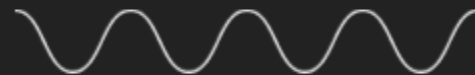


Aula 1 Java





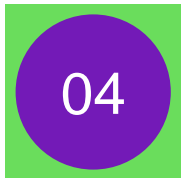
Plataforma Java



Primeira Aplicação



Alguns tipos de dados



Palavras reservadas



História, Ferramentas
e documentações,
Eclipse




Plataforma Java


(Introdução)

Processo de preparação de ambiente nas empresas

- A forma de rodar nas aplicações geralmente vem no github, no readme
- Algumas empresas duplicam para o Atlassian
- As empresas pedem para instalar os programas e dão a lista
- Ou elas dão uma máquina virtual pronta feita na AWS, Azure ou Google Cloud
- Exemplos práticos:
 - Porto Seguro (mostrar)
 - Banco Pan (mostrar)
 - Sulamérica

Exemplo Pan


 Conexão de Área de Trabalho Remota

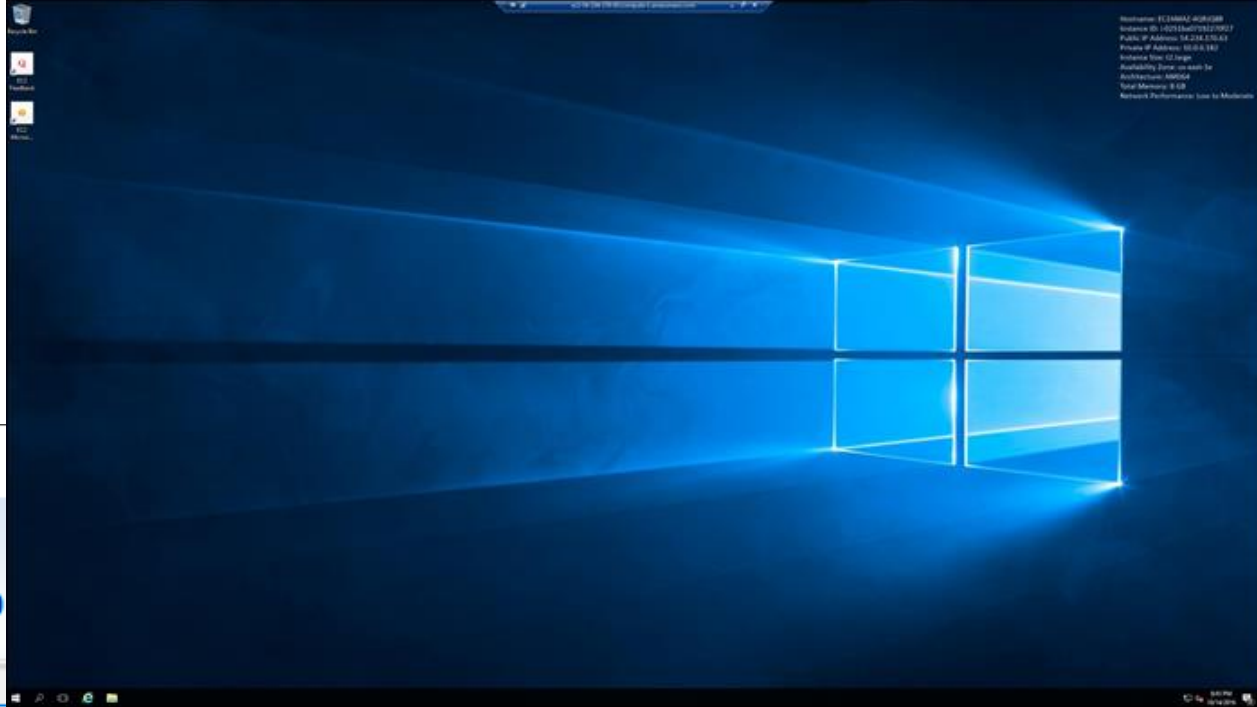
 **Conexão de
Área de Trabalho**

Computador:

