# Aula 02-Introdução ao Java

## Introdução

Olá, meus estudantes, sejam bem-vindos à nossa aula de Algoritmo e Programação em Java do Instituto da Oportunidade Social. Nessa aula, você vai conhecer um pouco fluxograma e conceitos de tipos de dados, operandos e comando de saída de dados.

## Mais sobre o Java

Como foi dito anteriormente o Java é uma linguagem de programação orientada a objetos, de alto nível e baseada em classes, mas ela também pode ser usada para ensinar programação estruturada, que é a proposta desse curso. A programação estruturada é um padrão ou paradigma de programação de engenharia de software, que têm ênfase em sequência, decisão e iteração (sub-rotinas, laços de repetição, condicionais e estruturas em bloco). Esse de programação é muito comum em linguagens como C e C++.

### Por que estudar Java?

O Java está entre as linguagens de programação mais populares e utilizadas no mundo atual. Hoje, em 2021, apenas o Python é mais utilizado do que o Java de acordo com o ranking de Top Programming Languages 2021 do IEEE Spectrum. A popularidade do Java pode ser vista pela quantidade de vagas disponíveis para profissionais que possuem esse conhecimento. Além disso, o Java possui uma grande comunidade ativa de projetos open-source, por exemplo a Apache Software Foundation mantem mais de 200 projetos de Java, que vão desde ferramentas de build a servidor como o Tomcat.

Outro ponto importante a se destacar é que o Java está presente em diversos projetos que já estão em produção em diversas empresas, sendo assim um desenvolvedor precisa entender como programar Java para criar funcionalidade ou dar manutenção nesses sistemas. E por fim, o Java é uma linguagem moderna que está em evolução e aderente aos novos paradigmas de programação.

### Por que estudar Java nesse curso?

O ensino de Java na forma de programação estruturada pode ser um grande passo para a familiarização com a linguagem e com estruturas mais simples de programação. No contexto de programação web, o Java está presente no backend de serviços de internet. E para o aluno que deseja continuar a sua formação como programador web, esse conteúdo pode ser o conhecimento fundamental para complementar os seus estudos.

|  |
| --- |
| Dica: programar não é um processo simples que você irá aprender em um curso ou poucas horas de estudo. A evolução em programação requer muitas horas de dedicação, enfrentar desafios e aprender a estudo por conta própria. Seguem duas indicações de vídeos do canal do Fábio Akita mais realistas sobre programação e aprendizado. Espero que eles sirvam de inspiração e motivação para que o aluno dedique muito para aprender de forma consistente a programar.  Vídeo 01: Programar NÃO é fácil - <https://www.youtube.com/watch?v=V7oUDL7E1g4>  Vídeo 02: Guia DEFINITIVO de Aprendendo a Aprender | A maior BRONCA da sua vida - <https://www.youtube.com/watch?v=oUPaJxk6TZ0> |

## Nosso primeiro programa em Java

Na aula passada, nós fizemos a nossa primeira classe (Figura 1). Ela foi o primeiro passo no estudo de Java estruturado, mas vamos aprender a função dessas instruções e assim entender o que esse pequeno programa faz.

|  |
| --- |
| public class OlaMundo {  *// método main inicia a execução do aplicativo Java* public static void main(String[] args)  {  System.*out*.println("Bem vindo ao IOS!");  } *// fim do método main* } *// fim da classe OlaMundo* |

Figura . Primeiro código em Java.

### Declarando uma classe

O Java é uma linguagem baseada em classes, portanto todo programa Java consiste em pelo menos uma classe que o programador define. Assim, a primeira instrução é a declaração de uma classe:

public class OlaMundo

A palavra-chave **class** introduz uma declaração de classe e é imediatamente seguida pelo nome da classe (OlaMundo).

|  |
| --- |
| Importante: Palavras-chave (às vezes chamadas de palavras reservadas) são reservadas para uso pelo Java e sempre escritas com todas as letras minúsculas. As palavras-chave do Java são:    O Java também contém as palavras reservadas true e false, que são literais boolean, e null, que é o literal que representa uma referência a nada. Assim como as palavras-chave, essas palavras reservadas não podem ser utilizadas como identificadores. |

Cada classe que definimos inicia com a palavra-chave public. Uma classe public deve ser inserida em um arquivo com um nome na forma NomeDaClasse.java, assim a classe OlaMundo é armazenada no arquivo OlaMundo.java. Public é um modificador de acesso e uma declaração com o modificador public pode ser acessada de qualquer lugar e por qualquer entidade que possa visualizar a classe a que ela pertence.

### Comentários

Comentários são anotações inseridas no código fonte com o objetivo de descrever alguma lógica, instrução ou um lembrete TO-DO. Por exemplo, você pode usar comentários para: lembrar algo importante quando o código foi desenvolvido, criar secções para organização e ainda para criar um cabeçalho do código. Comentários são ignorados pelo compilador na verificação da sintaxe do código. Os comentários mais comuns para programadores iniciantes em Java são:

* Comentário de linha, que é iniciado por //

*// método main inicia a execução do aplicativo Java*

* Comentário de bloco, que é iniciado por /\* e finalizado por \*/

|  |
| --- |
| /\* Esse é um comentário tradicional. Ele  pode ser dividido em várias linhas \*/ |

Para programadores iniciantes, é interessante incluir um comentário de linha depois de um fechamento de chave que termina uma declaração de método e após uma chave de fechamento que termina uma declaração de classe. Isso ajuda iniciantes a visualizarem o início de fim do bloco de comandos de um método ou classe.

