INSTITUTO FEDERAL
Rio Grande do Sul
Campus Farroupilha

Aula 01: Introdução ao Java

Algoritmos e Programação 2

Karine Leite de Miranda
karine.miranda@farroupilha.ifrs.edu.br

Sumário

1. Linguagem de programação

É uma linguagem formal que, através de uma série de instruções, permite que um programador escreva um conjunto de ordens, ações consecutivas, dados e algoritmos para criar programas que controlam o comportamento físico e lógico de uma máquina.

A linguagem de programação é um sistema de **comunicação estruturado**, composto por conjuntos de **símbolos**, **palavras-chave**, regras **semânticas** e **sintáticas** que permitem **a comunicação entre um programador e uma máquina**.

1.1 Tipos de linguagem

Os programadores escrevem instruções em várias linguagens de programação, algumas diretamente compreensíveis por computadores, outras requerendo passos intermediários de tradução:

Linguagem de máquina	São dependentes de máquina (uma determinada linguagem de máquina pode ser utilizada apenas em um tipo de computador). Elas são complicadas para seres humanos.
Linguagem assembly	Em vez de utilizar strings de números que os computadores poderiam entender de maneira direta, os programadores começaram a usar abreviações em inglês para representar operações elementares (utiliza programas tradutores).
Linguagem de alto nível	Instruções únicas podem ser escritas para realizar tarefas. Os programas tradutores (compiladores) convertem os programas de linguagem de alto nível em linguagem de máquina. Linguagens de alto nível permitem aos programadores escrever instruções que se pareçam com o inglês/português cotidiano.

1.1 Tipos de linguagem

Programa que soma os ganhos em horas extras ao salário de base e armazena o resultado no salário bruto:

Máquina	Assembly	Alto nível
+1300042774 +1400593419 +1200274027	load basepay add overpay store grosspay	grossPay = basePay + overTimePay

O Java é um exemplo de linguagem de programação de alto nível.

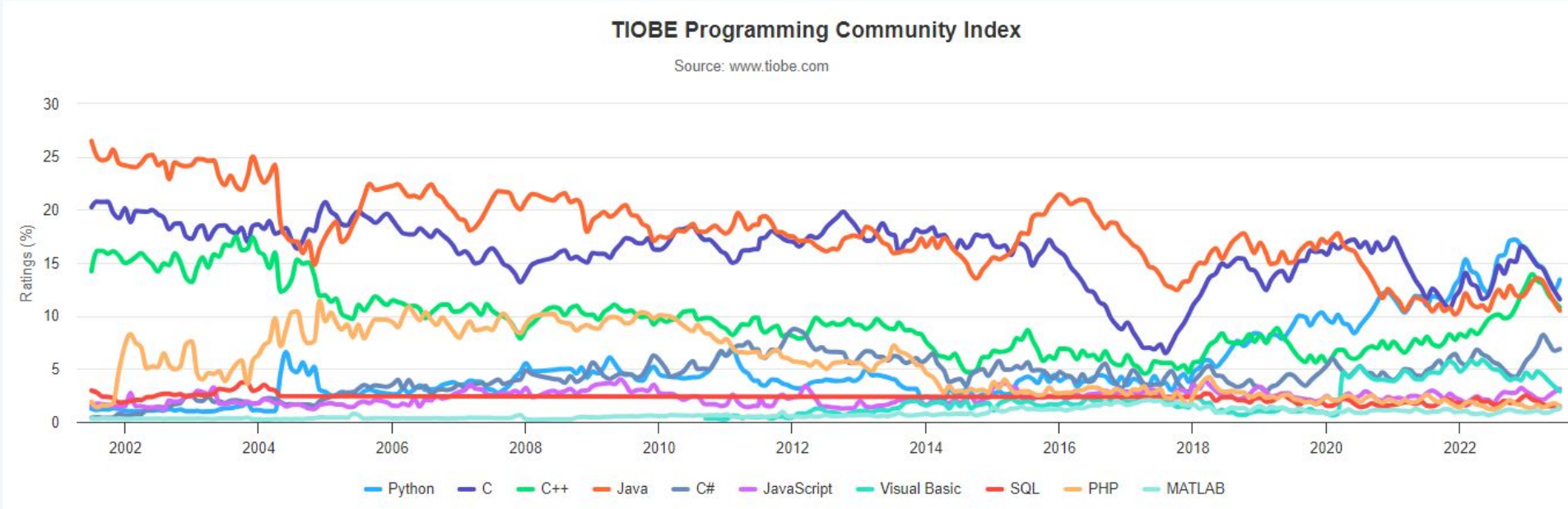
Qual linguagem usar?

1.2 Ranking das linguagens

Jul 2023	Jul 2022	Change	Programming Language	Ratings	Change
1	1		Python	13.42%	-0.01%
2	2		C	11.56%	-1.57%
3	4	↑	C++	10.80%	+0.79%
4	3	↓	Java	10.50%	-1.09%
5	5		C#	6.87%	+1.21%
6	7	↑	JavaScript	3.11%	+1.34%
7	6	↓	Visual Basic	2.90%	-2.07%
8	9	↑	SQL	1.48%	-0.16%
9	11	↑	PHP	1.41%	+0.21%
10	20	⇈	MATLAB	1.26%	+0.53%

https://www.tiobe.com/tiobe-index/

1.2 Ranking das linguagens



https://www.tiobe.com/tiobe-index/

2. Java

A Sun Microsystems, em 1991, financiou um projeto de pesquisa corporativa interna chefiado por James Gosling, que resultou em uma linguagem de **programação orientada a objetos** que a empresa chamou de Java.

Um objetivo-chave do Java é ser capaz de escrever programas a serem executados em uma grande v**ariedade de sistemas computacionais e dispositivos** controlados por computador. Isso às vezes é chamado de **“escreva uma vez, execute em qualquer lugar”.**


James Gosling

2. Java

Java Standard Edition

Contém os recursos necessários para desenvolver **aplicativos de desktop** e servidor.

Java Enterprise Edition

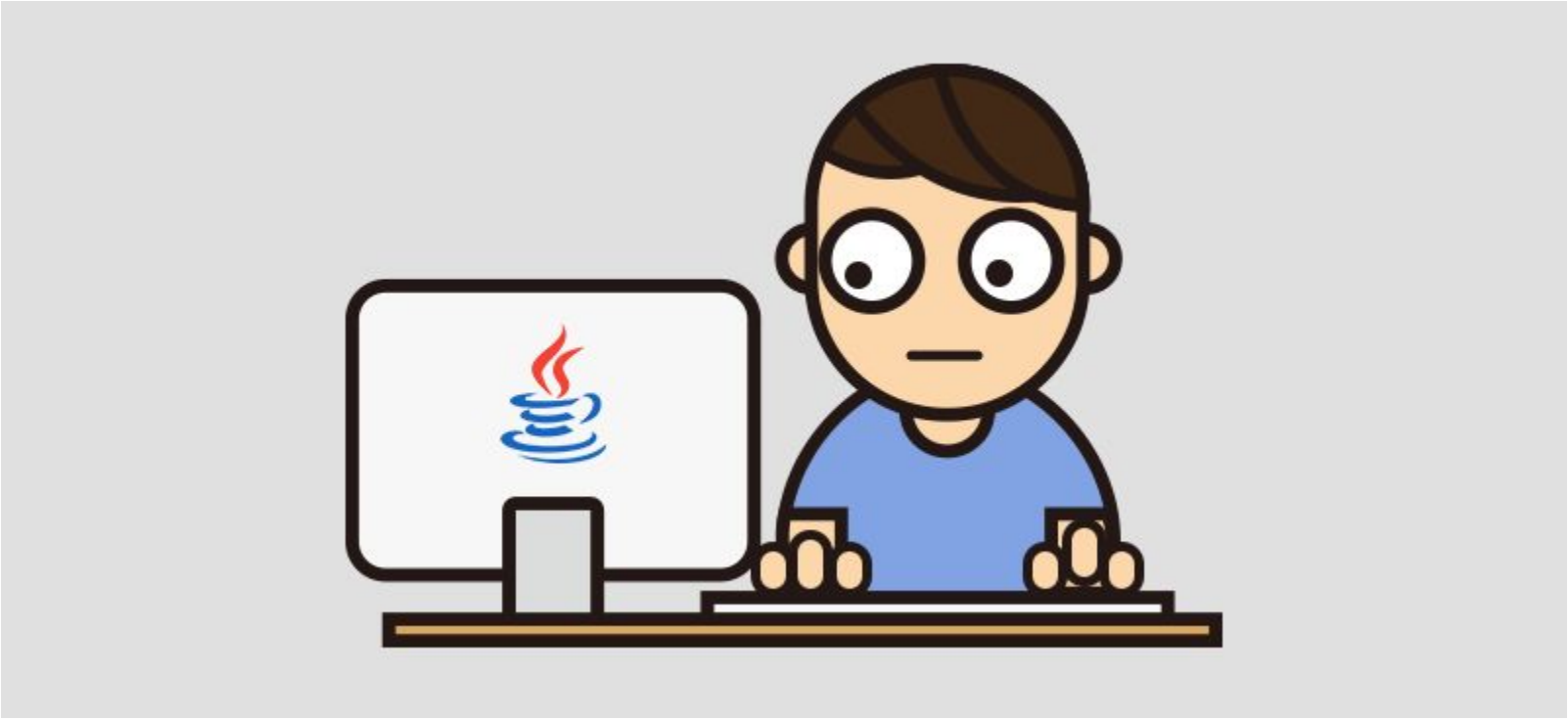
É adequado para desenvolver aplicativos em rede distribuída e em grande escala e também aplicativos **baseados na web.**

