



Faculdade de Design,
Tecnologia e Comunicação
ue Universidade Europeia

Conceitos Básicos de Programação

Fundamentos da Programação

Fernando Marson

fernando.marson@universidadeeuropeia.pt

Algoritmo

- Um algoritmo é um **conjunto não ambíguo e ordenado de instruções ou passos executáveis** que definem um **processo finito**.
- Se este conjunto de **instruções**, for **executado com as mesmas entradas**, sempre irá gerar a **mesma saída**.

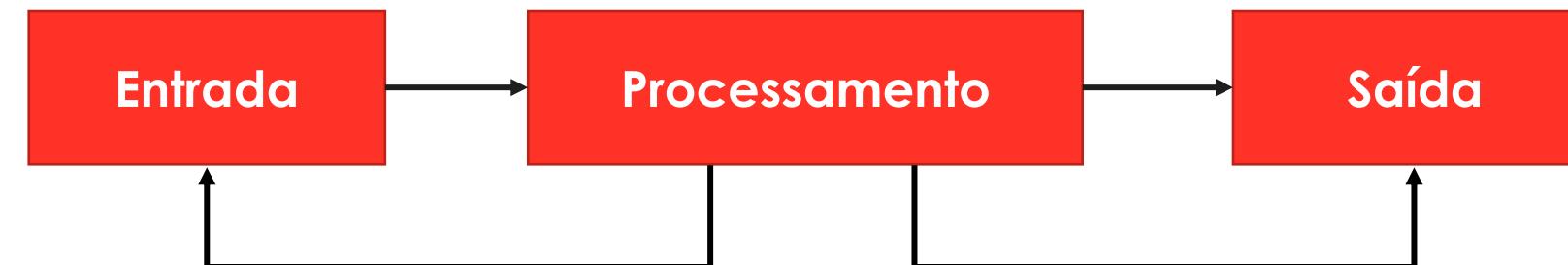
Algoritmo - Características

- Um algoritmo deve possuir as seguintes características:
 - **Definição clara:** Cada etapa precisa ser bem definida, sem margem para interpretações ou ambiguidades.
 - **Finitude:** O método deve sempre terminar em algum ponto, ou seja, deve ter um número finito de passos.
 - **Entradas e Saídas:** O algoritmo começa com entradas específicas e termina com uma solução para o problema.
 - **Eficiência:** Resolve o problema de forma eficiente, utilizando o mínimo de recursos (tempo, memória, etc.) possível.

Algoritmo - Funcionamento

Etapas de um algoritmo

- Entrada de dados
- Processamento de dados (operações)
- Saída de dados



Algoritmo - Representações

- **Descrição narrativa:** descrição da sequência de passos utilizando a língua nativa (português, inglês, espanhol).
- **Fluxograma:** representação utilizando símbolos gráficos predefinidos.
- **Pseudocódigo:** Une as principais características da descrição narrativa e fluxograma.

Descrição Narrativa

- A **descrição narrativa** explica as **etapas e processos** de um algoritmo em texto.
- Em vez de **listar código ou fórmulas**, ele **conta como o algoritmo funciona**, explicando de forma clara e lógica.
- Essa abordagem **ajuda tanto desenvolvedores quanto não desenvolvedores**, a **perceber a lógica** das soluções.