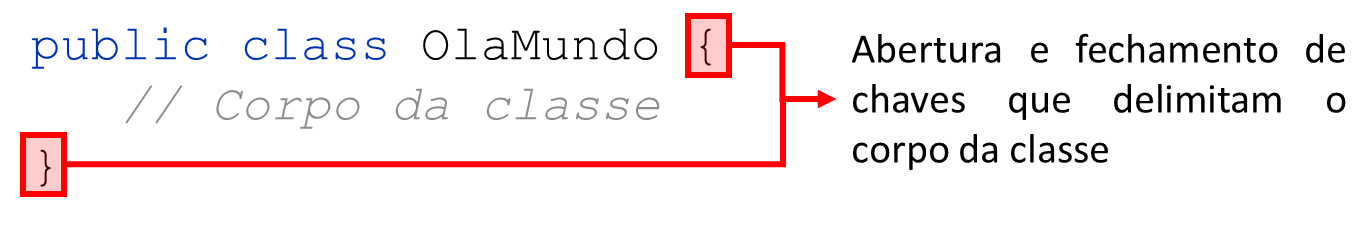
|  |
| --- |
| } *// fim do método main*  } *// fim da classe OlaMundo* |

O Java fornece comentários de um terceiro tipo: comentários Javadoc. Esses são delimitados por //\*\* e \*/. Os comentários no estilo Javadoc permitem-lhe incorporar a documentação do programa diretamente aos seus programas. O programa utilitário javadoc (parte do JDK) lê comentários Javadoc e os usa para preparar a documentação do programa no formato HTML.

|  |
| --- |
| Dica: Algumas organizações exigem que todo programa comece com um comentário que informa o objetivo e o autor dele, a data e a hora em que foi modificado pela última vez. |

### Corpo da classe

O corpo de uma classe, que pode também ser chamado de bloco de comandos da classe, é definido pela abertura e fechamento de chaves logo após a declaração da classe. Tudo que está dentro dessas chaves faz parte do código da classe.



|  |
| --- |
| Dica: faça do recuo de parágrafo para dar legibilidade a seu código e ficar bem claro a hierarquia de níveis do programa. Muitos programadores preferem dois ou quatro espaços. Qualquer que seja o estilo que você escolher, utilize-o de modo consistente. |

### Declarando um método

Um método em Java é equivalente a uma função, sub-rotina ou procedimento em outras linguagens de programação.

A instrução

public static void main(String[] args)

é o ponto de partida de cada aplicativo Java. Os parênteses depois do identificador **main** indicam que ele é um bloco de construção do programa chamado método. Declarações de classe Java normalmente contêm um ou mais métodos. Para um aplicativo Java, um dos métodos deve ser chamado **main**. Desse modo, o comando

System.*out*.println("Bem vindo ao IOS!");

instrui o computador para imprimir na tela a string Bem vindo ao IOS!, que foi passado entre aspas duplas. Uma string é uma sequência de caracteres e também conhecida como string de caracteres ou string literal. Os caracteres de espaço em branco em strings não são ignorados pelo compilador. As strings não podem distribuir várias linhas de código. Toda instrução em Java deve terminar com ponto-e-vírgula.

### Comando de saída

A saída de um programa é o resultado do seu processamento, pode ser armazenando em uma variável, um banco de dados ou exibição na tela. Em Java, System.out.println() é uma instrução para imprimir o argumento passado dentro dos parênteses. O método println() exibe o resultado na saída padrão (monitor) e realiza a quebra de linha, isto é, o curso vai para a próxima linha da tela. **System** é o nome de uma classe em Java e **out** é uma instância dessa classe.

### Modificando nosso primeiro programa

Vamos utilizar o método print() para imprimir uma string e sem quebra de linha. Você pode abrir o projeto anterior e modificar o código para ficar igual ao mostrado na Figura 2

|  |
| --- |
| public class OlaMundo {  *// método main inicia a execução do aplicativo Java* public static void main(String[] args)  {  System.*out*.print("Bem vindo ");  System.*out*.println("ao IOS!");  } *// fim do método main* } *// fim da classe OlaMundo* |

Figura . Código Java modificado.

Com o seu código atualizado e salvo, você pode compilar e executar o código acessando o botão Run e ver o resultado impresso no console. A Figura 3 mostra que o resultado é o mesmo do código anterior. Isso porque o método print() não provoca a quebra de linha e também tomamos o cuidado para colocar um espaço no final da primeira string para não apresentar o resultado todo junto (linha 5 do código mostrado na Figura 3).

