ByteCraft: Forjado em Java

O caminho do dev começa com uma linha de código

Lucas J. Da Cunha

Forjando a Base

Sua Jornada Começa Aqui

Se você está começando a programar ou quer dominar uma das linguagens mais usadas do mundo, este eBook foi feito para você.

Aqui, vamos falar sobre Java — uma linguagem robusta, orientada a objetos, que está presente em bancos, aplicativos Android, sistemas corporativos e até em dispositivos embarcados.

Mas afinal, o que é Java?

Java é uma linguagem de programação criada com o objetivo de ser simples, segura e multiplataforma. Isso significa que o mesmo código pode rodar em diferentes sistemas operacionais (como Windows, Linux e macOS) sem precisar ser reescrito.

Além disso, é uma das linguagens mais exigidas pelo mercado, com uma comunidade enorme e vasto material de apoio.

O que você vai encontrar neste eBook

Capítulo 1 – Estruturas de Dados em Java

Você vai aprender como guardar, organizar e acessar informações usando ferramentas como ArrayList, HashMap e HashSet, com exemplos reais para aplicar no dia a dia.

Capítulo 2 – Programação Orientada a Objetos (POO)

Aqui você entenderá os pilares da POO de forma descomplicada: como criar classes, usar construtores, proteger dados com encapsulamento e muito mais.

Capítulo 3 – Herança e Polimorfismo

Neste último capítulo, vamos explorar como tornar seu código mais reutilizável e flexível usando conceitos avançados como herança e polimorfismo, também com exemplos práticos e diretos ao ponto.

01

Trabalhando com Estruturas de Dados

ArrayList

Guardando dados de forma dinâmica

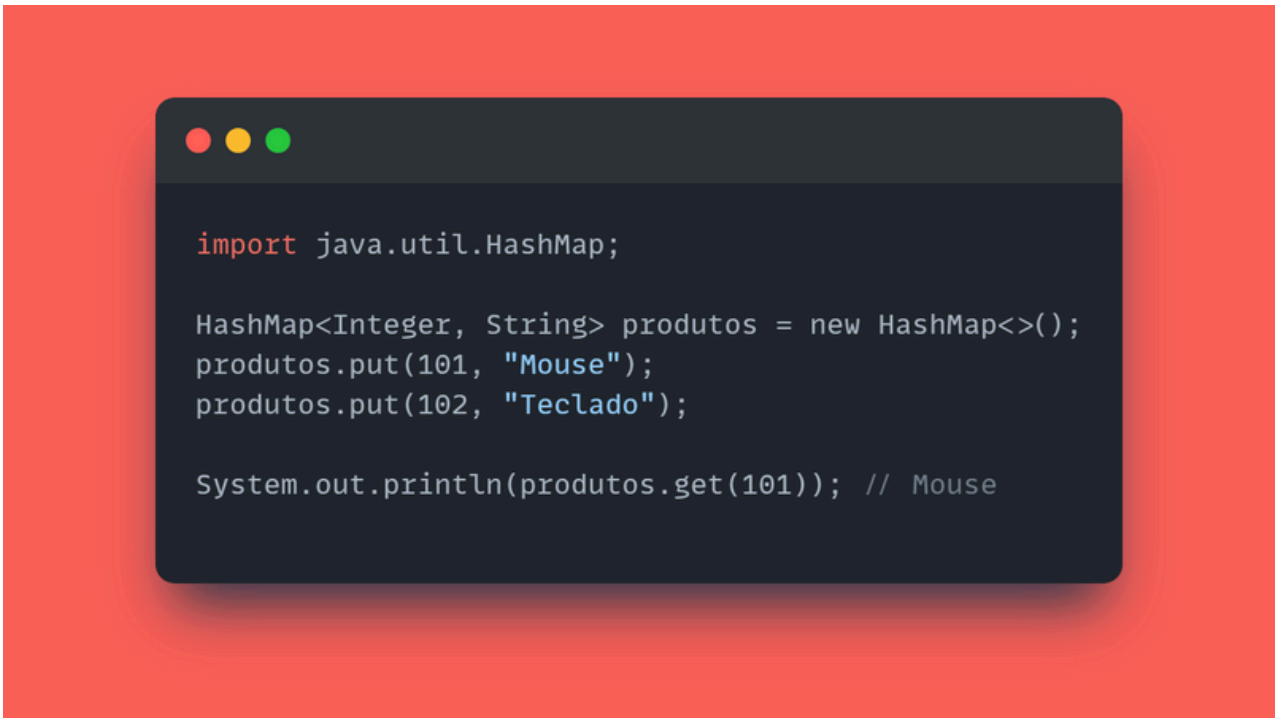
Uma ArrayList é como uma lista de tarefas: você pode adicionar, remover ou acessar elementos conforme precisar. É mais flexível que um array tradicional, pois seu tamanho pode crescer dinamicamente.

```java
import java.util.ArrayList;

ArrayList<String> tarefas = new ArrayList<>();
tarefas.add("Estudar Java");
tarefas.add("Ler um livro");
tarefas.add("Fazer exercícios");

for (String t : tarefas) {
    System.out.println("- " + t);
}
```