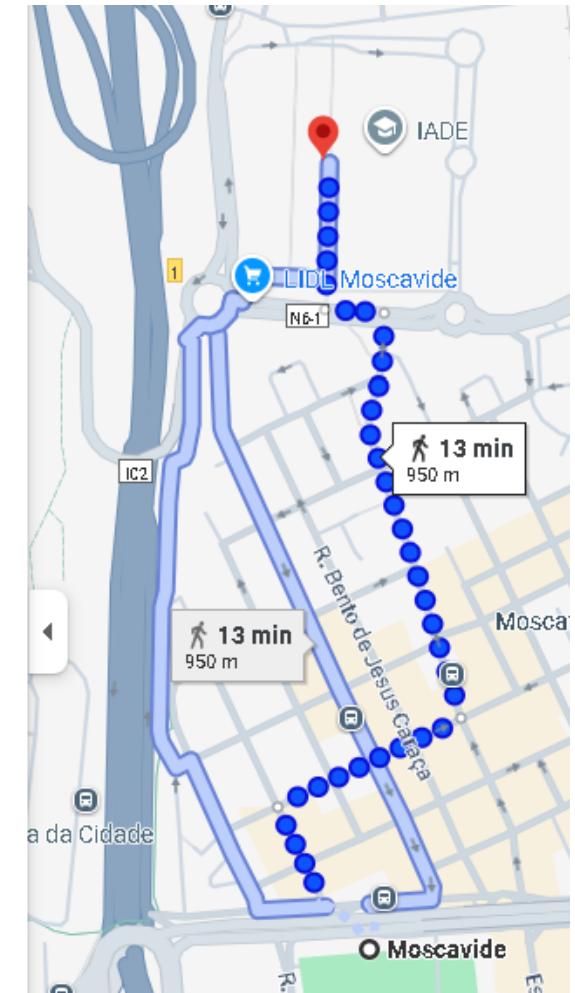
Descrição Narrativa

- **Clareza:** usar linguagem direta e evitar jargões técnicos.
- **Lógica sequencial:** apresentar as etapas em ordem cronológica (entrada → processamento → saída).
- **Exemplos e analogias:** empregar comparações do mundo real para tornar a explicação mais comprehensível e fácil de seguir.

Moscavide
1800-233 Lisboa

- ↑ Siga para norte em R. Alm. Gago Coutinho em direção a R. António Maria Pais
110 m
- Vire à direita em direção a R. Artur Ferreira da Silva
200 m
- ← Vire à esquerda em direção a Av. de Moscavide
400 m
- ← Vire à esquerda em direção a Av. de Moscavide/N6-1
59 m
- Vire à direita
O destino encontra-se à esquerda
150 m

