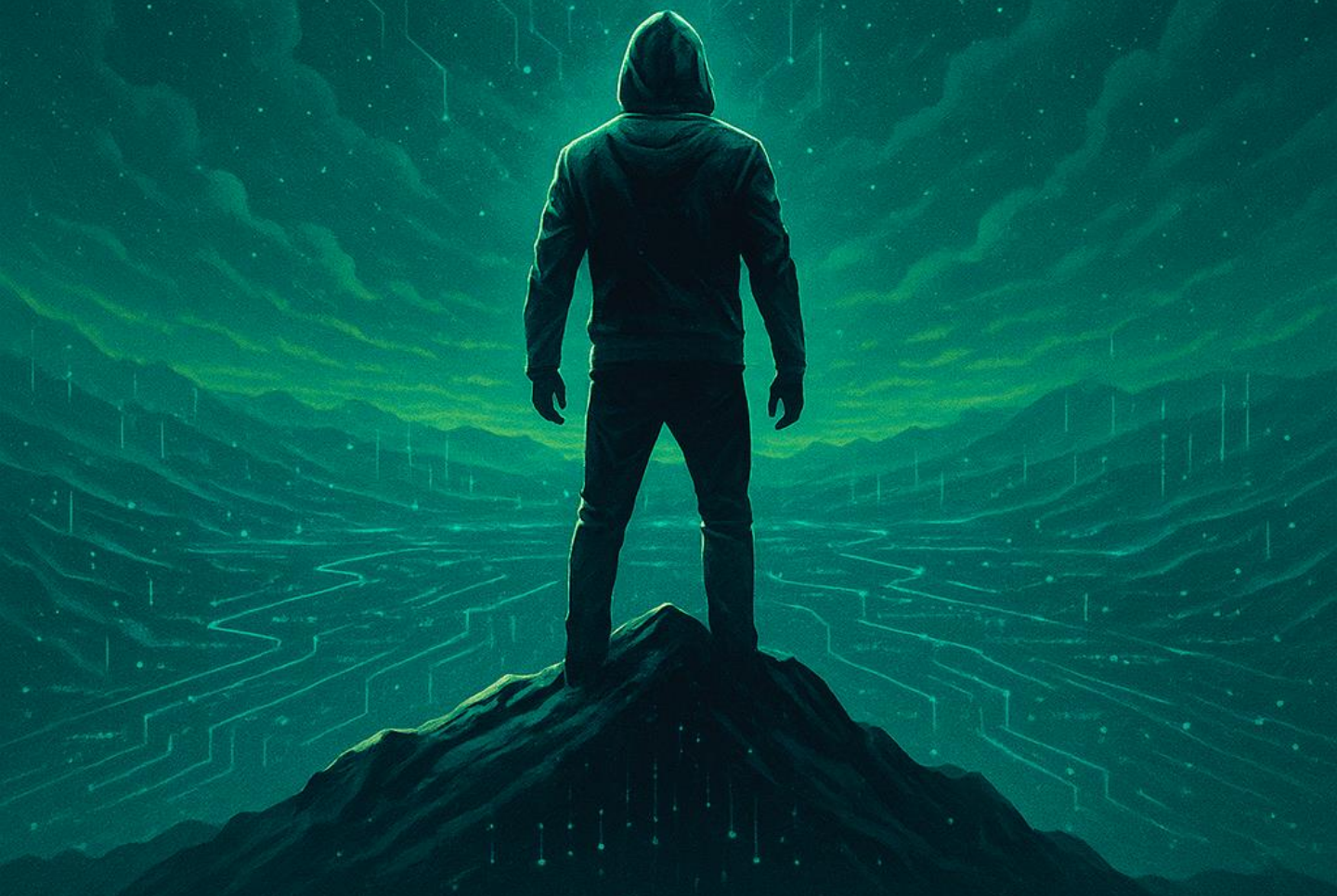
A JORNADA JAVA

O CÓDIGO DESPERTA



```
public static void main(
```

Kauan Miguel

01

POR QUE APRENDER JAVA HOJE?

Java é uma das linguagens de programação mais utilizadas e respeitadas no mundo da tecnologia. Mesmo após mais de duas décadas de existência, ela continua atual, poderosa e indispensável em diversos setores.

Neste capítulo, você vai entender por que o Java **continua sendo uma ótima escolha**, tanto para quem está começando quanto para quem quer construir uma carreira sólida em tecnologia.