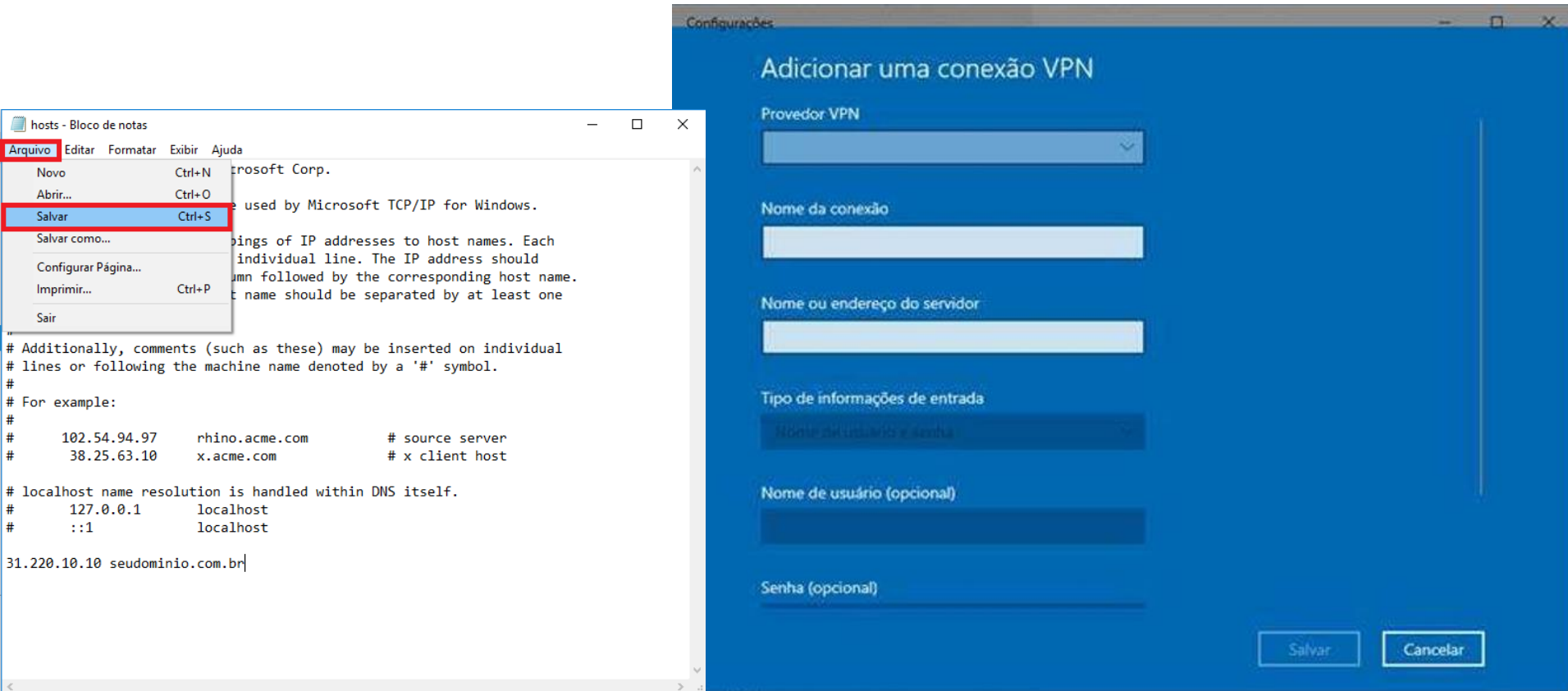
Nome de usuário: PAN-MATRIZ\...

Suas credenciais serão exigidas quando você se conectar.

 Mostrar Opções



Exemplo Porto Seguro





Programas

Java

- Eclipse
- NetBeans
- IntelliJ IDEA.
- Eclipse STS

Banco de Dados

- Dbeaver
- HeidiSql
- **Workbench**

Aws

- Aws CLI
- Plugin AWS para eclipse
- Plugin AWS para IntelliJ

Front

- Figma
- Visual Studio Code
- Sublime

Por quê Eclipse e não NETBEANS ?