Oriente Green Campus
R. Adão Manuel Ramos Barata 3, 1886-502 Moscavide



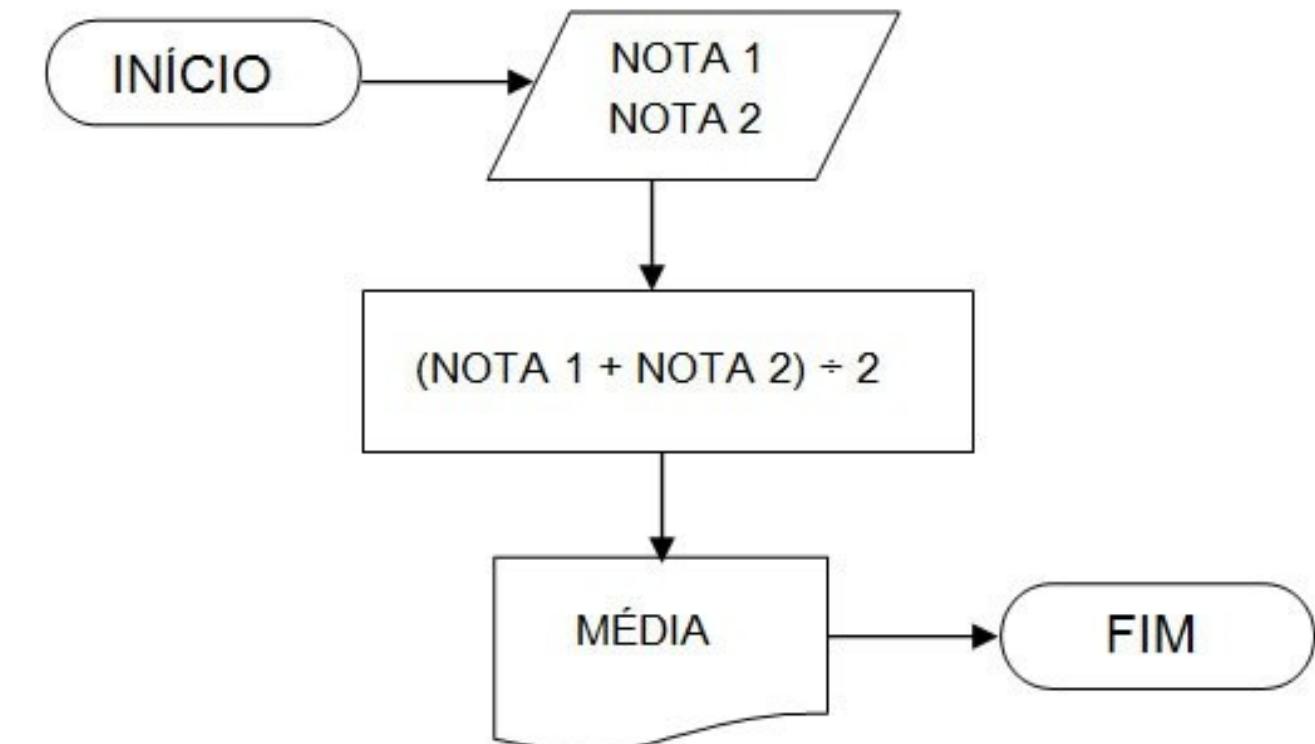
Fluxograma

- Um fluxograma é a **representação gráfica de um algoritmo**, usando **símbolos, formas e conexões com setas** para mostrar um **processo ou programa**.
- Com **fluxogramas**, podemos **perceber melhor um programa**.
- O **objetivo principal** é **simplificar as etapas** e tornar as mesmas **mais fáceis de executar**.

Fluxograma

- Existem vários **símbolos** padrão usados num fluxograma:

	Início ou fim do algoritmo
	Indica o sentido do fluxo de execução do algoritmo. Conecta os objetos gráficos
	Representa a entrada de dados
	Indica cálculos e atribuições de valores (processamento)
	Indica desvios ou tomadas de decisões (Por exemplo: SE isso, ENTÃO aquilo)
	Representa a saída de dados, no Portugol IDE
	Também representa a saída de dados



Pseudocódigo

- Pseudocódigo é uma **maneira simplificada de representar algoritmos**, usando **linguagem natural simplificada** para **planejar e organizar** as **etapas e ações** antes da implementação.
- Facilita a **compreensão** e a **estruturação lógica** do programa, permitindo que programadores descrevam processos e algoritmos de forma intuitiva.
- A sua **função principal** é auxiliar no **planejamento dos algoritmos**, otimizando o processo de desenvolvimento.

Pseudocódigo

```
var
    nota1 : real
    nota2 : real
    media : real

    inicio
        escrever ("Informe a nota 1: ")
        ler (nota1)
        escrever ("Informe a nota 2: ")
        ler (nota2)
        media <- (nota1 + nota2)/2
        escrever ("A sua média é ", media)

    finalgoritmo
```

Código-Fonte

- É o **conjunto de instruções escritas numa linguagem de programação** que um programador escreve para dizer ao computador **o que fazer**.
- A escrita desse **código-fonte segue as regras definidas pela linguagem de programação** escolhida.

Java

```
public class OláMundo {  
    public static void main(String[] args) {  
        System.out.println("Olá, mundo!");  
    }  
}
```

Python

```
print("Olá, mundo!")
```

C++

```
int main() {  
    std::cout << "Olá, mundo!" << std::endl;  
    return 0;  
}
```

C#

```
class OláMundo {  
    static void Main() {  
        Console.WriteLine("Olá, mundo!");  
    }  
}
```

Código-Fonte

```
var
nota1 : real
nota2 : real
media : real

inicio
    escrever ("Informe a nota 1: ")
    ler (nota1)
    escrever ("Informe a nota 2: ")
    ler (nota2)
    media <- (nota1 + nota2)/2
    escrever ("A sua média é ", media)
fimalgoritmo
```

```
1 import java.util.Scanner;
2 class Main {
3     public static void main(String[] args) {
4         Scanner input = new Scanner(System.in);
5         float nota1, nota2, media;
6
7         System.out.print("Informe a nota 1: ");
8         nota1 = input.nextFloat();
9         System.out.print("Informe a nota 2: ");
10        nota2 = input.nextFloat();
11        media = (nota1 + nota2) / 2;
12        System.out.print("A sua média é " + media);
13    }
14 }
```

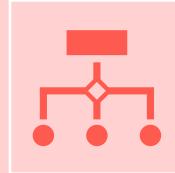
Arquivos Utilizados na Codificação

- **.java** – código-fonte em Java

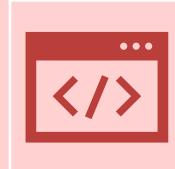
- **Exemplos**
 - Carro.java
 - Pessoa.java

- Normalmente identificamos o arquivo contendo a **parte principal de um programa** como **Main**:
Main.java

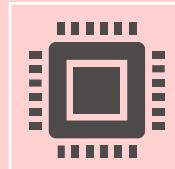
Conceitos de algoritmo, código-fonte e programa



Algoritmo é o **modelo lógico** para a resolução de um programa.