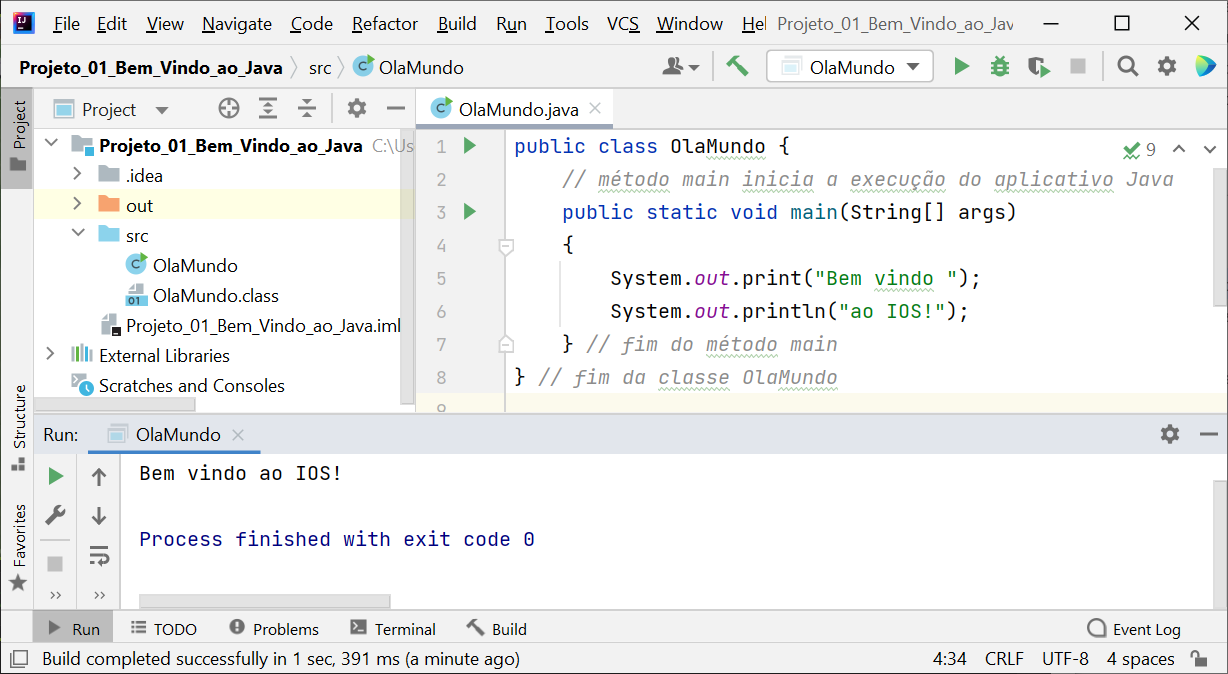


Figura . Resultado do código modificado compilado.

Se usarmos o método println() nas duas instruções de saída de dados como mostrado na Figura 4a, o resultado é as string impressas em linhas distintas (Figura 4b).

|  |  |
| --- | --- |
| public class OlaMundo {  *// método main inicia a execução do aplicativo* public static void main(String[] args)  {  System.*out*.println("Bem vindo");  System.*out*.println("ao IOS!");  } *// fim do método main* } *// fim da classe OlaMundo* |  |

Figura . (a) método println() nas duas instruções de saída de dados e (b) resultado da compilação e execução do código.

## Representações de Algoritmos (Descrição narrativa, Fluxograma e Pseudocódigo)

Existem várias formas de representar um algoritmo, entre elas podemos citar: descrição narrativa, fluxograma e algoritmo estruturado. A descrição narrativa é usar uma linguagem natural (ex. português) para escrever um enunciado e os passos a serem seguidos para a resolução do problema. A Figura 4 mostra a descrição narrativa de um programa para receber dois números, realizar a multiplicação desses números e exibir o resultado na tela.

|  |
| --- |
| Multiplicação de dois números:  Passo 1: Receber os dois números que serão multiplicados  Passo 2: Multiplicar os números  Passo 3: Mostrar o resultado obtido na multiplicação |

Figura . Descrição narrativa do algoritmo.

O Fluxograma é um tipo de diagrama, que representa esquematicamente um processo ou um algoritmo. Muitas vezes, ele construído através de gráficos que ilustram de forma descomplicada a transição de informações entre os elementos que o compõem. O fluxograma utiliza alguns símbolos padrões que possuem um significado dentro do diagrama. A Figura 3 mostra os símbolos mais comuns de um fluxograma para descrever um algoritmo em programação.

|  |  |
| --- | --- |
| A picture containing diagram  Description automatically generated | Graphical user interface  Description automatically generated with medium confidence |

Figura . Símbolos comuns do fluxograma.

|  |  |
| --- | --- |
| Voltando ao nosso exemplo de multiplicação de dois números, a mostra a representação em fluxograma do programa.  A Figura 4 mostra o fluxograma para o programa de multiplicação de dois números. Inicialmente, temos a entrada dos dois valores, em seguida o processamento da multiplicação e no final a exibição na tela. | Figura . Fluxograma do algoritmo de multiplicação de dois números. |

O algoritmo estruturado, também conhecido como pseudocódigo, é uma maneira intermediária entre a linguagem natural, no nosso caso o Português e uma linguagem de programação para representar um algoritmo computacional. Os pseudocódigos são estruturados de forma que qualquer programador que for codificá-lo no futuro, consiga reproduzi-lo na linguagem de programação escolhida.