Java Micro Edition

É voltado para o desenvolvimento de aps para **dispositivos embarcados** com recursos limitados, como smartwatches.

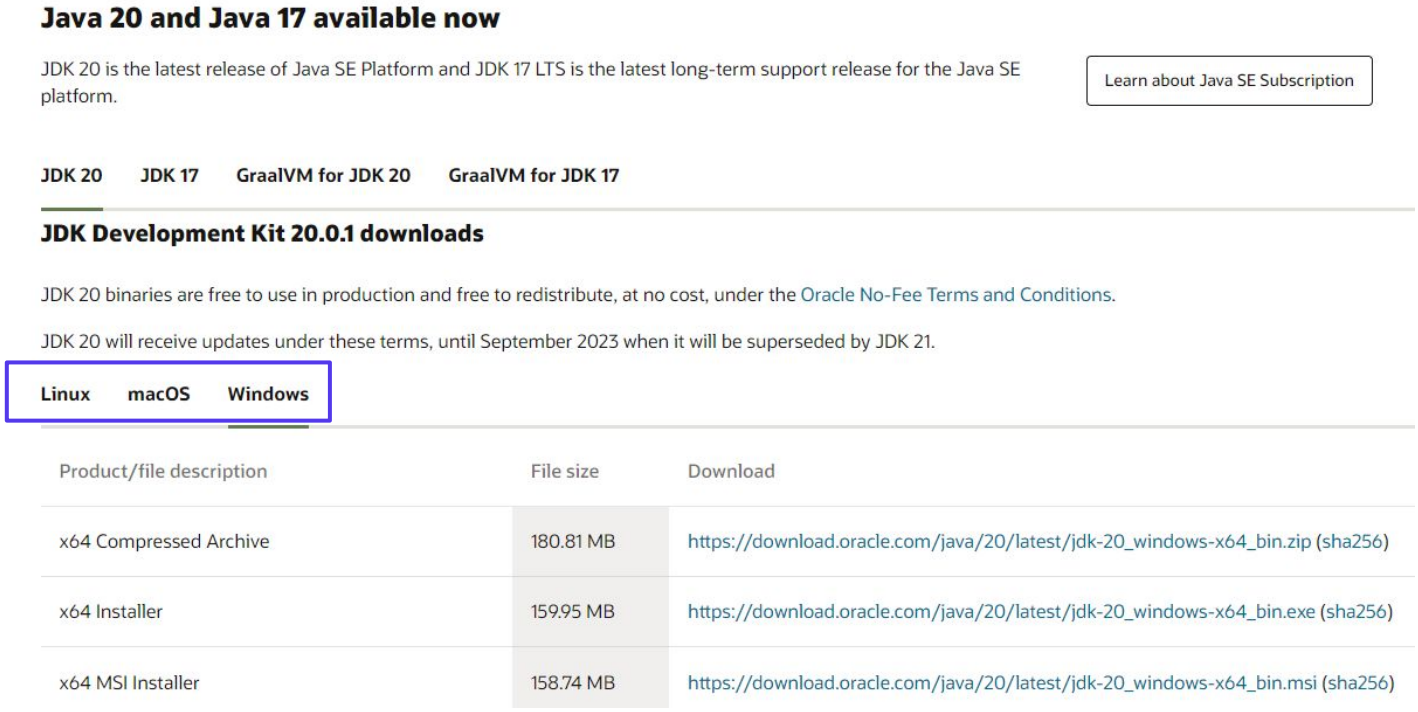
Três principais **kits de desenvolvimento de programação** (SDKs) disponíveis para criar vários tipos de aplicativos Java.

3. Preparação do ambiente



3.1 Instalação do JDK

Etapa 1: Baixar o JDK: https://www.oracle.com/java/technologies/downloads/



Java 20 and Java 17 available now

JDK 20 is the latest release of Java SE Platform and JDK 17 LTS is the latest long-term support release for the Java SE platform.

Learn about Java SE Subscription

JDK 20 | JDK 17 | GraalVM for JDK 20 | GraalVM for JDK 17

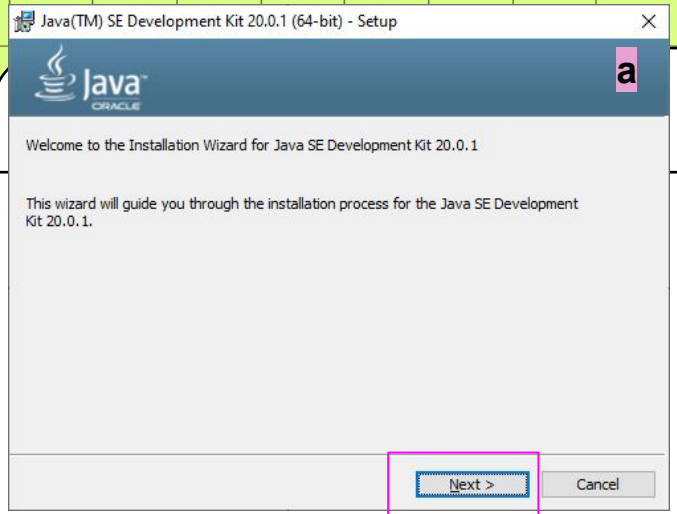
JDK Development Kit 20.0.1 downloads

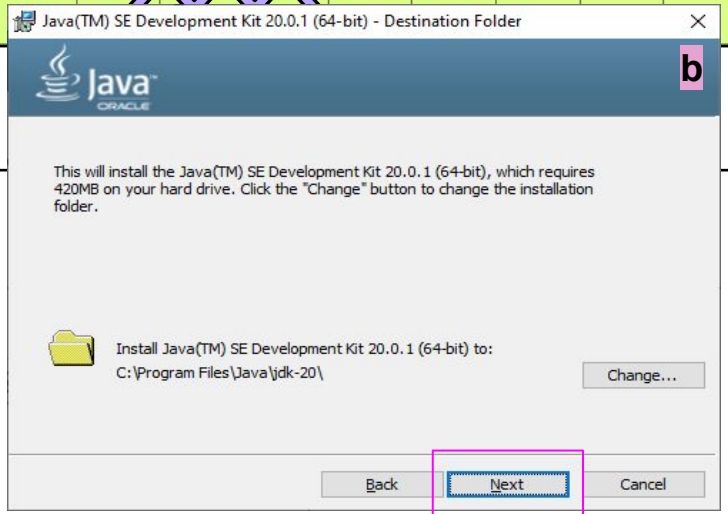
JDK 20 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions.