Código-fonte é a formalização de um **algoritmo em uma linguagem de programação**.



Programa é um conjunto de instruções em **código de máquina** que determinam o que o computador deve realizar.

Comentários em Java

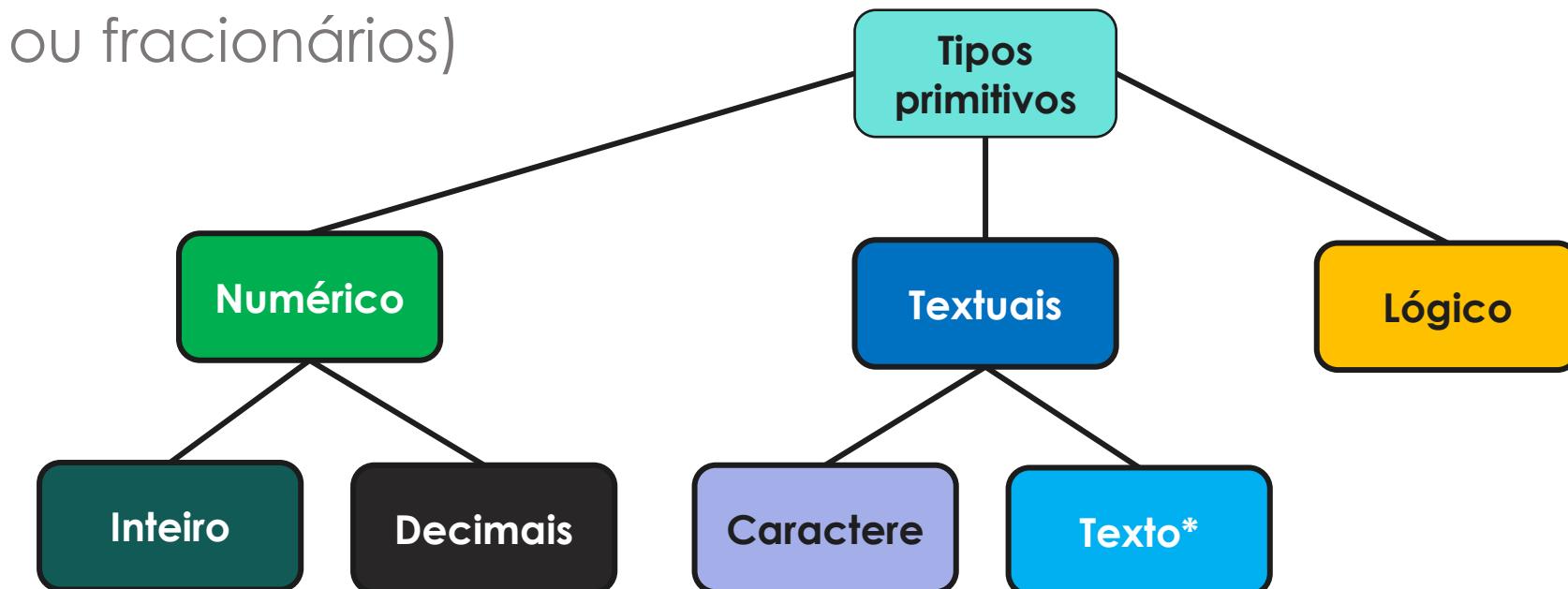
Os comentários em **Java** podem ser:

- de uma única linha usando `//` no início dela
- de várias linhas usando `/*` no começo da primeira linha e depois usando `*/` no começo da última linha.

```
// comentário de uma linha só  
  
// outro comentário de uma linha  
  
/*  
Primeira linha do comentário  
  
Este é um comentário de várias linhas!  
Coloque aqui quantas linhas precisares.  
  
Última linha do comentário  
*/
```

Tipos de Dados

- **Variáveis** podem armazenar dados de diferentes tipos, e esses tipos definem que operações podem ser executadas com ou sobre eles.
- **Tipos de dados primitivos:**
 - **Numérico** (inteiro e decimais ou fracionários)
 - **Textual** (caractere ou texto*)
 - **Lógico** (booleano)



*Um texto não é tipo primitivo, mas possui um comportamento próximo ao de um tipo primitivo em algumas linguagens de programação.