- Mais rápido com Spring Boot do que o NET BEANS
- Netbeans é melhor para front e JSP
- Mais rápido

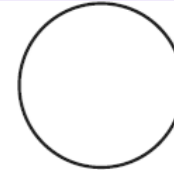
Por quê STS e não Eclipse

- Muito rápido, muito leve, plugins recentes de Spring Boot

E o IntelliJ Idea

- Apesar de ser bem popular precisa de 2 a 4 gb de memória
- Pago
- oferece poucos plug-ins quando comparado ao **eclipse**

Primeiro programa





Rodar primeiro programa

- <https://www.jdoodle.com/online-java-compiler/>

- ```
public class MainAplicattion {
```

- ```
    public static void main(String[] args) {
```
- ```
 System.out.println("Bem vindo ao java!");
```
- ```
        System.out.println("Olá Mundo!");
```
- ```
 }
```
- 
- ```
        System.out.print("Não pulo linha!");
```
- ```
 System.out.print("Não pulo linha!");
```
- ```
        System.out.println("Bem vindo\n ao\n mundo\nJava!");
```

- ```
 // Usar o código com cautela,
```

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

- ```
    //Control barra
```

- ```
}
```

|                 |           | Valores possíveis          |                            |              |         |                                                   |
|-----------------|-----------|----------------------------|----------------------------|--------------|---------|---------------------------------------------------|
| Tipos           | Primitivo | Menor                      | Maior                      | Valor Padrão | Tamanho | Exemplo                                           |
| Inteiro         | byte      | -128                       | 127                        | 0            | 8 bits  | byte ex1 = (byte)1;                               |
|                 | short     | -32768                     | 32767                      | 0            | 16 bits | short ex2 = (short)1;                             |
|                 | int       | -2.147.483.648             | 2.147.483.647              | 0            | 32 bits | int ex3 = 1;                                      |
|                 | long      | -9.223.372.036.854.770.000 | 9.223.372.036.854.770.000  | 0            | 64 bits | long ex4 = 1l;                                    |
| Ponto Flutuante | float     | -1,4024E-37                | 3.40282347E + 38           | 0            | 32 bits | float ex5 = 5.50f;                                |
|                 | double    | -4,94E-307                 | 1.79769313486231570E + 308 | 0            | 64 bits | double ex6 = 10.20d;<br>ou<br>double ex6 = 10.20; |
| Caractere       | char      | 0                          | 65535                      | \0           | 16 bits | char ex7 = 194;<br>ou<br>char ex8 = 'a';          |
| Booleano        | boolean   | false                      | true                       | false        | 1 bit   | boolean ex9 = true;                               |

# Tipos de dados

```
1 public class Main {
2 public static void main(String[] args) {
3 int myNum = 5;
4 float myFloatNum = 5.99f;
5 //float myFloatNumTeste = 5.99;
6 double myDoubleNum = 5.99;
7 double myDoubleNum2 = 5.99d;
8 char myLetter = 'D';
9 boolean myBool = true;
10 String myString = "Hello";
11
12 System.out.println(myNum);
13 System.out.println(myFloatNum);
14 //System.out.println(myFloatNumTeste);
15 System.out.println(myDoubleNum);
16 System.out.println(myDoubleNum2);
17 System.out.println(myLetter);
18 System.out.println(myBool);
19 System.out.println(myString);
20 }
21 }
```

```
public class MainAplicattion {
```

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

```
 String firstName = "Luiza ";
```

```
 String lastName = "Cerchiari";
```

```
 String fullName = firstName + lastName;
```

```
 System.out.println(fullName);
```

```
 int versao = 12;
```

```
 double preco = 10.2;
```

```
 int quantidade = 2;
```

```
 double total = preco + quantidade;
```