Voltando ao nosso exemplo de multiplicação de dois números, a mostra a representação em algoritmo estruturado.

|  |  |  |
| --- | --- | --- |
| A Figura 5 mostra o pseudocódigo para o programa de multiplicação de dois números. Inicialmente, temos a entrada dos dois valores, em seguida o processamento da multiplicação e no final a exibição na tela. | |  | | --- | | ALGORITMO  DECLARE N1,N2,M NUMÉRICO  ESCREVA “Digite dois Digite dois números:”  LEIA N1, N2  M ← N1 \* N2  ESCREVA “Multiplicação = ”, M  FIM Do ALGORITMO |   Figura . Pseudocódigo do algoritmo de multiplicação de dois números. |

|  |
| --- |
| Importante: Algoritmo estruturado não é a mesma coisa do que portugol. O algoritmo estruturado é uma representação descrição em forma de pseudocódigo de um programa. Ele é importante na documentação de projetos open-source, em artigos científicos, etc. O portugol é uma linguagem usada para fins didáticos e serve para ensinar o aluno lógica de programação. Portugol não tem utilização na indústria. |

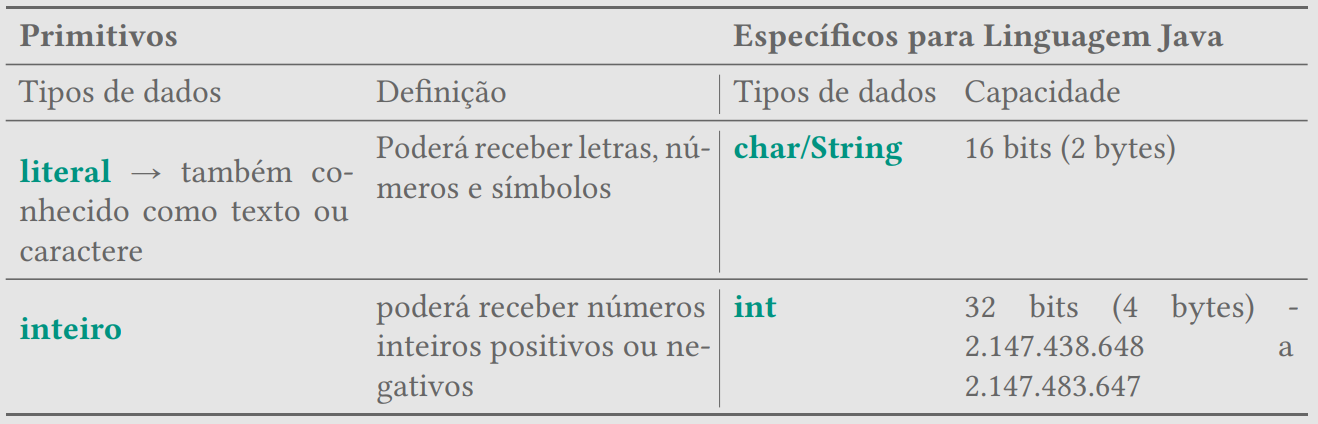
Tanto na descrição narrativa, fluxograma e pseudocódigo pode um ou mais caminhos para serem percorridos. A Figura 6 mostra a descrição narrativa, o fluxograma e o pseudocódigo para um programa de divisão de números.

|  |  |  |  |
| --- | --- | --- | --- |
| Descrição narrativa | | | |
| Divisão de dois números:  Passo 1: Receber os dois números que serão divididos  Passo 2: Verificar se o segundo número é igual a zero  Passo 3: Se segundo número não for igual a zero, a divisão poderá acontecer, caso contrário a divisão não pode ser realizada.  Passo 4: Exibir resultado da se possível, caso não exibir que não possível dividir | | | (a) |
| Fluxograma | | Pseudocódigo | |
|  | (b) | ALGORITMO  DECLARE N1, D2, D NUMÉRICO  ESCREVA “Digite dois números:”  LEIA N1, N2  SE N2 = 0 ENTÃO  ESCREVA “Impossível dividir.”  SENÃO  INÍCIO  D ← N1/N2  ESCREVA “Divisão = ”,D  FIM  FIM ALGORITMO | (c) |

Figura . (a) descrição narrativa, (b) fluxograma e (c) pseudocódigo do programa de dividir dois números.

## Constantes e tipos de dados

Constantes são valores fixos em um algoritmo e podem ser classificados como literal (caractere ou sequência de caracteres), numérico (inteiro ou real) ou lógico (verdadeiro ou falso). Já o tipo de dado indica a capacidade e o tipo de conteúdo de um valor ou constante. A Figura 10 exibe os principais tipos de dados do Java, bem como a sua descrição e capacidade:



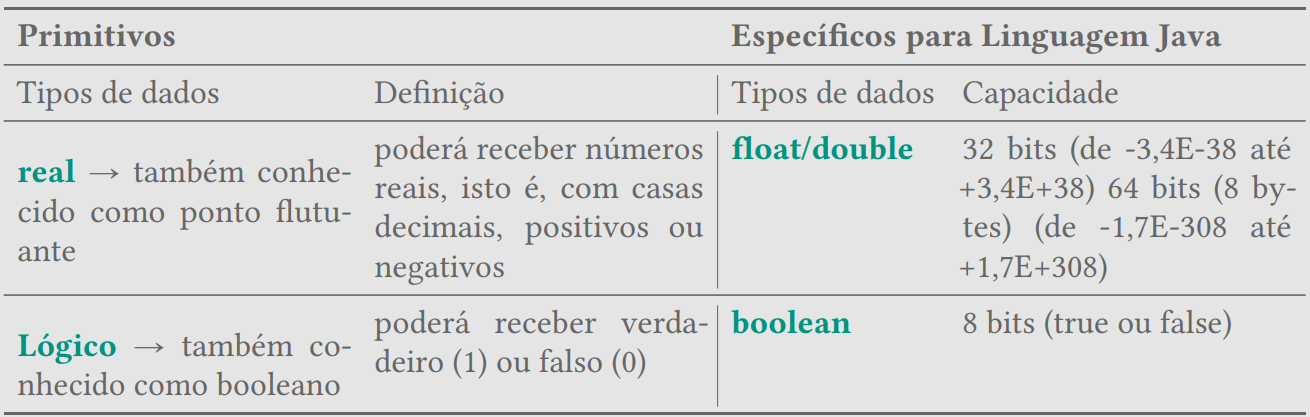


Figura . Tipos de dados no Java

Vamos ver alguns exemplos e revisar esse assunto:

|  |  |  |
| --- | --- | --- |
| false → lógico (boolean) | “O resultado é:” → literal (String) | “true” → literal (String) |
| 21 → inteiro (int) | 3.1415 → real (double) | ‘h’ → literal (char) |

## Operadores

Os operadores especificam uma avaliação/ação a ser executada em um ou mais operandos e podem ser aritméticos, relacionais ou lógicos. É através de operandos que os programas em Java executam cálculos aritméticos, lógicos e relacionais.

### Operadores aritméticos

Os operadores aritméticos são o conjunto de símbolos que representam as operações básicas da matemática. No Java, os operadores aritméticos são:

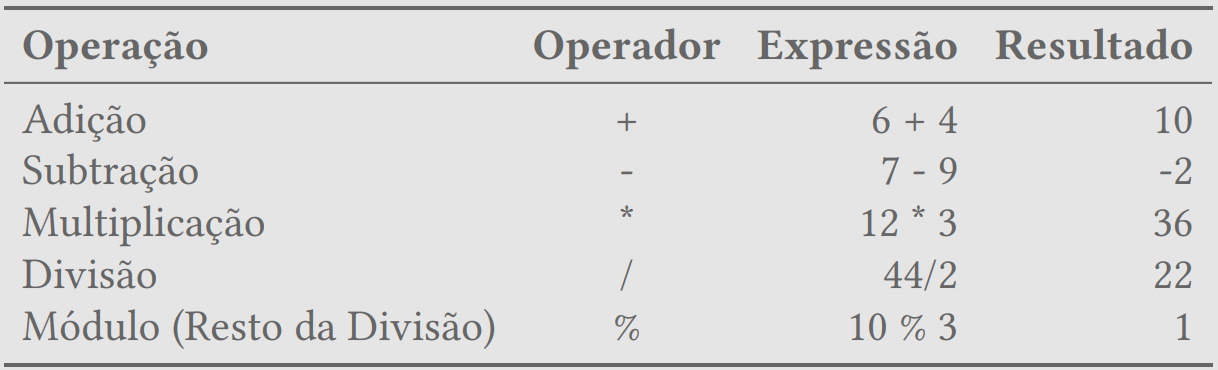


Figura . Operadores aritméticos.

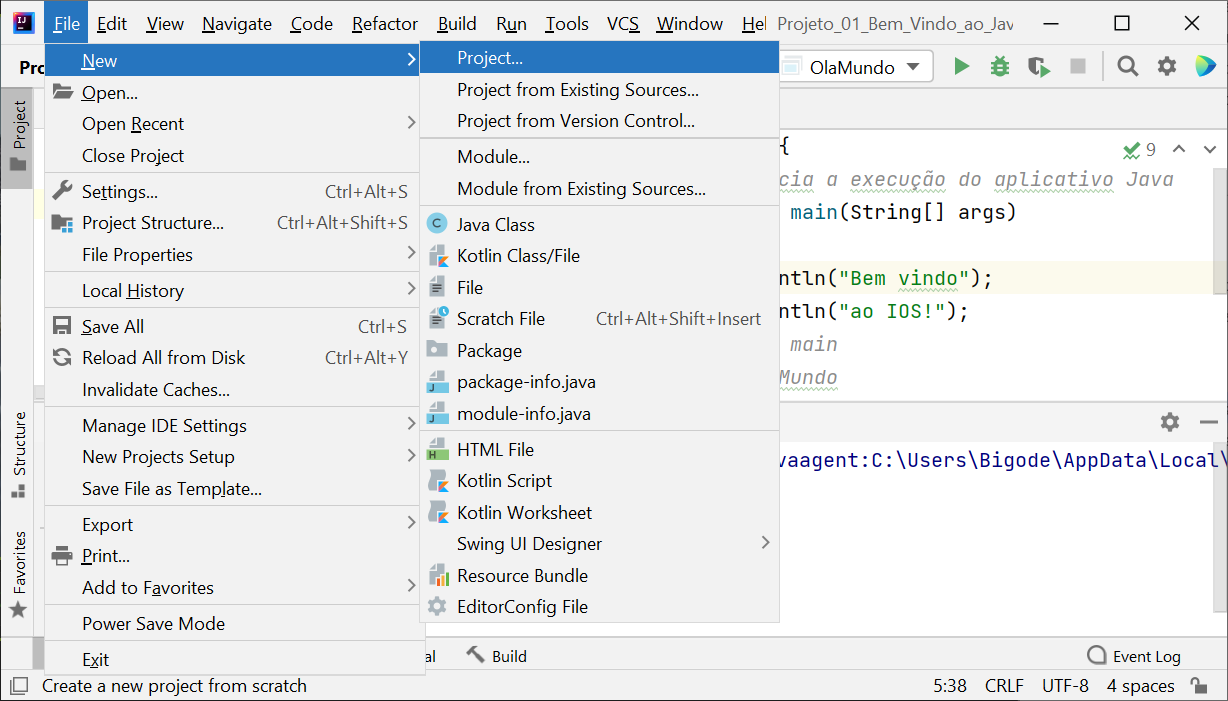
A ordem de precedência dos operandos aritméticos é:

