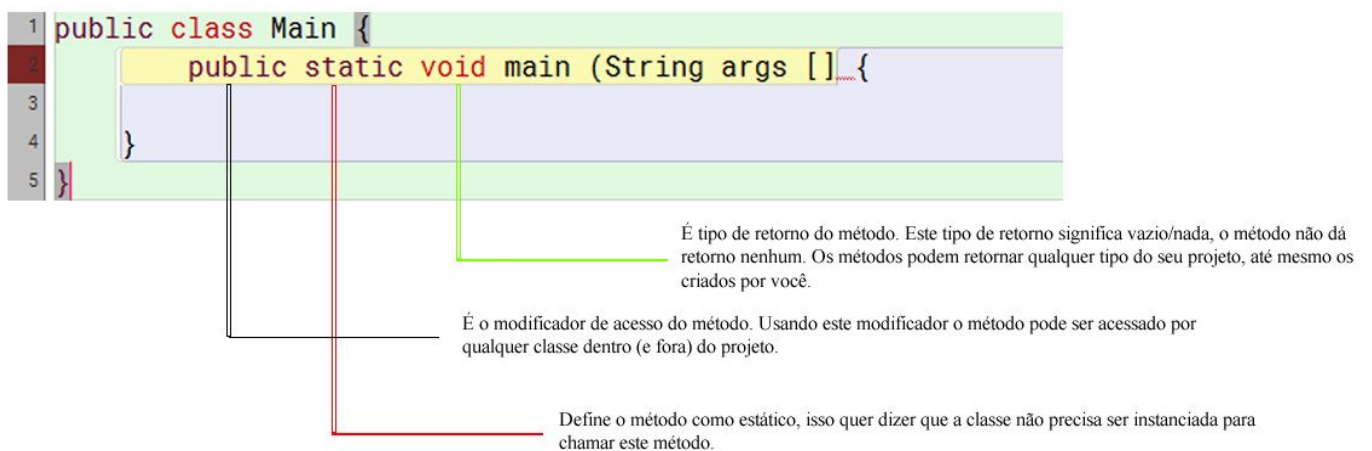


## Classe Main

Quando escrevemos o código fonte de um programa em Java e o compilamos, ele é transformado em **bytecode**, que nada mais é que uma linguagem intermediária interpretada por uma máquina virtual, a **JVM (Java Virtual Machine)**. É a JVM que faz a ponte entre seu código fonte e a máquina, convertendo os *bytecodes*. Alguns criticam esse reprocessamento porque a linguagem perderia desempenho se comparada com outras, como o C, em que compilamos o código diretamente para linguagem de máquina.

Um projeto em Java pode ser composto por **diversas classes**. Mas uma delas deve ser **responsável pela execução do programa, um ponto de partida**.

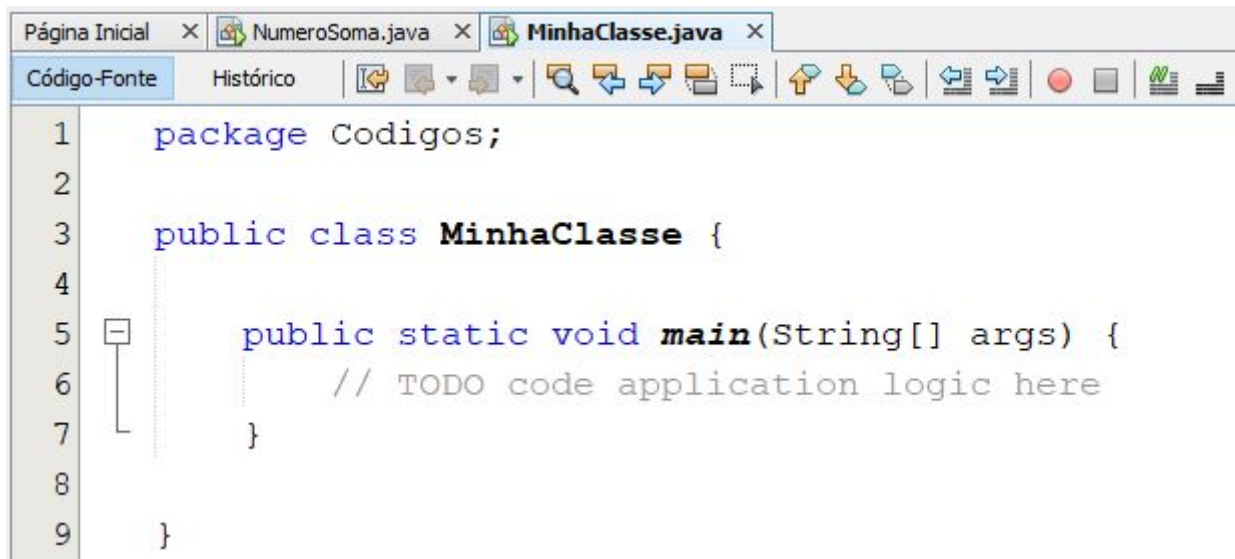
**Um início.** Normalmente chamamos essa classe de **Programa, Principal, Main, Teste, etc.** Essa classe não possui atributos, e normalmente possui apenas um método chamado de **main** (principal, em inglês). Nesta classe inserimos os comandos para o programa "funcionar". Esta é a classe que será "executada". E que nos retornará finalmente algo na tela.