# Como criar variáveis

- Algo que as pessoas entendam (mostrar)
- Padrozinhar inglês ou português
- Usar **lowerCamelCase**

## Válidos:

- nomeCliente
- telefone\_1
- preco\$
- produtoAdquirido

## Inválidos:

- 1Telefone

# Palavras reservadas não devem ser variáveis

---

|          |           |         |              |          |            |
|----------|-----------|---------|--------------|----------|------------|
| 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   |           |         |              |          |            |

# História do java



# História Java

## Versão 11

- Métodos pré configurados
- `variavel.isBlank()`
- `variavel.strip`

## Linha do tempo das versões Java

|                   |                          |
|-------------------|--------------------------|
| Meta: 1994        | J2SE 5.0 (JDK 1.5) 2005  |
| 1996 (8 pacotes!) | Java SE 6 (JDK 1.6) 2006 |
| 5                 | Java SE 7 (JDK 1.7) 2011 |
| 1.2) 1998         | Java SE 8 (JDK 1.8) 2014 |
| 10                | Java SE 9 (JDK 9) 2017   |
| 12                | Java 10 (18.3) 3/2018    |
|                   | Java 11 (18.9) 9/2018    |

# Strip()

```
1 public class Main
2 {
3 public static void main(String[] args)
4 {
5 String str = " Hello World !! ";
6
7 System.out.println(str.strip()); /*"Hello World !" */ System.out.println("fim");
8
9 System.out.print(str.stripLeading()); /*"Hello World !! " */ System.out.println("fim");
10
11
12 System.out.print(str.stripTrailing()); /*" Hello World !" */ System.out.println("fim");
13 }
14 }
```

Hello World !!

fim

Hello World !! fim

Hello World !!fim

JDK 1.8.0\_66 ▼



Interactive

Stdin Inputs

CommandLine Arguments



Execute



Result

CPU Time: sec(s), Memory: kilobyte(s)

```
Main.java:7: error: cannot find symbol
 System.out.println(str.strip());
 ^
```

```
/*"Hello World !!" */ Sy
```

```
symbol: method strip()
location: variable str of type String
```

# Repeat()

---

```
1 public class Main
2 {
3 public static void main(String[] args)
4 {
5 String str = "Abc";
6
7 System.out.println(str.repeat(3));
8 }
9 }
```

**AbcAbcAbc**

# Antes

---

String repeat example usign regex

```
public class Main
{
 public static void main(String[] args)
 {
 String str = "Abc";

 String repeated = new String(new char[3]).replace("\0", str);

 System.out.println(repeated);
 }
}
```

# Depois

```
1 public class Main
2 {
3 public static void main(String[] args)
4 {
5 String str = "Abc";
6
7 System.out.println(str.repeat(3));
8 }
9 }
```

# Is blank

```
1 public class Main
2 {
3 public static void main(String[] args)
4 {
5 System.out.println("ABC".isBlank()); //false
6 System.out.println(" ".isBlank()); //true
7
8 System.out.println("ABC".isEmpty()); //false
9 System.out.println(" ".isEmpty()); //false
10 }
11 }
```

```
false
true
false
false
```

# Navegar

---

- <https://howtodoinjava.com/java11/>

# História Java

## Versão 14

- Helpful  
NullPointerExceptions;
- Pattern Matching

## versões Java

J2SE 5.0 (JDK 1.5) **2005**

Java SE 6 (JDK 1.6) **2006**

Java SE 7 (JDK 1.7) **2011**

Java SE 8 (JDK 1.8) **2014**

Java SE 9 (JDK 9) **2017**

Java 10 (18.3) **3/2018**

Java 11 (18.9) **9/2018**



# NullPointerExceptions

```
a.b.c = 99;
```

Irá lançar o NPE:

```
Exception in thread "main" java.lang.NullPointerException at
Prog.main(Prog.java:5)
```

A única informação que temos é que ocorreu na linha 5. Mas foi causado pelo “a” ou “b”? A partir de hoje saberemos:

```
Exception in thread "main" java.lang.NullPointerException: Cannot
read field "c" because "a.b" is null at Prog.main(Prog.java:5)
```