|  |  |
| --- | --- |
|  | Desse modo a operação:  3 \* 5 + 2  O programa primeiro faz a multiplicação e com o resultado dessa multiplicação realiza a soma. Os parênteses têm a maior precedência, portanto a operação:  3 \* (5 + 2)  Primeiro é realizado a soma dentro dos parênteses e com o resultado dessa soma realiza a multiplicação. |

#### Vamos praticar

Siga os passos para escrever o programa utilizando operadores aritméticos:

1. Crie um novo projeto no InterlliJ IDEA. Para isso, acesse o Menu **File**, a opção **New** e depois clique na opção **Project**.



1. As janelas seguintes são as mesmas descritas na criação de um projeto, portanto se tiver alguma dúvida consulte o material didático. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Projeto\_02\_Oper\_Arit
2. No diretório **src**, crie um arquivo para a classe. Não se esqueça de dar um nome representativo para o projeto, por exemplo: OperArit e insira o código mostrado na Figura 12.

|  |  |
| --- | --- |
| 1 | public class OperArit {  *// método main inicia a execução do aplicativo* public static void main(String[] args)  {  System.*out*.println("Precedência: " + (10/2+3));  System.*out*.println("Alterando a Precedência: " + 10/(2+3));   System.*out*.println("Divisão inteira: " + 9/2);  System.*out*.println("Divisão real: " + 9/2.0);  System.*out*.println("Divisão real: " + 9.0/2);  System.*out*.println("Divisão real: " + 9.0/2.0);   System.*out*.println("Resto da divisão inteira: " + 9%2);  System.*out*.println("Resto da divisão inteira: " + 8%2);  System.*out*.println("Resto da divisão inteira: " + 9%5);  } *// fim do método main* } *// fim da classe OperArit* |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |
| 17 |

Figura . Código de operadores aritméticos.

1. O resultado do código pode ser visto ao compilar e executar o código, clicando com o botão direito na classe e selecionando a opção Run.

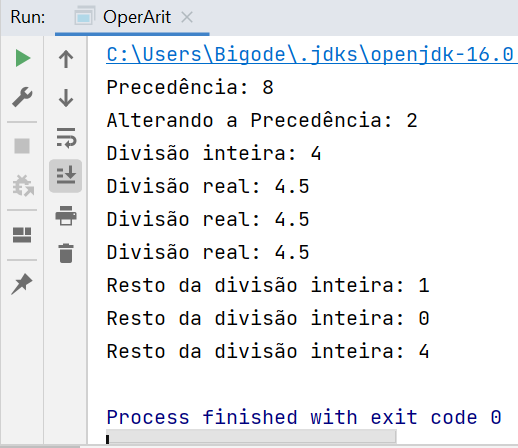


Figura . Resultado da execução do código.

A instrução na linha 5 da Figura 12 mostra que a divisão tem precedência em relação à soma, pois o resultado impresso na tela foi 8, ou seja, 10/2 = 5 + 3 = 8. Os parênteses do lado de fora da operação são necessários para que o Java entenda o símbolo + é uma soma e não uma concatenação de strings. Já a instrução na linha 6 da Figura 12 mostra como os parênteses podem alterar a precedência da operação.

|  |
| --- |
| Importante: em programação, concatenar é um termo usado para indicar a ação de unir duas ou mais strings. Por exemplo, considerando as strings "casa" e "mento" a concatenação da primeira com a segunda gera a string "casamento". |