**(String[] args)**

Toda a classe Principal possuirá uma linha com o seguinte comando:

**public static void main(String args[])** - Este comando especifica que a classe é estática, sem retorno e principal. Abaixo deste comando devemos escrever todos os passos que devem ser seguidos pelo programa. Define que o método deve receber como parâmetro um array de String (nomeado args). Nesse caso específico: este parâmetro serve para caso seu programa precise receber algum valor como argumento, isso é muito comum quando o programa é iniciado por outro programa ou pelo terminal (CMD, Shell, Bash, etc.).



```
1 package Codigos;
2
3 public class MinhaClasse {
4
5     public static void main(String[] args) {
6         // TODO code application logic here
7     }
8
9 }
```

## Anatomia do método main (imagem acima)

**public** = Do mesmo modo que um método comum, refere-se a visibilidade deste método. Quando dizemos que o método é de visibilidade “public”, estamos dizendo que este método poderá ser acessado por outras classes.

**static** = Nos garante que somente haverá uma, e não mais que uma, referência para nosso método main, ou seja, todas as instâncias da classe irão compartilhar a mesma cópia do método main.

**void** = Assim como um método comum, refere-se ao tipo de retorno que esse método terá. Nesse caso, como o tipo de retorno deve ser “void”, ou seja, “vazio”, esse método não retornará valor nenhum.

**(String[] args)** = Refere-se aos argumentos que serão passados para esse método, sendo obrigatório no caso do método main

**{}** = Assim como um método comum, as chaves indicam até onde certa classe ou método se estende. O código que queremos inserir neste método deverá ser escrito dentro do espaço das chaves.

## O que são métodos? (genérico)

Um método em Java é equivalente a uma função, subrotina ou procedimento em outras linguagens de programação. **Não existe em Java o conceito de métodos globais. Todos os métodos devem sempre ser definidos dentro de uma classe.**

## MÉTODOS DE SAÍDA DE DADOS (IMPRESSÃO NA TELA)

### Método System.out.println()

A instrução System.out.println(), gera uma saída de texto entre aspas duplas significando uma String, criando uma nova linha e posicionando o cursor na linha abaixo, o que é identificado pela terminação "ln".

Sempre temos a possibilidade de mandar mensagens ao usuário, essas mensagens podem conter somente texto, somente respostas, textos e respostas, texto, respostas e mais texto... Isso é possível através de um método de saída de dados, o método out, este método é da classe do sistema, **class System**, portanto o comando fica: **System.out.print("mensagem");**

Exemplos:

```
public class Main {  
    public static void main (String args[]) {  
        System.out.println("Aqui via o texto que aparece para o usuário!");  
    }  
}
```

```
public class Main {  
    public static void main (String args[]) {  
        System.out.println("Aqui via o texto que aparece para o usuário!");  
        System.out.println("Sempre quando eu quero colocar quebra de linha ....");  
        System.out.println("Eu coloco -ln- junto do print");  
    }  
}
```

Senão ....

```
public class Main {  
    public static void main (String args[]) {  
        System.out.print("Aqui vai o texto que aparece para o usuário!");  
        System.out.print("Posso deixar sem o -ln- ...");  
        System.out.print("Que fica tudo na mesma linha");  
    }  
}
```

# Operadores

Os operadores são utilizados para representar expressões de cálculo, comparação, condição e atribuição. Temos os seguintes tipos de operadores: de atribuição, aritméticos, relacionais e lógicos.

## Operadores Aritméticos Básicos

Em nossas classes teremos métodos que irão executar alguma operação matemática. Pensando nisso, precisamos saber quais operadores matemáticos podemos utilizar em nossos cálculos. Os símbolos que representam as operações matemáticas em Java são:

Símbolo matemática	Significado	Comando/símbolo em Java
+	Adição	+
-	Subtração	-
x ou .	Multiplicação	*
÷	Divisão	/
mod	Resto da divisão	%

São executados/calculados nesta ordem de precedência:

## Cálculos matemáticos

### Juros e Descontos

#### Juros / Valor com Juros (valor a prazo)

Para calcularmos o valor de juros, basta pegarmos o valor e multiplicá-lo pelo percentual de juros, porém no programa **não poderemos utilizar o símbolo “%”** para representar percentuais, pois “%” significa módulo. Teremos que convertê-lo para um número com vírgula, para isto, basta dividi-lo por 100.

#### Exemplo:

10% → 0,10 → Em Java 0.1

5% → 0,05 → Em Java 0.05

## Operadores de Atribuição

São utilizados para expressar o armazenamento de um valor em uma variável. Esse valor pode ser pré-definido (variante ou não) ou pode ser o resultado de um processamento.

Representação utilizando-se a notação para linguagem Java
=
Exemplo: nome = "Fulano de tal" resultado = a + 5 valor = 3