# Pattern matching

```
if (obj instanceof String) {
 String s = (String) obj;
 // usa s
}
```

```
if (obj instanceof String s) {
 // agora poderá usar s aqui
}
```

# Documentações (15 minutos para navegar)

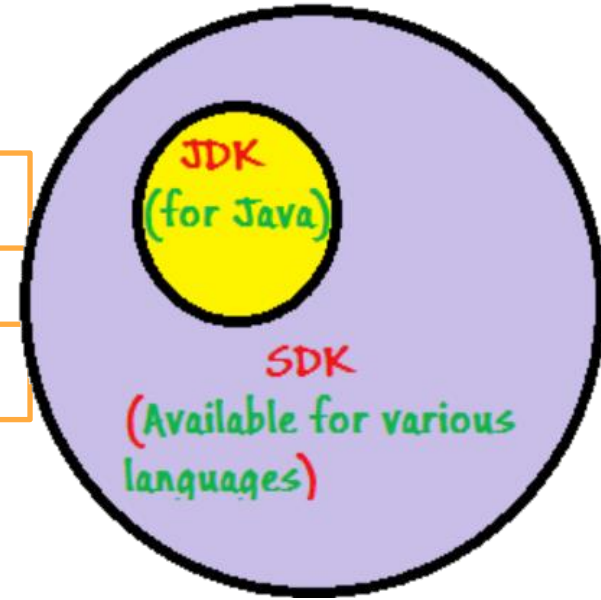
---

- [Java 11: https://howtodoinjava.com/java11/](https://howtodoinjava.com/java11/)
- Java 14: <https://medium.com/mobicareofficial/top-5-novidades-do-java-14-9082ee360bbe>
- Documentação: [https://www.w3schools.com/java/java\\_getstarted.asp](https://www.w3schools.com/java/java_getstarted.asp)
- <https://docs.oracle.com/javase/tutorial/getStarted/cupojava/netbeans.html>

JDK é o kit de desenvolvimento da linguagem **java** (Java JDK11)

SDK é o kit para várias linguagens (Android studio JDK)

JDK (Java Development Kit)  
SDK (Software Dev Kit)



# Mais diferenças

---

- Java SE
- Java EE
- Java ME
- <https://qastack.com.br/programming/2857376/difference-between-java-se-ee-me>

---

# Instalações



Um arquivo de código-fonte (com a extensão .java) contém uma definição de classe. A classe representa uma parte do seu programa e deve ficar dentro de um par de chaves:

```
public class PrimeiroPrograma{
}
```

Tudo o que acontece em Java, acontece dentro de uma classe. Quando você executar um programa em Java, na verdade vai executar uma classe. Então, executar um programa em Java significa informar para a JVM qual classe deve ser carregada para, em seguida, executar o método main desta classe. O main é um método especialmente desenvolvido para “dar partida” na aplicação:

```
public class PrimeiroPrograma{
 public static void main(String[] args) {
 // Seu código entra aqui
 }
}
```

# IDENTIFICADORES

Na linguagem de programação Java, um identificador é o nome dado a uma variável, classe ou método.

- O Java case-sensitive, ou seja, faz diferenciação entre letras maiúsculas e minúsculas.
- Não é permitido o uso de palavras reservadas como identificador.

|          |           |         |              |          |            |
|----------|-----------|---------|--------------|----------|------------|
| 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   |           |         |              |          |            |



## Identificadores válidos:

nomeDoUsuario

nome\_do\_usuario

\$variavel

\_variavel

## Identificadores inválidos:

3idade

nome do usuário

nome-do-usuario

# CONVENÇÕES DE CÓDIGO

- Pacotes – Os nomes dos pacotes devem ser substantivos escritos em letras minúsculas.

```
package locadora.cliente
```

- Classes e Interfaces – Os nomes das classes e das interfaces devem ser substantivos, combinando maiúsculas e minúsculas, com a primeira letra de cada palavra em maiúscula. Dentro de cada nome de classe ou interface, as palavras são separadas por letras maiúsculas.

```
class Cliente
```

```
interface PessoaFisica
```

- Métodos – Os nomes dos métodos devem ser verbos, combinando maiúsculas e minúsculas, tendo a primeira letra minúscula. Dentro de cada nome de método, as palavras são separadas por letras maiúsculas.