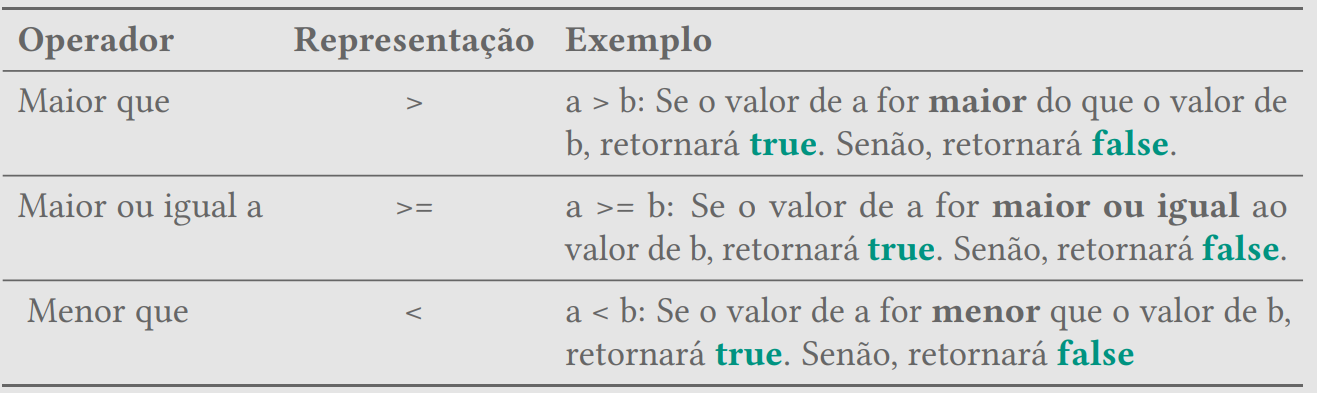
Na linha 8 da Figura 12, você pode ver a instrução que realiza a divisão inteira entre dois números inteiros. Divisão inteira é aquela que resulta um número inteiros se casas decimais. Caso você deseje que o resultado seja real, pelo menos um dos dois números deve ser escrito com o ponto decimal como mostrado nas instruções seguintes.

|  |
| --- |
| Importante: em programação, números reais, que são números com casas decimais, são representados com ponto decimal e não com vírgula como estamos acostumados no Brasil. |

Nas linhas 13, 14 e 15 da Figura 12, você pode observar como o operador de resto da divisão inteirao funciona. O operador % retorna o resto de uma divisão inteira, por exemplo, a divisão inteira de 5 por 3 tem quociente 1 e resto 2.

### Operadores relacionais

Os operadores relacionais são utilizados na realização de comparação entre valores do mesmo tipo primitivo, ou seja, eles podem ser representados por variáveis, constantes e até mesmo em expressões aritméticas. No Java, os operadores relacionais são:



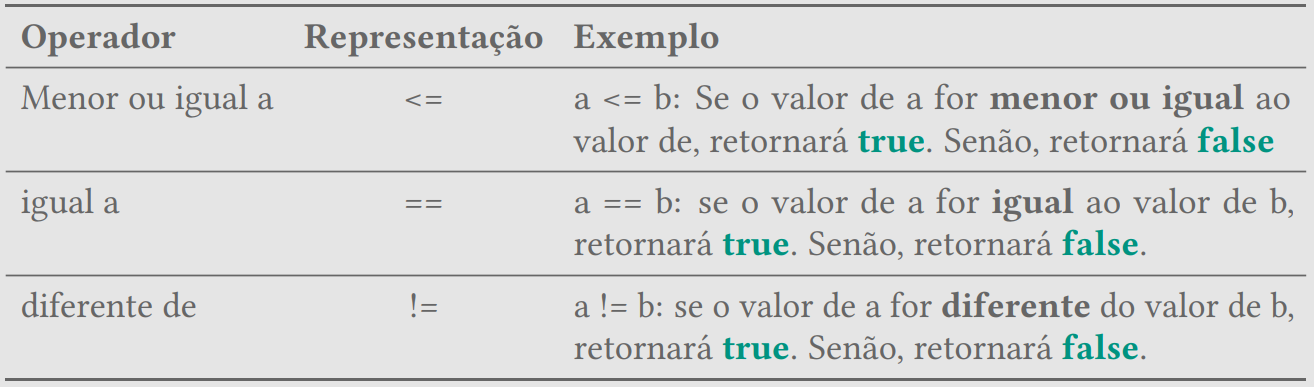


Figura . Operadores relacionais.

#### Vamos praticar

Siga os passos para escrever o programa utilizando operadores relacionais:

1. Crie um novo projeto no InterlliJ IDEA. Para isso, acesse o Menu **File**, a opção **New** e depois clique na opção **Project**.
2. As janelas seguintes são as mesmas descritas na criação de um projeto, portanto se tiver alguma dúvida consulte o material didático. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Projeto\_03\_Oper\_Relac
3. No diretório **src**, crie um arquivo para a classe. Não se esqueça de dar um nome representativo para o projeto, por ex.: OperRelac e insira o código mostrado na Figura 15.

|  |  |
| --- | --- |
| 1 | public class OperRelac {  *// método main inicia a execução do aplicativo* public static void main(String[] args)  {  System.*out*.println("Teste1: " + (3==2));  System.*out*.println("Teste2: " + (3!=2));  System.*out*.println("Teste3: " + (3>=2));  System.*out*.println("Teste4: " + (3<=2));  } *// fim do método main* } *// fim da classe OperRelac* |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |

Figura . Código de operadores relacionais.

1. O resultado do código pode ser visto ao compilar e executar o código, clicando com o botão direito na classe e selecionando a opção Run.