Tipos de Dados

- **Inteiros:** números que **não possuem** parte fracionária e podem ser tanto **positivos** quanto **negativos**.
- **Decimais:** números que **possuem** parte fracionária e podem ser tanto **positivos** quanto **negativos**.
- **Textuais:** uma ou mais letras, dígitos ou símbolos especiais, delimitados por aspas simples (**char**) ou aspas duplas (**String**).
- **Lógico:** possuem apenas dois valores possíveis, **falso (false)** ou **verdadeiro (true)**.

Tipos de Dados em Java

```
// Inteiros                                // Textuais
int numero, quantidade, valor;           String frase, texto;
                                         char letra, simbolo;

numero = 35;
quantidade = 0;
valor = -84;

                                         frase = "Esta é uma frase.";
                                         texto = "Não funciona com aspas simples";
                                         letra = 'A';
                                         simbolo = '&';

// Decimais                                 // Lógicos
float preco;
double temperatura;

                                         boolean ligado, pago;

preco = 20.65;
temperatura = -0.5;
```

Linguagens de Programação - Tipagem



- Linguagens fracamente tipadas
 - permite operações entre diferentes tipos de dados sem a necessidade de conversão explícita ou declaração rígida de tipos.



- Linguagens fortemente tipadas
 - as variáveis têm um tipo bem definido e precisam ter esse tipo informado no momento de sua declaração. O tipo não pode ser alterado durante a execução do programa.



- Linguagens dinamicamente tipadas
 - não é preciso definir o tipo das variáveis ao declará-las, sendo possível alterá-las em tempo de execução.

Variáveis

- **Variáveis são posições de memória** referenciadas no programa. Elas **armazenam informações voláteis** (temporárias) de um determinado **tipo de dado**.
- Em **Java** a **variável é declarada como sendo de um determinado tipo** e permanece como sendo desse tipo até o final do código, ou seja, **não pode mudar**.

Constantes

- Da mesma forma que uma variável, uma **constante também é uma referência para um espaço de memória**, mas, **após o seu valor ter sido definido**, o mesmo **não pode mais ser modificado**.
- Em **Java**, a declaração de uma constante normalmente é feita adicionando a palavra reservada **final** antes do tipo de dado da constante.

```
1 final char CARACTERE = '#';
2 final int QTDMAXIMA = 50;
3 final double PI = 3.14159265358979323846;
```

Descobrindo o tipo do dado

- Em Java podemos descobrir o tipo do dado através da seguinte chamada:

((Object) variavel).getClass().getSimpleName()

```
int idade = 5;
float valor = 2.1f;
boolean ligado = true;
String texto = "Teste";

System.out.println(((Object)idade).getClass().getSimpleName());
System.out.println(((Object)valor).getClass().getSimpleName());
System.out.println(((Object)texto).getClass().getSimpleName());
System.out.println(((Object)ligado).getClass().getSimpleName());
```

Integer
Float
String
Boolean

Padronização de Nomenclatura

- Um padrão de nomenclatura em programação, significa um **conjunto de convenções a serem seguidas** quando se decide o nome de seus identificadores, como variáveis, constantes, métodos, classes, etc.
- O uso de um padrão **não é obrigatório**, mas **é recomendado**, pois **facilita a leitura do código** tanto para você quanto para outros programadores.
- **Existem convenções de nomenclatura gerais** e outras que são aplicadas em uma linguagem de programação específica como, por exemplo, em **Java**.

snake_case

A nomenclatura **snake_case** tem o nome é derivado da semelhança com um animal: a **cobra**.

- Ao **aglutinarmos palavras** para representar o nome do identificador, as palavras são separadas por um símbolo de sublinhado ou underline.

```
quantidade_de_itens  
valor_mercadoria  
preco_total
```



snake_case

SCREAMING_SNAKE_CASE

O padrão **snake_case** apresenta uma variante, onde todas as letras são capitalizadas, chamado **SCREAMING_SNAKE_CASE**.