`cadastrarCliente()`

- Variáveis – Todas as variáveis devem ter uma combinação de maiúsculas e minúsculas, com a primeira letra em minúscula. As palavras são separadas por letras maiúsculas.

`clienteAtual`

- Os nomes das variáveis devem ser significativos e dar uma indicação de seu uso ao leitor. Evite nomes com um único caractere.

## COMENTÁRIOS

**// comentário de uma linha**

**/\* comentário de uma ou mais linhas**

**\*/**

**/\*\* Comentário para documentação**

**\* que também pode ter uma ou mais linhas**

**\*/**

# DECLARANDO E USANDO VARIÁVEIS

Java é fortemente tipado, o que significa que toda variável tem um tipo que não pode ser mudado, uma vez declarado.

```
tipoDaVariavel nomeDaVariavel;
```

```
int quantidade;
```

A partir dessa linha de código, a variável quantidade passa a existir na memória e, por ser do tipo **int**, armazena valores inteiros. Para isso, basta atribuir um valor a variável:

```
quantidade = 5;
```

ou

```
int quantidade = 5
```

```
// Exibe o conteúdo da variável quantidade
System.out.println(quantidade);
```

Tipos inteiros:

byte idade = 30;

short distancia = 31000;

int quantidade = 4552314;

long populacao = 23456787899l;

ou

long populacao = 23456787899L;

## Ponto flutuante ou números reais

`float altura = 1.87F;`

`float altura = 1.87f;`

`double peso = 90.35;`

`boolean maiorIdade = false;`

`char letra = 'b';`

Do tipo **boolean** armazena **true** ou **false** e o tipo **char** para armazenar um único caractere entre aspas simples.

O tipo `String` não é um tipo primitivo, mas sim uma classe que representa uma sequência de caracteres entre aspas duplas:

```
String nome = "Eduardo";
```



## OPERAÇÕES MATEMÁTICAS EM JAVA

### Adição

```
int adicao = 7 + 5;
```

### Subtração

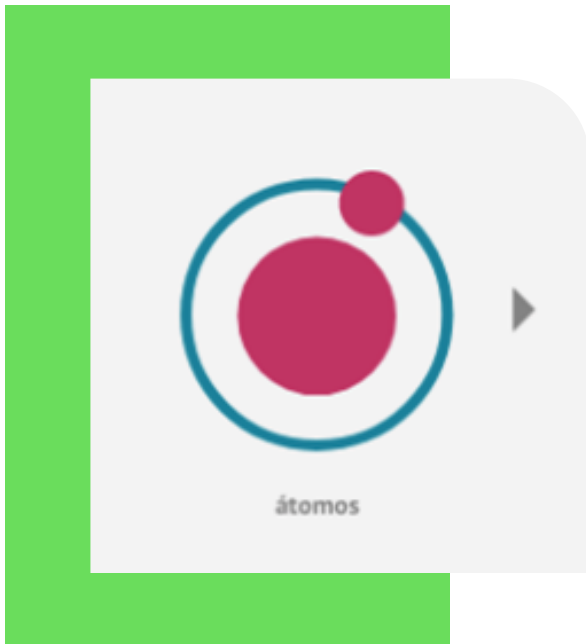
```
int subtracao = 7 - 5;
```

### Multiplicação

```
int multiplicacao = 7 * 5;
```

### Divisão

```
double divisao = 5 / 2d; // somente na divisão se coloca o d no final
```



Baixar Eclipse



Baixar Java 11

# Primeira APP em STS

---