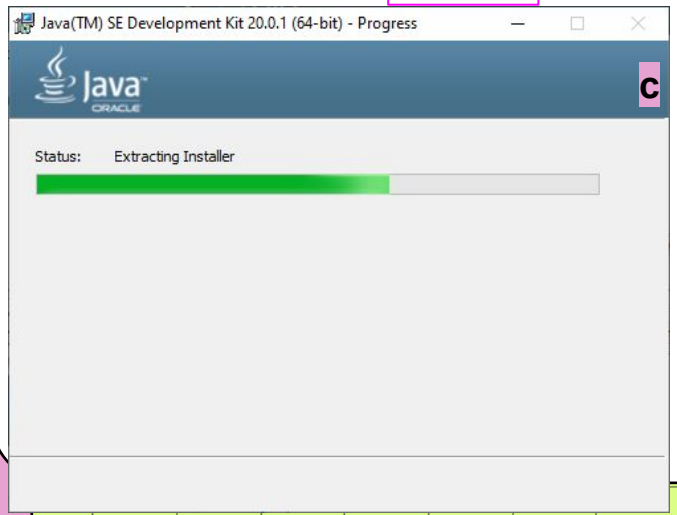
JDK 20 will receive updates under these terms, until September 2023 when it will be superseded by JDK 21.

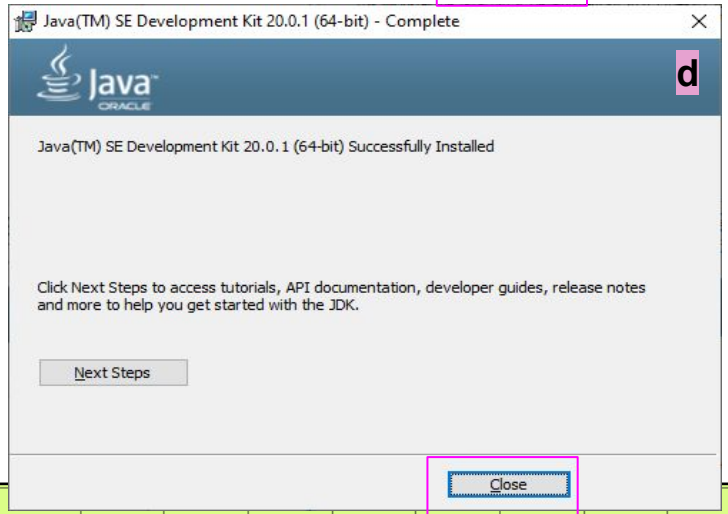
Linux | macOS | Windows

Product/file description	File size	Download
x64 Compressed Archive	180.81 MB	https://download.oracle.com/java/20/latest/jdk-20_windows-x64_bin.zip (sha256)
x64 Installer	159.95 MB	https://download.oracle.com/java/20/latest/jdk-20_windows-x64_bin.exe (sha256)
x64 MSI Installer	158.74 MB	https://download.oracle.com/java/20/latest/jdk-20_windows-x64_bin.msi (sha256)









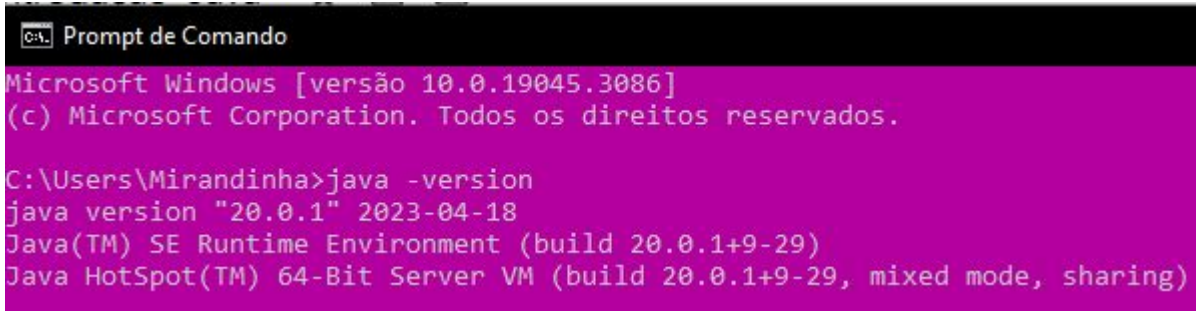
Etapa 2:

Executar o instalador

3.3 Teste - JDK

Etapa 3: Testar a instalação do JDK.

Abrir o prompt e executar o comando:
java - version



3.1.1 Variável de ambiente

Etapa 3.1: Se o teste falhar, é necessário criar uma variável de ambiente indicando onde o java está instalado.

A **variável de ambiente PATH** especifica em quais diretórios o computador pesquisa ao procurar aplicativos, como os aplicativos que permitem **compilar e executar seus aplicativos Java** (chamados javac e java, respectivamente).

https://java.tutorials24x7.com/blog/how-to-install-java-18-on-windows

3.4 Ambientes de desenvolvimento

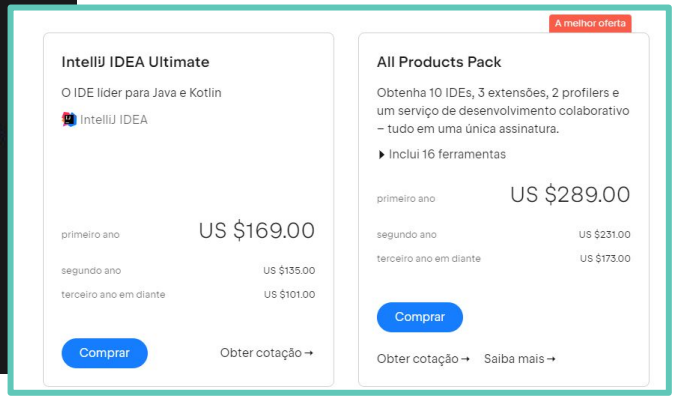
O IDE ou Integrated Development Environment **(Ambiente de Desenvolvimento Integrado)** é um software que auxilia no desenvolvimento de aplicações. Desta forma, combinam **ferramentas comuns** em uma única interface gráfica do usuário (GUI).

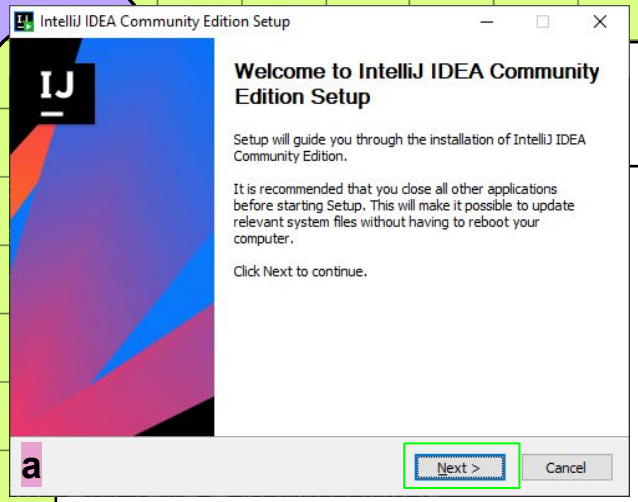
3.4.1 IntelliJ

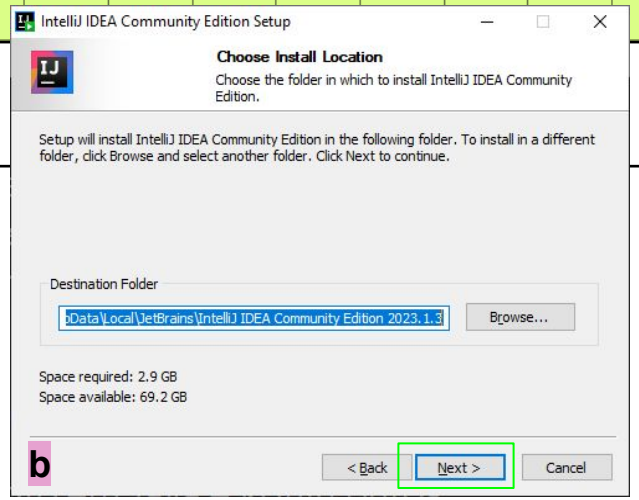
Etapa 3: https://www.jetbrains.com/pt-br/idea/download/?section=windows