- Em **Java**, este padrão é utilizado para declarar **constantes**, dados que **não podem ser alterados após terem sido definidos**.

```
VALOR_DE_PI  
QUANTIDADE_MESES_ANO  
NUM_MAX_DE_TENTATIVAS  
TAXA_DE_JUROS_FIXA
```



SCREAMING_SNAKE_CASE

camelCase

- A nomenclatura **camelCase** é uma das mais utilizadas na programação. O nome vem semelhança do padrão com o **formato das corcovas dos camelos**.
- Ao aglutinarmos palavras para representar o nome do identificador, **a primeira letra de cada palavra fica capitalizada, com exceção da primeira palavra**. Por exemplo, **valorTotal** ou **valorTotalDaCompra**.



CamelCase

- Existe uma variação, onde **todas as primeiras letras das palavras são capitalizadas**, também chamada de **CamelCase**.
- Então chamamos normalmente o primeiro caso somente de **camelCase** ou de **lowerCamelCase** e o segundo de **PascalCase** ou **UpperCamelCase**.



lower**CamelCase**



PascalCase ou
UpperCamelCase

Boas Práticas

- Toda **variável** só pode conter **letras, dígitos** e sublinhado.
- **Todo primeiro caractere** deve ser sempre uma **letra** ou sublinhado.
- As letras podem ser **minúsculas** ou **maiúsculas**, mas sempre **sem acento**.
- Letras **maiúsculas** e **minúsculas** são consideradas **caracteres diferentes**
 - **variavel1** é diferente de **Variavel1**
- **Palavras reservadas** não podem ser usadas como nome de variáveis.
 - Por exemplo:
for, while, do, switch, if,

Boas Práticas

- Utilizar **nomes sugestivos**:
 - soma
 - pesoLiquido
 - valorMaximo
- Variáveis com nomes **não representativos** ou **curtos**, com apenas **uma letra**, como:

a, b, c, x, y, z, i, j

devem ser utilizadas em **contextos restritos**.

Boas Práticas - Java

- Utilizar a notação **camelCase** para **variáveis**.

```
valor  
preco  
quantidadeDeItens  
valorMercadoria  
precoTotal
```

- Utilizar a notação **SCREAMING_SNAKE_CASE** para **constantes**.

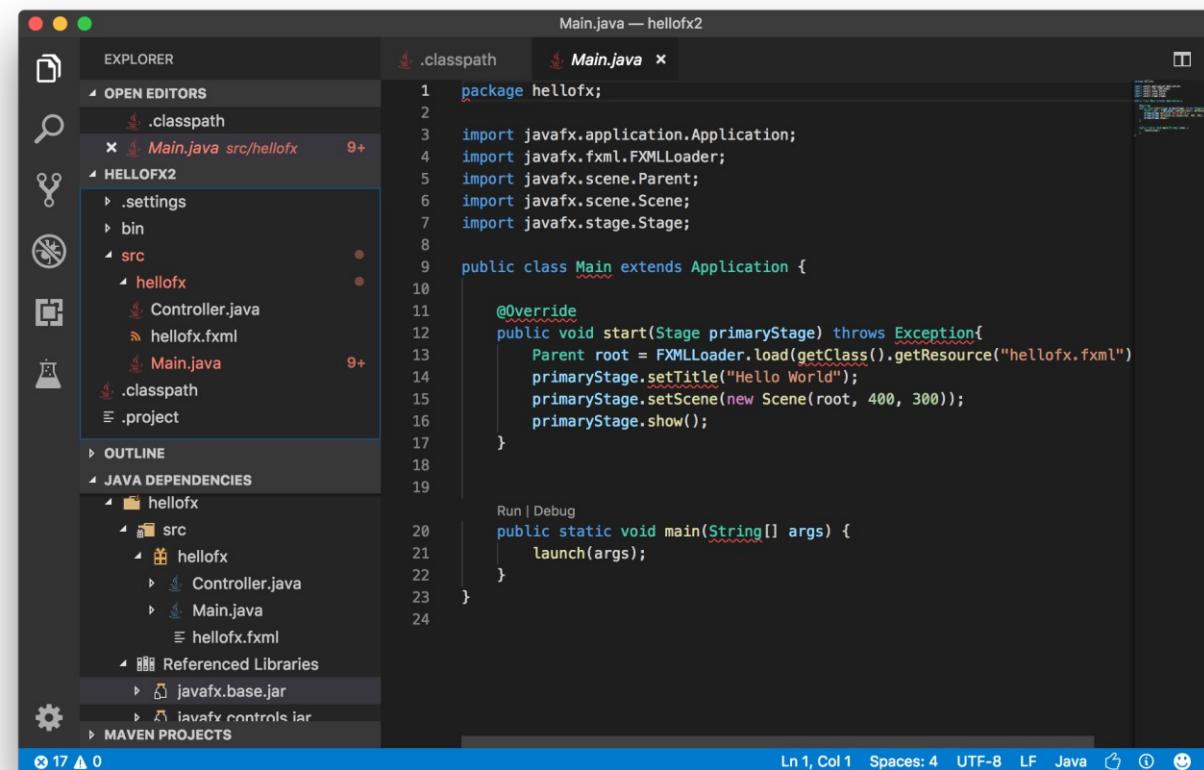
```
VALOR_DE_PI  
QUANTIDADE_MESES_ANO  
NUM_MAX_DE_TENTATIVAS  
TAXA_DE_JUROS_FIXA
```