A História do Java

Java foi criado em 1995 pela Sun Microsystems com um grande diferencial: escreva uma vez, rode em qualquer lugar. Essa proposta revolucionária deu origem a aplicações portáteis, compatíveis com diferentes sistemas operacionais. Em 2010, a Oracle adquiriu o Java, que segue evoluindo e se adaptando às novas demandas do mercado.

Onde o Java é usado hoje?

Java está presente em:

- Aplicações bancárias (sistemas seguros e robustos)
- Serviços web e APIs (back-end de grandes sites e plataformas)
- Aplicativos Android (é a linguagem base do Android)
- Sistemas corporativos e ERPs
- IoT (Internet das Coisas) e dispositivos embarcados
- Jogos e simulações (como Minecraft, originalmente feito em Java)

Por que o Java ainda é relevante?

Aqui vão motivos reais para aprender Java hoje:

- Alta demanda no mercado de trabalho: Java está entre as linguagens mais procuradas por empresas no Brasil e no exterior.
- Usado por grandes empresas: Amazon, Netflix, Google, IBM e bancos gigantes utilizam Java.
- Boa performance e segurança: Ideal para aplicações críticas.
- Comunidade ativa e vasto material de apoio: Milhares de fóruns, tutoriais, bibliotecas e cursos disponíveis.
- Base para Android: Se quiser programar para celular, Java é essencial.

Java é bom para iniciantes?

Sim! Java tem uma sintaxe estruturada e clara, e ao mesmo tempo oferece profundidade para crescer junto com o programador. Muitos cursos de ciência da computação começam com Java justamente por esse equilíbrio entre simplicidade e poder.

Exemplo prático: Onde o Java aparece no mundo real?

Imagine um sistema bancário onde você consulta saldo, transfere valores e gera extratos. Toda essa lógica — validação, segurança, cálculos — pode ser feita em Java, rodando em servidores robustos e acessados por diversos canais (web, app, caixas eletrônicos).

02

SEU PRIMEIRO CÓDIGO: “HELLO, JAVA!”

Antes de começar a programar de fato, você precisa preparar o ambiente de desenvolvimento e entender como o Java funciona na prática. Vamos criar o seu primeiro programa passo a passo.

Instalando e configurando seu ambiente

Você pode usar o VS Code com a extensão de Java ou o IntelliJ IDEA, que é mais completo para projetos Java.

Passos básicos:

1. Instale o JDK (Java Development Kit) – de preferência a versão 17 ou superior.
2. Baixe e instale o VS Code ou o IntelliJ IDEA Community.
3. No VS Code, instale as extensões:

    - Extension Pack for Java (da Microsoft)
    - Debugger for Java
    - Java Test Runner

Criando o primeiro programa

Hello Java

```
1  classe pública HelloJava {
2      public static void main(String[] args) {
3          System.out.println("Olá, Java!");
4      }
5  }
```

Explicação do código

`public class HelloJava` → Define a classe principal.

`public static void main` → É o método que o Java executa primeiro.

`System.out.println` → Imprime uma mensagem no terminal.

Como compilar e rodar o código

No terminal:

```bash
javac HelloJava.java
java Olá Java
```

Saída esperada:

```css
Olá, Java!
```

03

VARIÁVEIS E TIPOS DE DADOS DESCOMPLICADOS

Variáveis guardam dados que seu programa usa para funcionar. Aqui, você vai aprender os principais tipos de dados em Java e como usá-los na prática.

Tipos primitivos em Java

- int – números inteiros
- double – números com casas decimais
- char – um único caractere
- boolean – verdadeiro ou falso
- String – sequência de caracteres (não é primitivo, mas muito usado)

Declarando variáveis

Java

```java
int idade = 25;
altura dupla = 1.75;
char sexo = 'M';
boolean ativo = true;
String nome = "João";
```

Entrada de dados com Scanner

Java

```java
importar java.util.Scanner;

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

        System.out.print("Digite seu nome: ");
        String nome = scanner.Próxima linha();

 System.out.println("Olá, " + nome + "!");
 scanner.fechar();
    }
}
```

04

CONTROLE DE FLUXO: TOME DECISÕES COM SEU CÓDIGO

Com estruturas de controle, seu programa pode tomar decisões e repetir tarefas. Isso torna seu código dinâmico e poderoso.