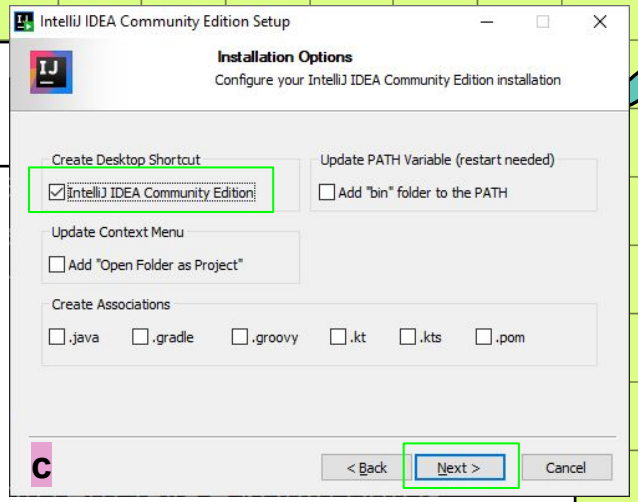
3.4.1 IntelliJ

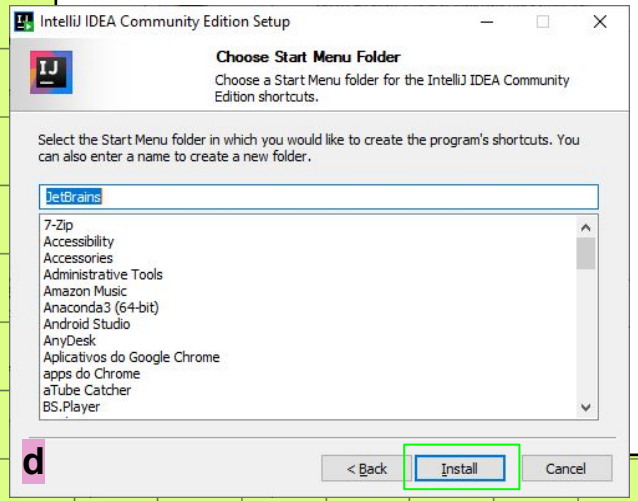
Baixar a versão **Community**.

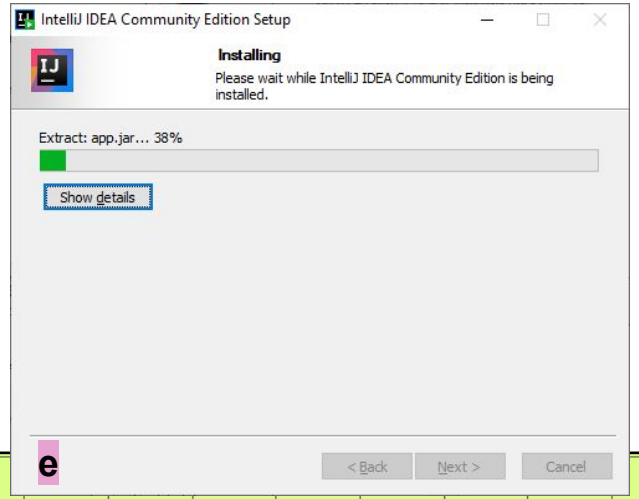


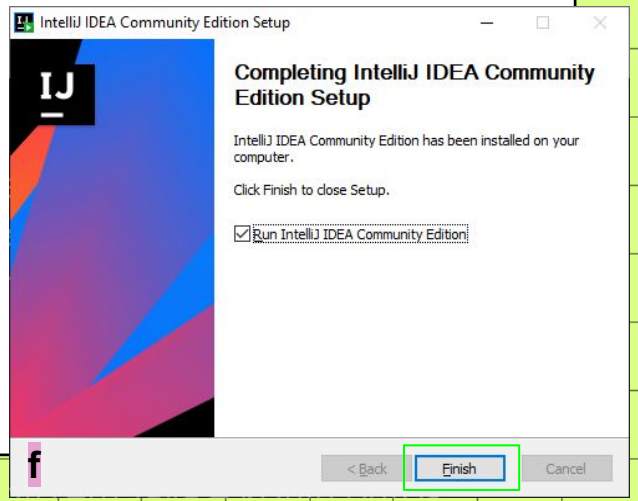

IntelliJ IDEA Community Edition Setup
Welcome to IntelliJ IDEA Community Edition Setup
Setup will guide you through the installation of IntelliJ IDEA Community Edition.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel
a


IntelliJ IDEA Community Edition Setup
Choose Install Location
Choose the folder in which to install IntelliJ IDEA Community Edition.
Setup will install IntelliJ IDEA Community Edition in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
oData\Local\JetBrains\IntelliJ IDEA Community Edition 2023.1.3
Browse...
Space required: 2.9 GB
Space available: 69.2 GB
< Back
Next >
Cancel
b


IntelliJ IDEA Community Edition Setup
Installation Options
Configure your IntelliJ IDEA Community Edition installation
Create Desktop Shortcut
IntelliJ IDEA Community Edition
Update PATH Variable (restart needed)
Add "bin" folder to the PATH
Update Context Menu
Add "Open Folder as Project"
Create Associations
.java
.gradle
.groovy
.kt
.kts
.pom
< Back
Next >
Cancel
c

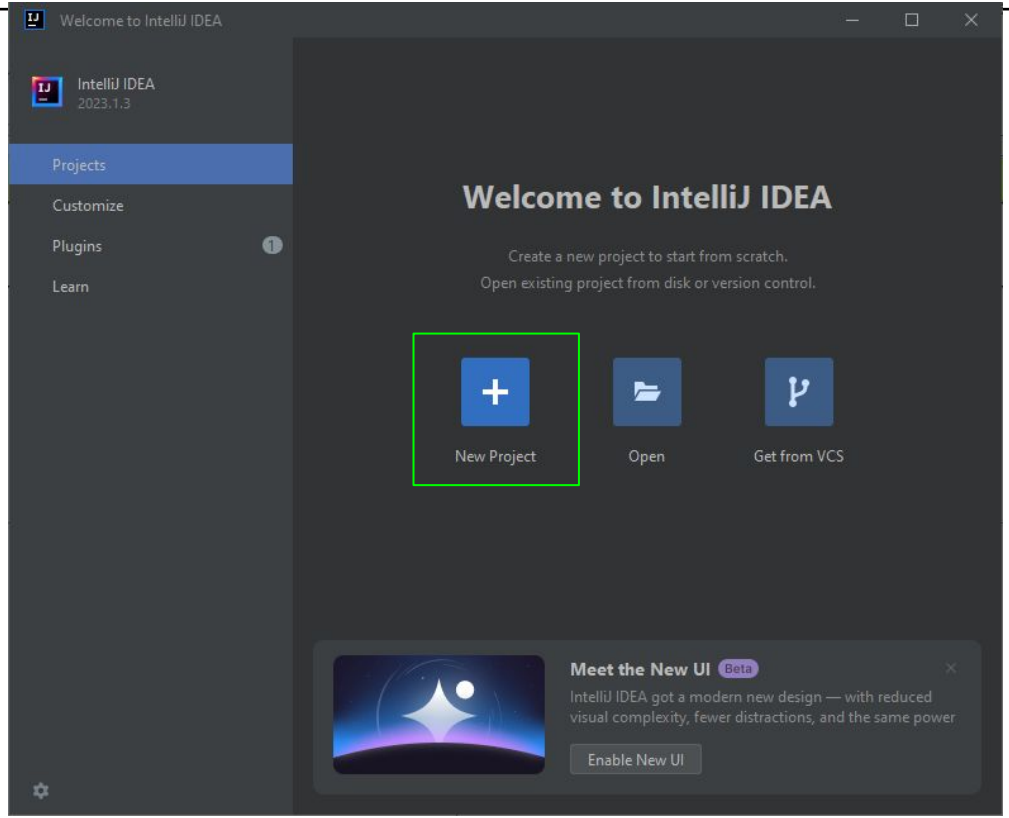

IntelliJ IDEA Community Edition Setup
Choose Start Menu Folder
Choose a Start Menu folder for the IntelliJ IDEA Community Edition shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
JetBrains
7-Zip
Accessibility
Accessories
Administrative Tools
Amazon Music
Anaconda3 (64-bit)
Android Studio
AnyDesk
Aplicativos do Google Chrome
apps do Chrome
aTube Catcher
BS.Player
< Back
Install
Cancel
d