- Utilizar a notação **PascalCase** para **classes**.

```
Main  
Personagem2D  
FiguraGeometrica
```

IDE - Integrated Development Environment

Uma IDE é um ambiente de desenvolvimento integrado, um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.



IDE - Integrated Development Environment



VSCode



NetBeans



Eclipse



IntelliJ

IDE - Integrated Development Environment



VSCode

A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with a project named 'HELLOFX2'. The main editor window displays Java code for a JavaFX application, specifically a Main.java file. The code initializes a stage and loads a FXML file. The bottom status bar indicates the code is in Java mode.

```
1 package hellofx;
2
3 import javafx.application.Application;
4 import javafx.fxml.FXMLLoader;
5 import javafx.scene.Parent;
6 import javafx.scene.Scene;
7 import javafx.stage.Stage;
8
9 public class Main extends Application {
10
11     @Override
12     public void start(Stage primaryStage) throws Exception{
13         Parent root = FXMLLoader.load(getClass().getResource("hellofx.fxml"));
14         primaryStage.setTitle("Hello World");
15         primaryStage.setScene(new Scene(root, 400, 300));
16         primaryStage.show();
17     }
18
19
20     public static void main(String[] args) {
21         launch(args);
22     }
23 }
```

[Download Visual Studio Code - Mac, Linux, Windows](#)



IntelliJ

A screenshot of the IntelliJ IDEA interface. The left sidebar shows a project structure with a 'src' folder containing Java files. The main editor window shows Java code for a 'NewProjectAction' class, which extends 'AnAction'. The code handles actions like creating a new project wizard and updating icons. The bottom status bar indicates the code is in Java mode.

```
1 import ...
2
3 public class NewProjectAction extends AnAction implements DumbAware, NewProjectOrModuleAction {
4     @Override
5     public void actionPerformed(@NotNull AnActionEvent e) {
6         NewProjectWizard wizard = new NewProjectWizard(null, ModulesProvider.EMPTY_MODULES_PROVIDER,
7             NewProjectUtil.createNewProject(wizard));
8     }
9
10    @Override
11    public void update(@NotNull AnActionEvent e) {
12        updateActionIcon(e);
13        updateActionText(this, e);
14    }
15
16    @Override
17    public @NotNull ActionUpdateThread getActionUpdateThread() { return ActionUpdateThread.BGT; }
18
19    private static void updateActionIcon(@NotNull AnActionEvent e) {
20        if (NewWelcomeScreen.isNewWelcomeScreen(e)) {
21            NewWelcomeScreen.updateNewProjectIconIfWelcomeScreen(e);
22        }
23        else if (ExperimentalUI.isNewUI() && ActionPlaces.PROJECT_WIDGET_POPUP.equals(e.getPlace())) {
24            e.getPresentation().setIcon(ExpUiIcons.General.Add);
25        }
26    }
27
28 }
```

[Download IntelliJ - Mac, Linux, Windows](#)

Cadastrros Importantes

Links para registo de benefícios:

- [JetBrains Student Pack](#)

- [Github Education](#)

- [Office 365 Education](#)