HashMap

Associando valores com etiquetas

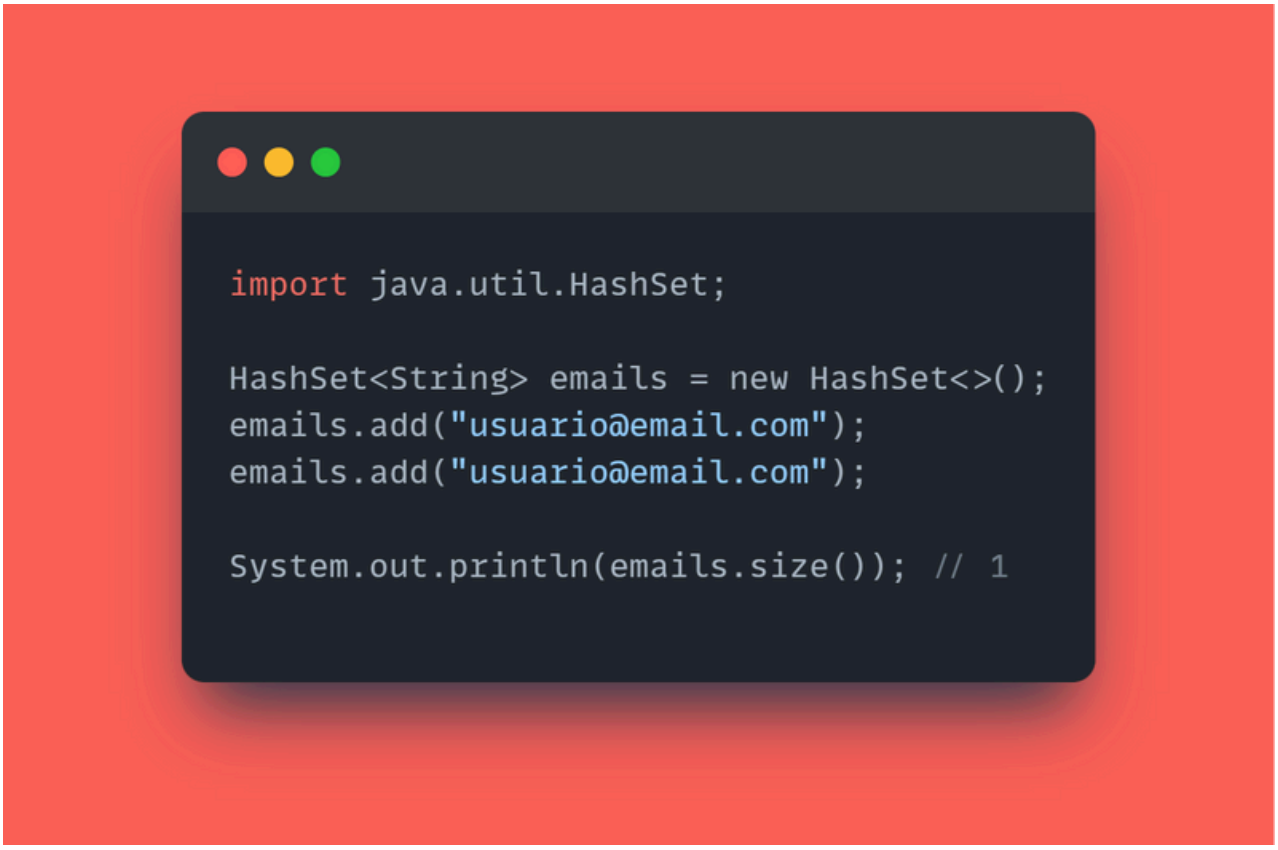
Com um HashMap, você pode relacionar dois valores — por exemplo, o código de um produto e seu nome. Você acessa os dados usando a chave, como se fosse um índice personalizado.