IntelliJ IDEA Community Edition Setup
Installing
Please wait while IntelliJ IDEA Community Edition is being installed.
Extract: app.jar... 38%
Show details
< Back
Next >
Cancel
e

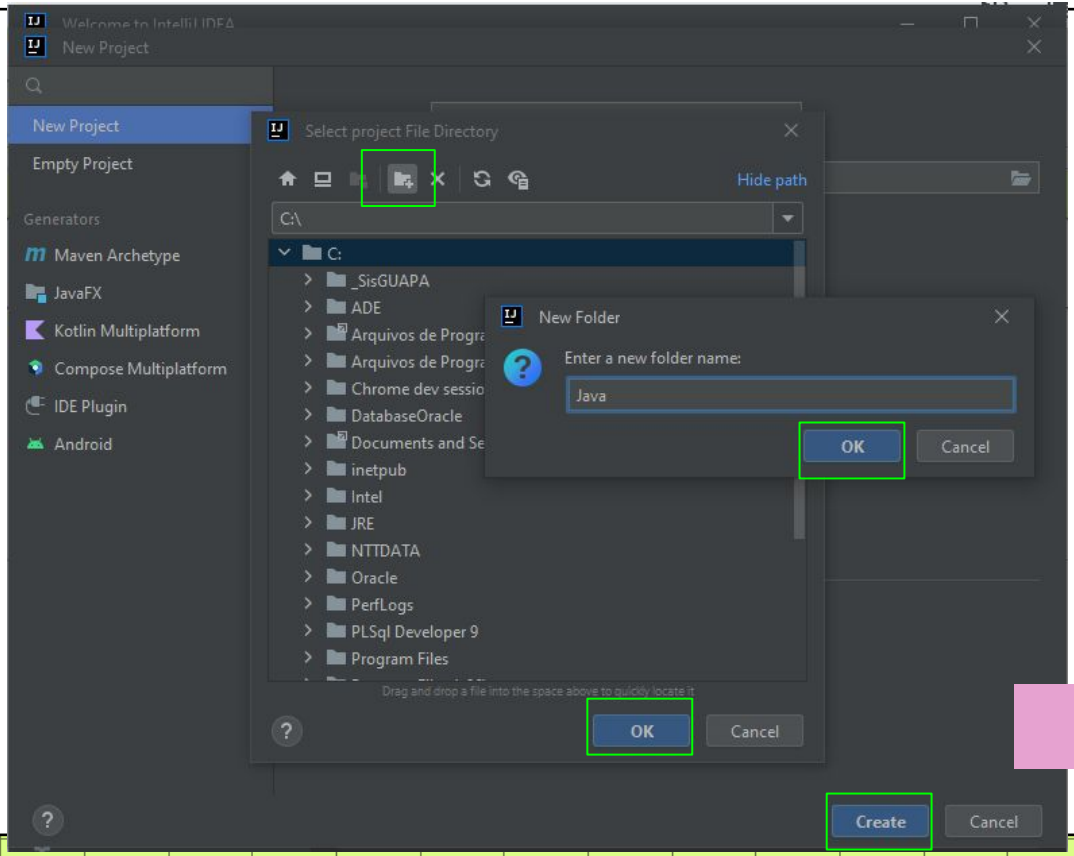

IntelliJ IDEA Community Edition Setup
Completing IntelliJ IDEA Community Edition Setup
IntelliJ IDEA Community Edition has been installed on your computer.
Click Finish to close Setup.
Run IntelliJ IDEA Community Edition
< Back
Finish
Cancel
f

4. Hands-on

Novo projeto

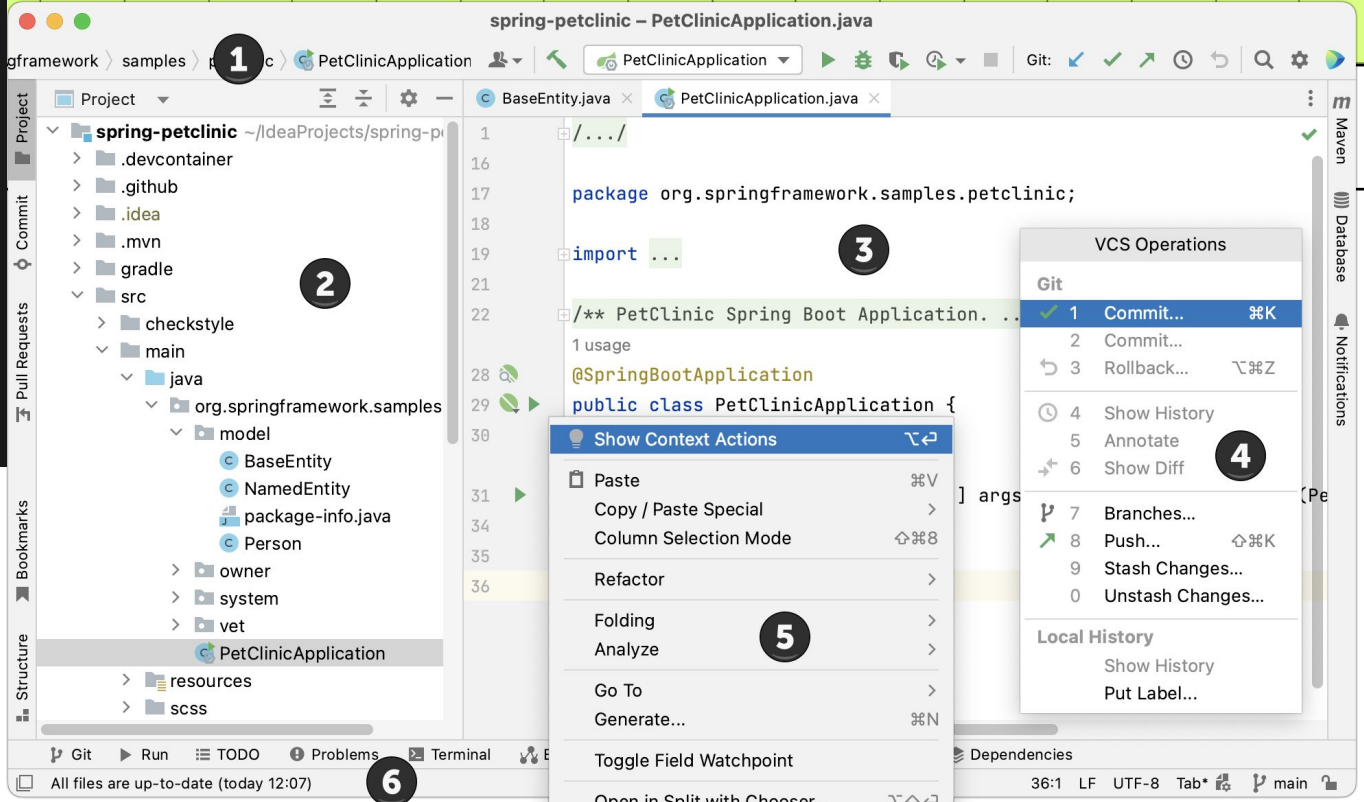


Novo projeto



Projeto_aula_01

1. Navigation bar
2. Project tool window
3. Editor
4. Popup menu
5. Context menu
6. Status bar



Sintaxe

```
1  // Figura 2.1: Welcome1.java
2  // Programa de impressão de texto.
3
4  public class Welcome1
5  {
6     // método main inicia a execução do aplicativo Java
7     public static void main(String[] args)
8     {
9        System.out.println("Welcome to Java Programming!");
10    } // fim do método main
11 } // fim da classe Welcome1
```

```
Welcome to Java Programming!
```

Figura 2.1 | Programa de impressão de texto.