Decisões com if, else e switch

Java

```java
int idade = 18;
if (idade ⩾ 18) {
    System.out.println("Você é maior de idade.");
} mais {
    System.out.println("Você é menor de idade.");
}
```

Switch

Java

```java
int opção = 2;
interruptor(opção) {
    caso 1:
        System.out.println("Novo Jogo");
        quebrar;
    Caso 2:
        System.out.println("Carregar Jogo");
        quebrar;
    inadimplência:
        System.out.println("Sair");
}
```

Repetição com for, while, do-while

```java
durante (int eu = 0; i < 5; i++) {
    System.out.println("Repetição número: " + i);
}
```

05

ENTENDENDO A PROGRAMAÇÃO ORIENTADA A OBJETOS

A Programação Orientada a Objetos (POO) é um pilar do Java. Ela permite que você modele o mundo real em forma de classes, objetos, atributos e métodos. Dominar POO torna seu código mais organizado, reutilizável e fácil de manter.

Conceitos fundamentais

- Classe: molde que define atributos e comportamentos.
- Objeto: instância de uma classe.
- Atributos: características (variáveis).
- Métodos: ações (funções).

Criando uma classe simples

Java

```java
público classe Pessoa {
    Corda nomo;
    int idade;

      void apresentar() {
 System.out.println("Olá, meu nome é " + nome + " e tenho " + idade + " anos.");
    }
}
```

Instanciando objetos

```java
público classe Principal {
    público static void main(String[] args) {
        Pessoa pág. 1 = novo Pessoa();
 p1.nome = "Carlos";
        p1.idade =  void 28;
 pág. 1.apresentar();
    }
}
```

Encapsulamento, herança e polimorfismo

- Encapsulamento: proteger os dados (uso de private e getters/setters)
- Herança: uma classe pode herdar outra (ex: Funcionario extends Pessoa)
- Polimorfismo: um método se comporta de formas diferentes em classes diferentes

06

LIDANDO COM ERROS COMO UM PROFISSIONAL

Em qualquer linguagem, erros acontecem. Com o Java, você pode tratar exceções de forma organizada para evitar que seu programa "quebre" inesperadamente.

try, catch e finally

Java

```java
tentar {
    int resultado = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Erro: divisão por zero.");
} finalmente {
    System.out.println("Operação finalizada.");
}
```

Exceções comuns

- NullPointerException
- ArrayIndexOutOfBoundsException
- IOException

Leitura de arquivo com tratamento

Java

```
importar java.io.*;

público classe Leitor {
    público estático   void main(String[] args) {
        tentar {
            Leitor Bufferizado Br = new BufferedReader(new FileReader("dados.txt"));
            String linha;
            while ((linha = br.readLine()) =/= null) {
                System.out.println(linha);
            }
 Br.fechar();
        } catch (IOException e) {
            System.out.println("Erro ao ler o arquivo.");
        }
    }
}
```

07

COLEÇÕES EM JAVA NO DIA A DIA

Coleções são estruturas que permitem guardar e manipular múltiplos elementos de forma eficiente. São muito utilizadas em projetos reais.

Principais coleções

- ArrayList – lista dinâmica
- HashMap – pares chave/valor
- HashSet – conjunto sem duplicatas

Exemplo com ArrayList

```java
importar java.util.ArrayList;

público classe ListaNomes {
    público estático   void main(String[] args) {
        ArrayList<String> nomes = new ArrayList<>();
        nomes.adicionar("Ana");
        nomes."Joãoadicionar(");
        durante (Corda nomo : nomes) {
            System.out.println(nome);
        }
    }
}
```

Exemplo com HashMap

Java

```
importar java.util.HashMap;

público classe Agenda {
    público estático   void main(String[] args) {
        Mapa de hash<String, String> contatos = new Mapa de hash<>();
        contatos.pôr("João", "9999-1234");
        System.out."Telefone do João: println(" + contatos.get("João"));
    }
}
```

08

TRABALHANDO COM ARQUIVOS

Ler e escrever em arquivos permite armazenar dados entre execuções. Você pode criar históricos, registros, salvar usuários etc.

Escrevendo em arquivos .txt

Java

```java
importar java.io.FileWriter;
importar java.io.IOException;

público classe Escritor { void
    público estático vazio principal(Argumentos de string[] ) {
        tentar {
            = Gravador de arquivos escritor novo FileWriter("registro.txt");
 escritor.write("Nome: Lucas\nIdade: 30");
 escritor.fechar();
            System.out.println("Arquivo criado com sucesso.");
        } catch (IOException e) {
            System.out.println("Erro ao escrever.");
        }
    }
}
```

09

JAVA COM INTERFACE GRÁFICA (SWING)

Com bibliotecas como Swing, é possível criar janelas, botões e caixas de texto em Java — ideal para programas mais interativos.