HashSet

Guardando valores únicos

O HashSet é ideal quando você precisa garantir que não haverá repetições em uma lista. Se tentar adicionar o mesmo item duas vezes, ele será ignorado automaticamente.



02

Entendendo a Programação Orientada a Objetos (POO)

Entendendo a Programação Orientada a Objetos (POO)

Criando classes e objetos

Em POO, criamos objetos baseados em "modelos" chamados de classes. Isso facilita a organização do código, principalmente em sistemas maiores. Cada objeto tem atributos (características) e métodos (ações).

```java
public class Pessoa {
    String nome;
    int idade;

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

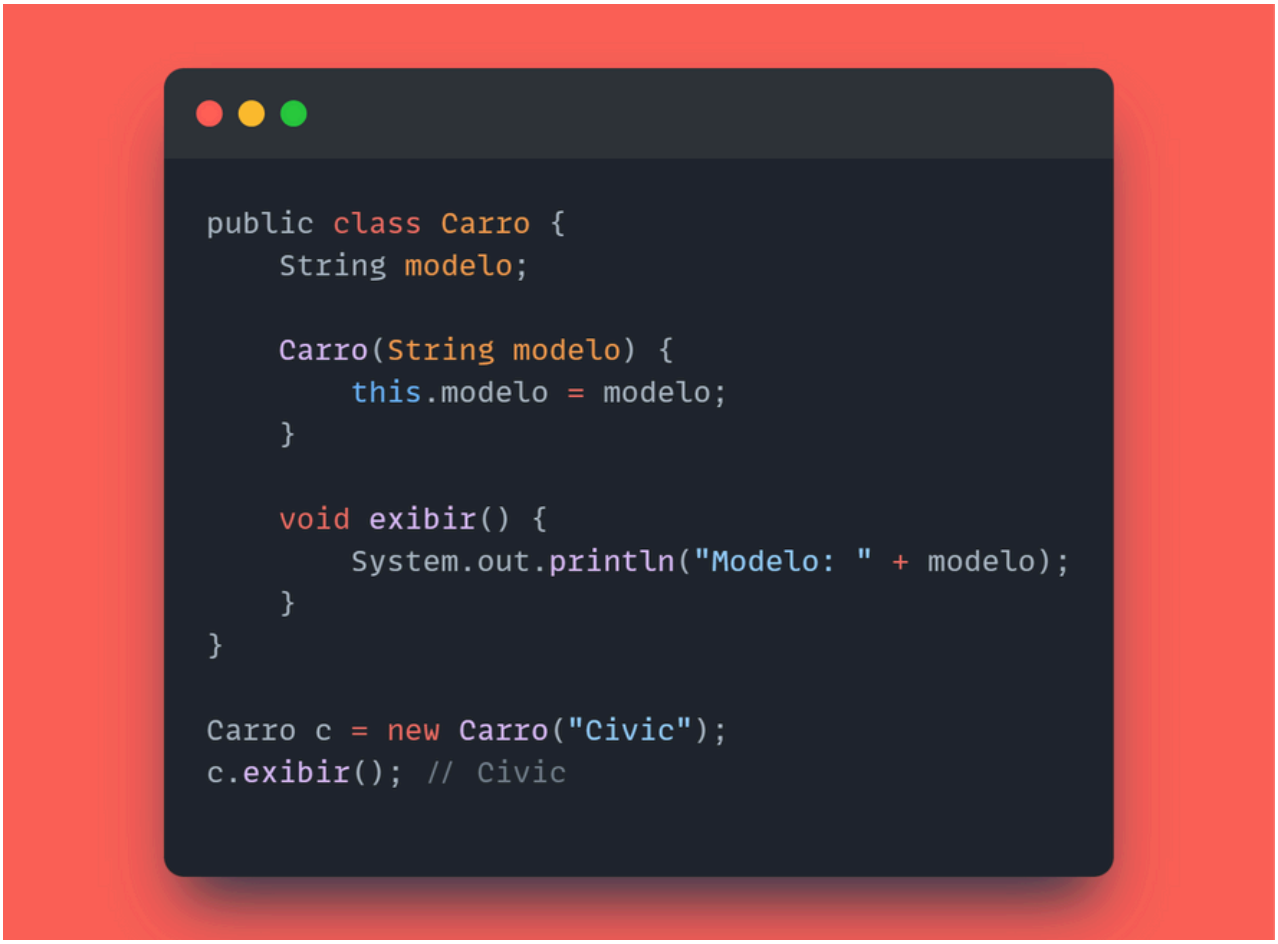
Uso:

```java
Pessoa p = new Pessoa();
p.nome = "Lucas";
p.idade = 22;
p.apresentar();
```

Construtores

Criando objetos com valores prontos

Um construtor é um método especial que define os valores iniciais de um objeto. Ele é chamado automaticamente quando você usa new.

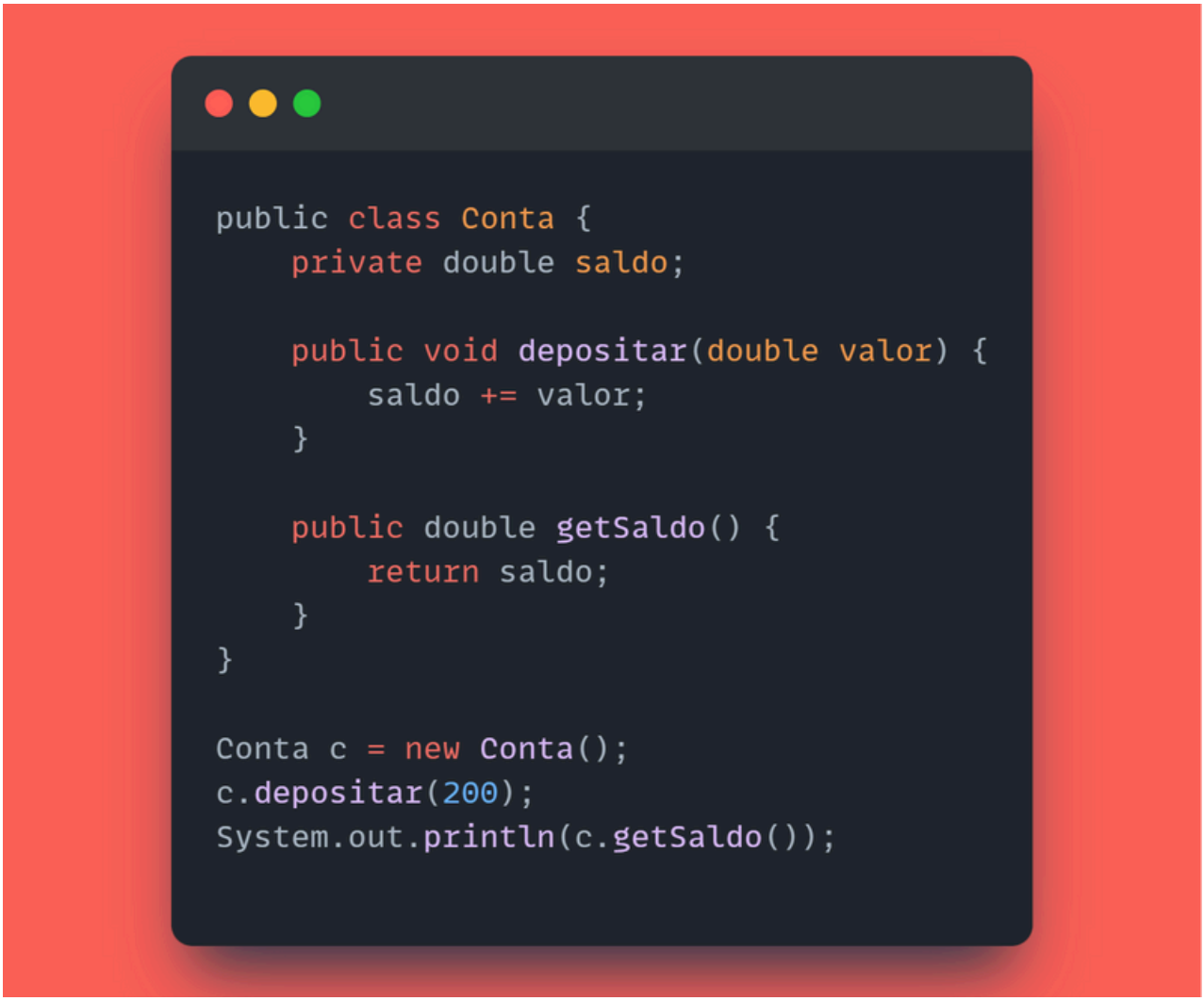