Sintaxe: comentários

Inserimos comentários para documentar programas e aprimorar sua legibilidade. O compilador Java ignora os comentários, portanto, eles não fazem o computador realizar qualquer ação quando o programa é executado.

Comentário de fim de linha	Comentário tradicional
`// Programa de impressão de texto.`	`/* Esse é um comentário tradicional. Ele pode ser dividido em várias linhas */`

Linhas (1,2, 6,10 e 11).

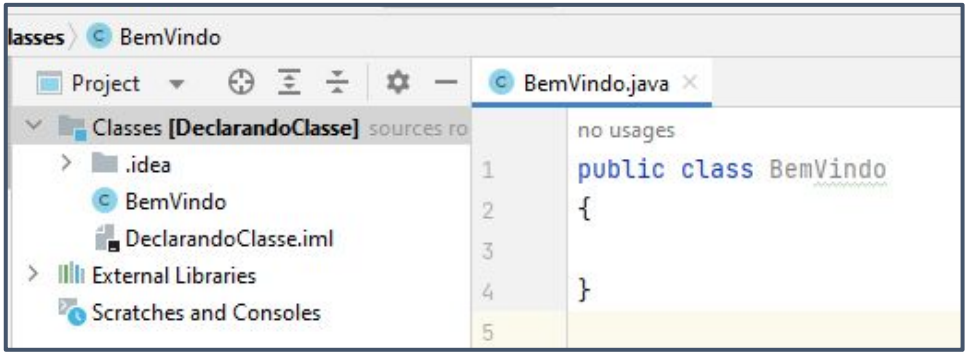
Sintaxe: classe

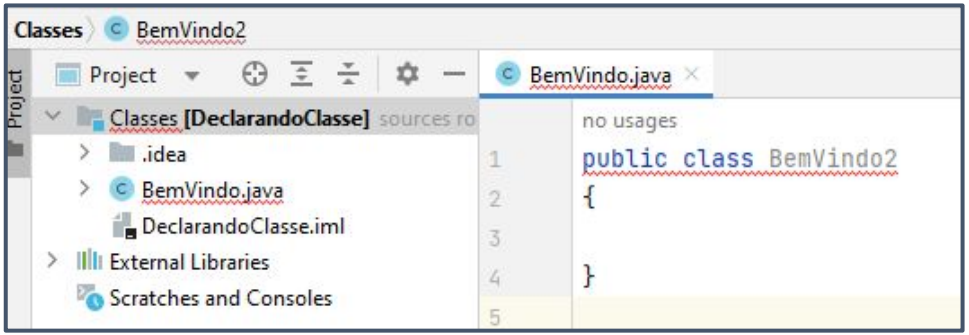
A linha 4 começa uma declaração de **class** para a **classe** Welcome1. Todo programa Java consiste em pelo menos uma classe que você (o programador) define. A palavra-chave class introduz uma declaração de classe e é imediatamente seguida pelo nome da classe (Welcome1).

```
4  public class Welcome1
5  {
6     // método main inicia a execução do aplicativo Java
7     public static void main(String[] args)
8     {
9        System.out.println("Welcome to Java Programming!");
10    } // fim do método main
11 } // fim da classe Welcome1
```

Palavras-chave (às vezes chamadas de palavras reservadas) são reservadas para uso pelo Java e sempre escritas com todas as **letras minúsculas**.

Sintaxe: classe





Erro de compilação!

Uma classe public deve ser inserida em um arquivo com um nome na forma NomeDaClasse.java, assim a classe BemVindo é armazenada no arquivo BenVindo.java.

Sintaxe: classe

- Por convenção, os nomes de classes iniciam com uma **letra maiúscula** e apresentam **a letra inicial de cada palavra que eles incluem em maiúscula** (por exemplo, SampleClassName).

- **O nome de uma classe é um identificador** — uma série de caracteres que consiste em letras, dígitos, sublinhados (_) e sinais de cifrão ($) que não inicie com um dígito e não contenha espaços.

- Alguns identificadores válidos são Welcome1, $valor, _valor, m_campoDeEntrada1 e botao7.

- O nome 7botao **não é** um identificador válido porque inicia com um dígito, e o nome campo de entrada não é um identificador válido porque contém espaços.

- O **Java faz distinção entre maiúsculas e minúsculas** — letras maiúsculas e letras minúsculas são diferentes — assim, value e Value são identificadores diferentes (mas ambos válidos).

Sintaxe: classe

```
4  public class Welcome1
5  {
6     // método main inicia a execução do aplicativo Java
7     public static void main(String[] args)
8     {
9        System.out.println("Welcome to Java Programming!");
10    } // fim do método main
11 } // fim da classe Welcome1
```

A chave esquerda (como na linha 5), **{**, **inicia o corpo de cada declaração de classe.** Uma chave direita correspondente (na linha 11), **}**, **deve terminar cada declaração de classe**. As linhas 6 a 10 são recuadas.

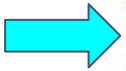
Sintaxe: palavras-chave

Palavras-chave do Java				
abstract	assert	boolean	break	byte
case	catch	char	class	continue
default	do	double	else	enum
extends	final	finally	float	for
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		
Palavras-chave que não são atualmente utilizadas				
const	goto			

Figura C.1 | Palavras-chave do Java.

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.

Sintaxe: método



```
4  public class Welcome1
5  {
6     // método main inicia a execução do aplicativo Java
7     public static void main(String[] args)
8     {
9        System.out.println("Welcome to Java Programming!");
10    } // fim do método main
11 } // fim da classe Welcome1
```

Na linha 7 tem-se **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** e ser definido como mostrado na linha 7; caso contrário, a JVM não executará o aplicativo.

A chave esquerda na linha 8 inicia o **corpo da declaração do método.** Uma chave direita correspondente deve terminá-la (linha 10).

Sintaxe: método

```
4  public class Welcome1
5  {
6     // método main inicia a execução do aplicativo Java
7     public static void main(String[] args)
8     {
9        System.out.println("Welcome to Java Programming!");
10    } // fim do método main
11 } // fim da classe Welcome1
```

A linha 9 instrui o computador a executar uma ação, ou seja, **exibir os caracteres** contidos entre as aspas duplas.

O objeto **System.out** — é conhecido como objeto de saída padrão. Ele permite que um aplicativo Java **exiba informações na janela de comando** a partir da qual ele é executado.

O método **System.out.println exibe** (ou imprime) u**ma linha de texto na janela de comando.** A string entre parênteses na linha 9 é o **argumento** para o método.

A linha 9 inteira é uma **instrução.**
Um método normalmente contém uma ou mais instruções que executam a tarefa.