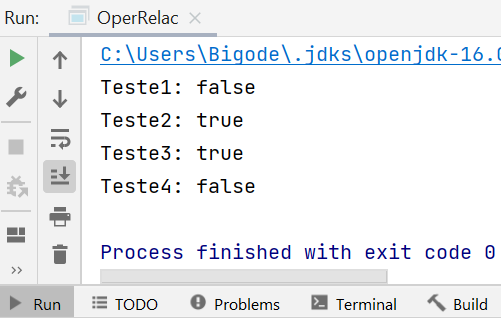


Figura . Resultado da execução do código.

O resultado de uma operação utilizando operadores relacionais é um tipo booleano: true ou false.

### Operadores Lógicos

Seguindo o raciocínio lógica booleana utilizamos os operadores lógicos para representar situações que não são representadas por operadores aritméticos. O resultado dessas expressões é sempre um valor lógico: true (verdadeiro) ou false (falso). Operadores lógicos são usados em programação para concatenar expressões que estabelecem uma relação de comparação entre valores.

Sempre utilizamos uma tabela verdade para representar todas as possíveis combinações das variáveis de entrada de uma função e os seus respectivos valores de saída.

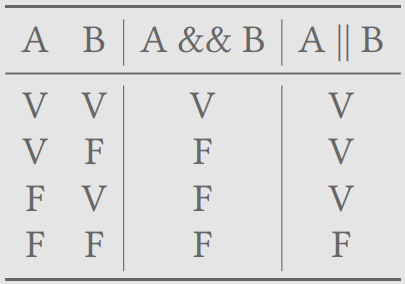
No Java, os operadores lógicos são:

* Operador E (&&)
  + somente resulta em verdadeiro (true), se todas as expressões condicionais forem true.
* Operador Ou (||)
  + se pelo menos uma expressão condicional for true, o resultado é verdadeiro (true).
* Operador Não (!)
  + se a expressão condicional for false, o resultado é true.
  + se a expressão condicional for true, o resultado é false.

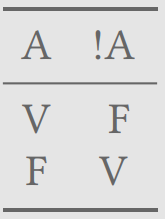
A tabela verdade para os operadores && e || é

* && → somente resulta em verdadeiro quando todas as sentenças avaliadas forem verdadeiras;
* || → somente resulta em falso quando todas as sentenças avaliadas forem falsas.

Exemplo:



O operador ! faz a negação de uma sentença. Exemplo:



#### Vamos praticar

Siga os passos para escrever o programa utilizando operadores relacionais:

1. Crie um novo projeto no InterlliJ IDEA. Para isso, acesse o Menu **File**, a opção **New** e depois clique na opção **Project**.
2. As janelas seguintes são as mesmas descritas na criação de um projeto, portanto se tiver alguma dúvida consulte o material didático. Não se esqueça de dar um nome representativo para o projeto, por exemplo: Projeto\_04\_Oper\_Log
3. No diretório **src**, crie um arquivo para a classe. Não se esqueça de dar um nome representativo para o projeto, por ex.: OperLog e insira o código mostrado na Figura 17.

|  |  |
| --- | --- |
| 1 | public class OperLog {  *// método main inicia a execução do aplicativo* public static void main(String[] args)  {  System.*out*.println("Teste1: " + (true && false));  System.*out*.println("Teste2: " + (true || false));  System.*out*.println("Teste3: " + (!true));  } *// fim do método main* } *// fim da classe OperLog* |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |

Figura . Código de operadores lógicos.

1. O resultado do código pode ser visto ao compilar e executar o código, clicando com o botão direito na classe e selecionando a opção Run.

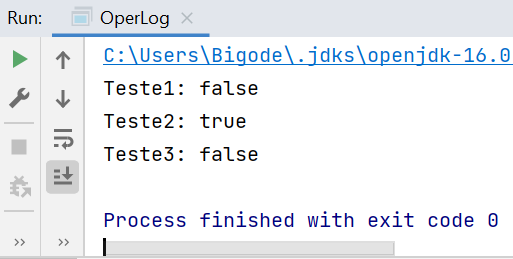


Figura . Resultado da execução do código.

O resultado de uma operação utilizando operadores lógicos é um tipo booleano: true ou false.

## Exercícios de revisão

1. Faça um programa em Java para imprimir em cada linha:

* Seu nome completo
* Sua cidade natal
* Seu hobby favorito

1. Faça um algoritmo para imprimir em cada linha:

* A sua idade
* A sua altura
* Seu status na aula:
  + O → online
  + F → offline

1. Faça um algoritmo para imprimir em cada linha o resultado lógico das seguintes expressões:
   1. ((120 - 30) = (3 ˆ 30))
   2. (não ((20 módulo 4)= 1) ou (9 != 9))
   3. ((5 módulo 2 2) > 3)
   4. (a = A)

Dica: Faça uma linha de cada vez. faça, teste, e avance para a próxima, não tente fazer todo código de uma única vez.

1. Faça um algoritmo para apresentar:
   1. o cálculo da média das seguintes notas 8.0, 7.5, 4.5 e 9.
2. Faça um algoritmo para apresentar:
   1. o cálculo da área de um quadrado de 350 metros de lado.