Criando uma janela simples

Java

```java
importar javax.swing.*;

aula pública Janela {
    público estático vazio void  main(String[] args) {
        JFrame moldura = Novo JFrame("Minha Janela");
        JLabel etiqueta = new JLabel("Olá, mundo gráfico!");
 moldura.adicionar(rótulo);
 moldura.tamanho do conjunto(300, 100);
 moldura.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 moldura.setVisible(verdadeiro);
    }
}
```

Exemplo: mini calculadora

```java
Pode ser expandido depois com layout e botões
```

10

BOAS PRÁTICAS E PRÓXIMOS PASSOS

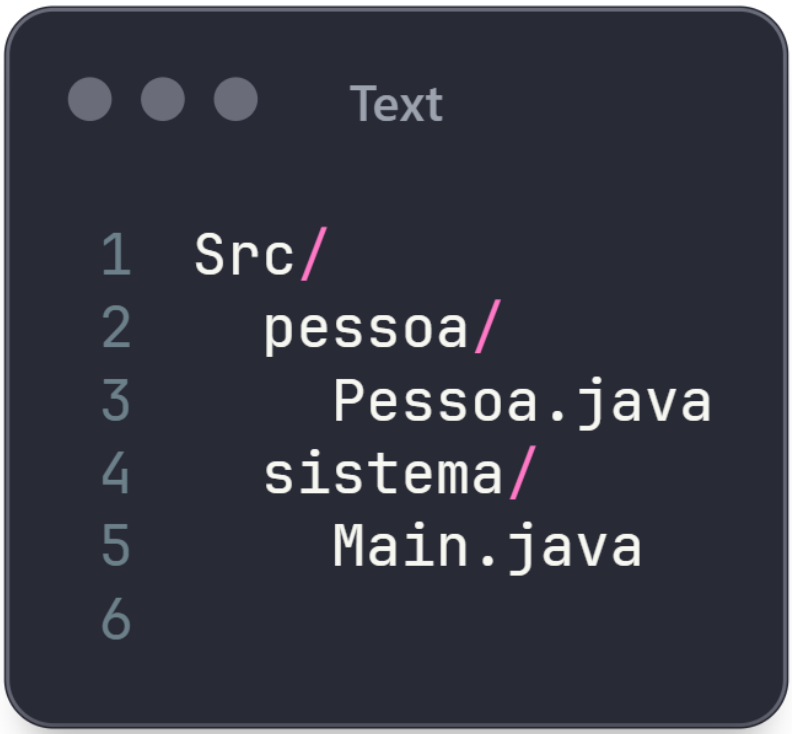
A qualidade do código importa tanto quanto ele funcionar. Aqui vão dicas para escrever códigos melhores e caminhos para continuar aprendendo.

Convenções de código

- Classes com letra maiúscula: Pessoa
- Variáveis com letra minúscula: idade, nome
- Use nomes claros e objetivos

Organização de pacotes

Agrupe arquivos relacionados:



Próximos passos

- Aprender Java Web (Servlets, Spring Boot)
- Conectar com banco de dados (JDBC, JPA, Hibernate)
- Criar APIs RESTful
- Contribuir em projetos open source
- Participar de comunidades no GitHub, Discord e fóruns

AGRADECIMENTOS

Agradecimentos Finais

Quero expressar minha sincera gratidão a você, leitor, que decidiu embarcar nesta jornada de aprendizado. Escolher aprender uma linguagem de programação é um ato de coragem, disciplina e determinação, e fico feliz em ter contribuído, de alguma forma, no seu desenvolvimento.

Agradeço também à comunidade Java, que, ao longo das décadas, se mantém ativa, colaborativa e apaixonada por compartilhar conhecimento. É graças a essa comunidade vibrante que o Java continua sendo uma das linguagens mais robustas, versáteis e relevantes do mercado.

Um agradecimento especial a todos os desenvolvedores, professores, mentores e entusiastas que inspiram diariamente aqueles que estão começando, provando que a tecnologia pode transformar vidas, abrir portas e construir futuros.

Se este material te ajudou, lembre-se de que o aprendizado não termina aqui. Continue explorando, errando, acertando e, principalmente, compartilhando o que você aprende com outras pessoas. O mundo da programação é muito mais poderoso quando cresce em comunidade.

Muito sucesso na sua jornada como programador(a) Java!