Sintaxe: tipos primitivos

Tipo	Tamanho em bits	Valores
`boolean`		`true` ou `false`
`char`	16	`'\u0000'` a `'\uFFFF'` (0 a 65535)
`byte`	8	–128 a +127 (-2^7 a $2^7 - 1$)
`short`	16	–32.768 a +32.767 (-2^{15} a $2^{15} - 1$)
`int`	32	–2.147.483.648 a +2.147.483.647 (-2^{31} a $2^{31} - 1$)
`long`	64	–9.223.372.036.854.775.808 a +9.223.372.036.854.775.807 (-2^{63} a $2^{63} - 1$)
`float`	32	*Intervalo negativo:* –3,4028234663852886E+38 a –1,4012984632481707e–45 *Intervalo positivo:* 1,4012984632481707e–45 a 3,4028234663852886E+38
`double`	64	*Intervalo negativo:* –1,7976931348623157E+308 a –4,94065645841246544e–324 *Intervalo positivo:* 4,94065645841246544e–324 a 1,7976931348623157E+308

5.Exemplo

```java
// Figura 2.7: Addition.java
// Programa de adição que insere dois números, então exibe a soma deles.
import java.util.Scanner; // programa utiliza a classe Scanner

public class Addition
{
   // método main inicia a execução do aplicativo Java
   public static void main(String[] args)
   {
      // cria um Scanner para obter entrada a partir da janela de comando
      Scanner input = new Scanner(System.in);

      int number1; // primeiro número a somar
      int number2; // segundo número a somar
      int sum; // soma de number1 e number2

      System.out.print("Enter first integer: "); // prompt
      number1 = input.nextInt(); // lê primeiro o número fornecido pelo usuário

      System.out.print("Enter second integer: "); // prompt
      number2 = input.nextInt(); // lê o segundo número fornecido pelo usuário

      sum = number1 + number2; // soma os números, depois armazena o total em sum

      System.out.printf("Sum is %d%n", sum); // exibe a soma
   } // fim do método main
} // fim da classe Addition
```

Este programa lê dois números digitados pelo usuário, calcula a sua soma e a exibe.

Os programas lembram-se dos números e de outros dados na **memória** do computador e os acessam por meio de elementos de programa chamados **variáveis**.

5.Exemplo

Um dos pontos fortes do Java é seu rico conjunto de **classes predefinidas** que você pode reutilizar. **Essas classes são agrupadas em pacotes** — chamados de grupos de classes relacionadas — e, coletivamente, são chamadas de **biblioteca de classes** Java, ou Java Application Programming Interface (Java API).

```
1  // Figura 2.7: Addition.java
2  // Programa de adição que insere dois números, então exibe a soma deles.
3  import java.util.Scanner; // programa utiliza a classe Scanner
```

Todas as declarações import devem aparecer antes da primeira declaração da classe no arquivo (caso contrário, erro de sintaxe).

Sintaxe: declaração de variável

Uma **variável** é uma posição na memória do computador na qual um valor pode ser **armazenado** para utilização posterior em um programa.

```
10          // cria um Scanner para obter entrada a partir da janela de comando
11          Scanner input = new Scanner(System.in);
12
13          int number1; // primeiro número a somar
14          int number2; // segundo número a somar
15          int sum; // soma de number1 e number2
```

Todas as variáveis Java devem ser declaradas com um **nome** e um **tipo** antes que elas possam ser utilizadas

5.Exemplo

Na linha 17, usa-se **System.out.print** para exibir uma mensagem (sem quebra de linha).

Na linha 18 utiliza o método **nextInt** do valor de input do objeto Scanner para obter um inteiro digitado pelo usuário. Nesse momento o programa espera que o usuário digite o número e pressione a tecla Enter para submeter esse número a ele.

```
17          System.out.print("Enter first integer: "); // prompt
18          number1 = input.nextInt(); // lê primeiro o número fornecido pelo usuário
```

Na linha 18 colocamos o resultado da chamada ao método nextInt (um valor int) na variável number1 utilizando o **operador de atribuição, =.** O operador = é chamado de operador binário, porque tem dois operandos — number1 e o resultado da chamada de método input.nextInt(). Essa instrução inteira é chamada instrução de atribuição, pois atribui um valor a uma variável.

Sintaxe: usando variáveis em um cálculo

A linha 20 exibe uma mensagem solicitando ao usuário que informe um segundo número, e este valor é atribuído a variável “number2”, na linha 21.

```
20          System.out.print("Enter second integer: "); // prompt
21          number2 = input.nextInt(); // lê o segundo número fornecido pelo usuário
```

A linha 23 é uma **instrução de atribuição** que calcula a soma das variáveis number1 e number2 e, então, atribui o resultado à variável sum utilizando o operador de atribuição, =.

```
23          sum = number1 + number2; // soma os números, depois armazena o total em sum
```

As partes das instruções que contêm cálculos são chamadas **expressões**. De fato, uma expressão é qualquer parte de uma **instrução que tem um valor associado com ela**. Por exemplo, o valor da expressão number1 + number2 é a soma dos números.

5.Exemplo

```
25          System.out.printf("Sum is %d%n", sum); // exibe a soma
```

Por fim, a linha 25, exibe o resultado da soma. O método System.out.printf exibe o resultado da soma. O especificador de formato **%d** é um marcador de lugar para um valor int (nesse caso, o valor de sum) — a letra d significa "**inteiro decimal**". Todos os caracteres restantes na string de formato são texto fixo.

Os cálculos também podem ser realizados dentro de instruções printf. Poderíamos ter combinado as instruções nas linhas 23 e 25 na instrução.

```
System.out.printf("Sum is %d%n", (number1 + number2));
```

Sintaxe: operadores aritméticos

A maioria dos programas realiza cálculos aritméticos. Note o uso de vários símbolos especiais não utilizados na álgebra. O asterisco (*) indica multiplicação, e o sinal de porcentagem (%) é o operador de resto, etc.

Operação Java	Operador	Expressão algébrica	Expressão Java
Adição	+	$f + 7$	`f + 7`
Subtração	-	$p - c$	`p - c`
Multiplicação	*	bm	`b * m`
Divisão	/	x/y ou $\frac{x}{y}$ ou $x \div y$	`x / y`
Resto	%	$r \bmod s$	`r % s`

Sintaxe: regras de precedência do operador

O Java aplica os operadores em expressões aritméticas em uma sequência precisa determinada pelas seguintes **regras de precedência de operador:**

Operador(es)	Operação(ões)	Ordem de avaliação (precedência)
*	Multiplicação	Avaliado primeiro. Se houver vários operadores desse tipo, eles são avaliados da *esquerda para a direita*.
/	Divisão	
%	Resto	
+	Adição	Avaliado em seguida. Se houver vários operadores desse tipo, eles são avaliados da *esquerda para a direita*.
-	Subtração	
=	Atribuição	Avaliado por último.

- Os operadores de multiplicação, divisão e resto têm o mesmo nível de precedência.
- Os operadores de adição e subtração têm o mesmo nível de precedência.
- Essas regras permitem que o Java aplique os operadores na ordem correta.

Exemplos

Álgebra: $m = \frac{a + b + c + d + e}{5}$

Java:
```
m = (a + b + c + d + e) / 5;
```

Usar parênteses: $a + b + c + d + \frac{e}{5}$

Álgebra: $y = mx + b$

Java:
```
y = m * x + b;
```

Equação de uma linha reta

Álgebra: $z = pr\%q + w/x - y$

Java:
```
z = p * r % q + w / x - y;
```

6 1 2 4 3 5

```
y = a * x * x + b * x + c;
```

6 1 2 4 3 5

Polinômio de segundo grau: ax2 + bx + c

Sintaxe: tomada de decisão - operadores de igualdade e operadores relacionais

Uma condição é uma expressão que pode ser true ou false. A instrução de seleção **if** do Java que permite a um programa tomar uma decisão com base no valor de uma condição.

Operador algébrico	Operador de igualdade ou relacional Java	Exemplo de condição em Java	Significado da condição em Java
Operadores de igualdade			
=	==	x == y	x é igual a y
≠	!=	x != y	x é não igual a y
Operadores relacionais			
>	>	x > y	x é maior que y
<	<	x < y	x é menor que y
≥	>=	x >= y	x é maior que ou igual a y
≤	<=	x <= y	x é menor que ou igual a y

As condições nas instruções if podem ser formadas utilizando os operadores de igualdade (== e !=) e os operadores relacionais (>, <, >= e <=).

Exemplo

Qual o erro?

```
a) if (c < 7);
       System.out.println("c is less than 7");
b) if (c => 7)
       System.out.println("c is equal to or greater than 7");
```

LeTexto.java

```java
import java.util.Scanner;

public class LeTexto {

    public static void main(String[] args)
    {
        Scanner leitor = new Scanner(System.in);

        System.out.print("Nome completo: ");
        String nome = leitor.nextLine();
        System.out.println("Seu  nome é " + nome);

        System.out.print("Nome completo: ");
        String nome2 = leitor.next();
        System.out.println("Seu  nome é " + nome2);
    }
}
```

Run: LeTexto

```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:(
Nome completo: Karine Miranda
Seu  nome é Karine Miranda
Nome completo: Karine Miranda
Seu  nome é Karine
```

Leitura de inputs no formato de texto.

Demais tipos de input:

```java
Scanner sc = new Scanner(System.in);

float numF = sc.nextFloat();
int num1 = sc.nextInt();
byte byte1 = sc.nextByte();
long lg1 = sc.nextLong();
boolean b1 = sc.nextBoolean();
double num2 = sc.nextDouble();
String nome = sc.nextLine();
```

Exercícios

2.2 Determine se cada uma das seguintes afirmações é *verdadeira* ou *falsa*. Se *falsa*, explique por quê.

a) Os comentários fazem com que o computador imprima o texto depois das `//` na tela quando o programa executa.

b) Todas as variáveis devem ser atribuídas a um tipo quando são declaradas.

c) O Java considera que as variáveis `number` e `NuMbEr` são idênticas.

d) O operador de resto (`%`) pode ser utilizado apenas com operandos inteiros.

e) Os operadores aritméticos `*`, `/`, `%`, `+` e `-` têm, todos, o mesmo nível de precedência.

Exercícios

2.5 Escreva declarações, instruções ou comentários que realizem cada uma das tarefas a seguir:

a) Declare que um programa calculará o produto de três inteiros.

b) Crie um `Scanner` chamado `input` que leia valores a partir da entrada padrão.

c) Declare as variáveis `x`, `y`, `z` e `result` como tipo `int`.

d) Solicite que o usuário insira o primeiro inteiro.

e) Leia o primeiro inteiro digitado pelo usuário e armazene-o na variável `x`.

f) Solicite que o usuário insira o segundo inteiro.

g) Leia o segundo inteiro digitado pelo usuário e armazene-o na variável `y`.

h) Solicite que o usuário insira o terceiro inteiro.

i) Leia o terceiro inteiro digitado pelo usuário e armazene-o na variável `z`.

j) Compute o produto dos três inteiros contidos nas variáveis `x`, `y` e `z` e atribua o resultado à variável `result`.

k) Use `System.out.printf` para exibir a mensagem "`Product is`" seguida pelo valor da variável `result`.

2.6 Usando as instruções que você escreveu no Exercício 2.5, elabore um programa completo que calcule e imprima o produto de três inteiros.

Respostas

2.2 a) Falso. Os comentários não causam nenhuma ação quando o programa executa. Eles são utilizados para documentar programas e melhoram sua legibilidade.

b) Verdadeiro.

c) Falso. Java diferencia letras maiúsculas de minúsculas, então essas variáveis são distintas.

d) Falso. O operador de resto também pode ser utilizado com operandos não inteiros em Java.

e) Falso. Os operadores `*`, `/` e `%` têm uma precedência mais alta que os operadores `+` e `-`.

Respostas

2.5

a)
```
// Calcula o produto de três inteiros
```

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

c)
```
int x, y, z, result;
```
ou
```
int x;
int y;
int z;
int result;
```

d)
```
System.out.print("Enter first integer: ");
```

e)
```
x = input.nextInt();
```

f)
```
System.out.print("Enter second integer: ");
```

g)
```
y = input.nextInt();
```

h)
```
System.out.print("Enter third integer: ");
```

i)
```
z = input.nextInt();
```

j)
```
result = x * y * z;
```

Respostas

```java
// Exercício 2.6: Product.Java
// Calcula o produto de três inteiros.
import java.util.Scanner; // programa utiliza Scanner

public class Product
{
   public static void main(String[] args)
   {
      // cria Scanner para obter entrada a partir da janela de comando
      Scanner input = new Scanner(System.in);

      int x; // primeiro número inserido pelo usuário
      int y; // segundo número inserido pelo usuário
      int z; // terceiro número inserido pelo usuário
      int result; // produto dos números

      System.out.print("Enter first integer: "); // solicita entrada
      x = input.nextInt(); // lê o primeiro inteiro

      System.out.print("Enter second integer: "); // solicita entrada
      y = input.nextInt(); // lê o segundo inteiro

      System.out.print("Enter third integer: "); // solicita entrada
      z = input.nextInt(); // lê o terceiro inteiro

      result = x * y * z; // calcula o produto dos números

      System.out.printf("Product is %d%n", result);
   } // fim do método main
} // fim da classe Product
```

Operadores Java

Operadores

Tipo	Operador	Propósito	Exemplo
Aritméticos	+	Adição	a = 4 + 1; // 5
	-	Subtração	a = 4 – 1; // 3
	*	Multiplicação	a = 2 * 4; // 8
	/	Divisão	a = 8 / 2; // 4
	%	Módulo (resto da divisão)	a = 5 % 2; // 1
Concatenação	+	Concatenação de Strings	String a = "Olá " + "Mundo";
Atribuição	=	Atribuição simples	a = 50;
Lógicos	&&	"e" lógico	(a > 1) && (b < 1)
	||	"ou" lógico	(a > 1) || (b < 1)
	!	não (inversão)	!(a > 2)
Condicionais (Comparação)	==	igualdade de valores ou endereços dos objetos.	(a == 0)
	!=	diferente de	(a != 0)
	<	menor que	(a < 0)
	>	maior que	(a > 0)
	<=	menor ou igual a	(a <= 0)
	>=	maior ou igual a	(a >= 0)
	instanceof	Verificação de tipo ()	//x é uma String?) (x instanceof String)
Incremento e **Decremento**	++	Incremento	a++;
	--	Decremento	a--;
Conversão	(tipo)	Conversão de tipo	int b = (int) 40.5;
Classe	new	Criação de objeto	Aluno a = new Aluno();

Comandos Java

Comando	Propósito	Sintaxe
Declaração de variável	Declaração de variável	`tipo nome_variavel = valor_inicial;`
Declaração de constante	Declaração de constante	`final tipo nome_constante = valor;`
Bloco	Marcar um bloco de cód.	`{ } //Abre e fecha chaves "{}"`
if	Comando condicional	`if (a > b) { System.out.println("A é maior que B"); } else { System.out.println("A é igual ou menor que B"); }`
switch	Comando condicional	`switch (i) { case 0 : System.out.println("ZERO"); break; case 1: System.out.println("UM"); break; case 2: System.out.println("DOIS"); break; }`
while	Laço com pré validação	`int i = 1; while (i <= 10) { System.out.println(i++); }`
do	Laço com pós validação	`int i = 1; do { System.out.println(i++); } while (i <= 10);`
for	Laço simplificado	`for (i=1;i<=10;i++){ System.out.println(i); }`
break	Saída de bloco	`break;`
continue	Reinício de bloco	`continue;`
return	Retorno de método	`return <valor ou objeto>;`

Tarefas

1. Leia 3 valores, no caso, variáveis A, B e C, que são as três notas de um aluno. A seguir, calcule a média do aluno, sabendo que a nota A tem peso 2, a nota B tem peso 3 e a nota C tem peso 5. Considere que cada nota pode ir de 0 até 10.0, sempre com uma casa decimal.

2. Faça um programa que leia o nome de um vendedor, o seu salário fixo e o total de vendas efetuadas por ele no mês (em dinheiro). Sabendo que este vendedor ganha 15% de comissão sobre suas vendas efetuadas, informar o total a receber no final do mês, com duas casas decimais.

3. Faça um programa que leia três valores e apresente o maior dos três valores lidos seguido da mensagem “O maior eh: x”.