```java
public class Carro {
    String modelo;

    Carro(String modelo) {
        this.modelo = modelo;
    }

    void exibir() {
        System.out.println("Modelo: " + modelo);
    }
}

Carro c = new Carro("Civic");
c.exibir(); // Civic
```

Encapsulamento

Protegendo dados

Encapsular é proteger os dados internos do objeto. Em vez de acessar diretamente um atributo, você usa métodos públicos (getters e setters). Isso evita problemas e mantém o controle.



```java
public class Conta {
    private double saldo;

    public void depositar(double valor) {
        saldo += valor;
    }

    public double getSaldo() {
        return saldo;
    }
}

Conta c = new Conta();
c.depositar(200);
System.out.println(c.getSaldo());
```

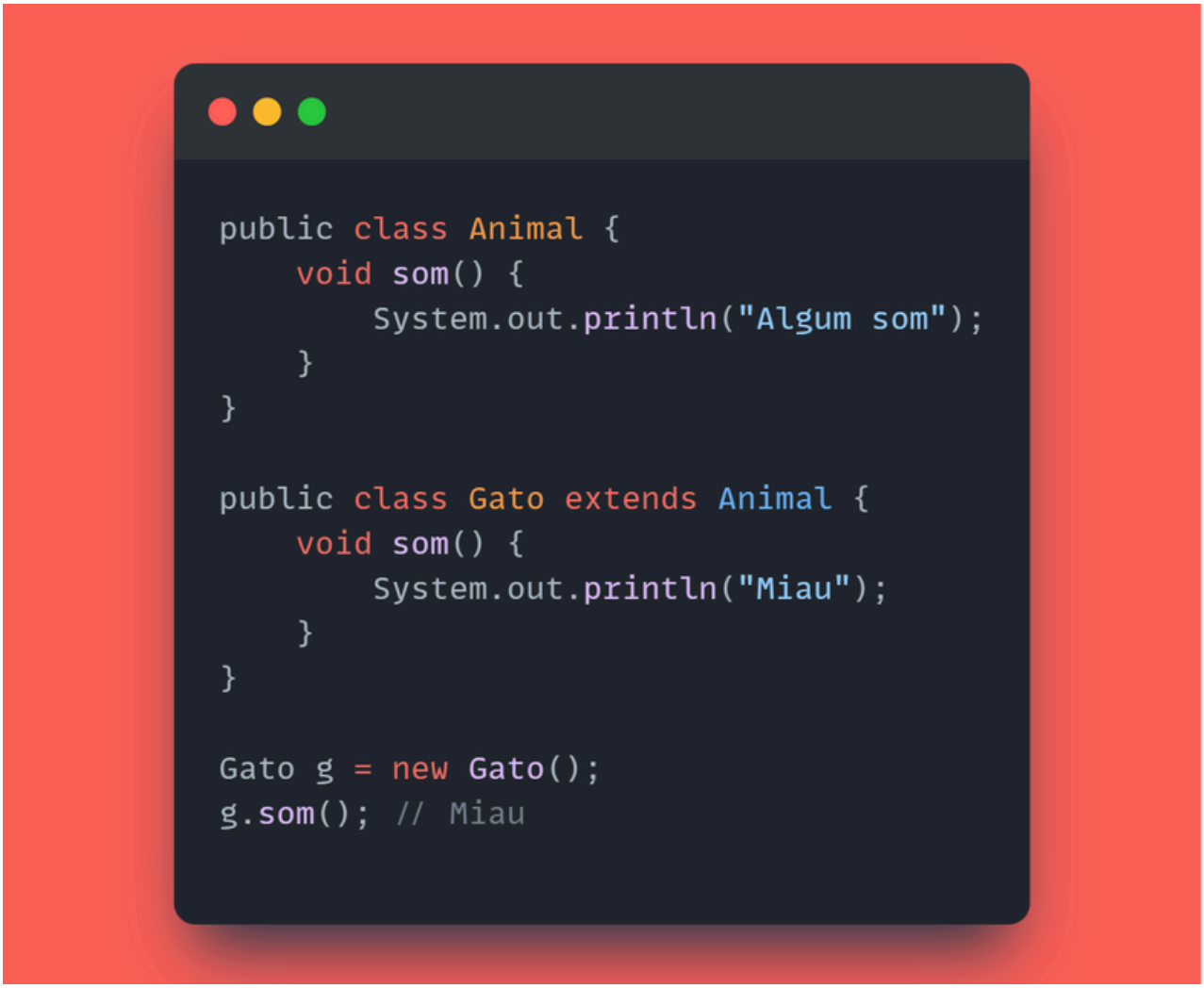
03

Herança, Polimorfismo e Reutilização de Código

Herança, Polimorfismo e Reutilização de Código

Herança: Reaproveitando comportamentos

Herança permite que uma classe "filha" aproveite atributos e métodos de uma classe "pai". É como se você criasse uma nova classe que já vem com funcionalidades prontas, podendo personalizá-la.

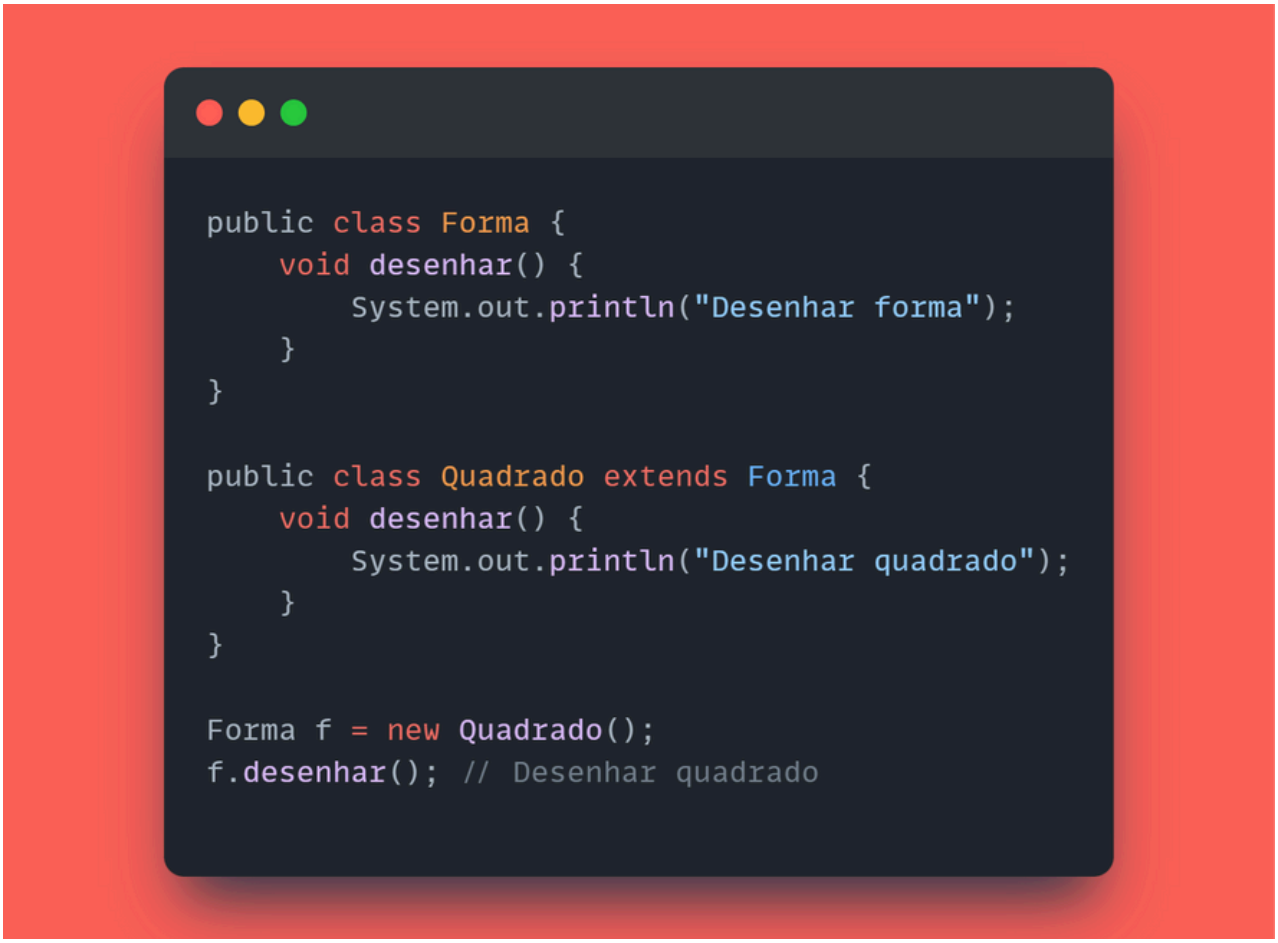


```
public class Animal {
    void som() {
        System.out.println("Algum som");
    }
}

public class Gato extends Animal {
    void som() {
        System.out.println("Miau");
    }
}

Gato g = new Gato();
g.som(); // Miau
```

Polimorfismo

Mesmo comando, ações diferentes

Com o polimorfismo, você pode usar o mesmo método em classes diferentes, mas cada uma com sua própria forma de executar. Isso torna o código mais flexível e reutilizável.



```
public class Forma {
    void desenhar() {
        System.out.println("Desenhar forma");
    }
}

public class Quadrado extends Forma {
    void desenhar() {
        System.out.println("Desenhar quadrado");
    }
}

Forma f = new Quadrado();
f.desenhar(); // Desenhar quadrado
```

Agradecimentos

OBRIGADO POR LER ATÉ AQUI

Este material foi criado com o apoio de Inteligência Artificial, mas toda a estruturação, adaptação, revisão e diagramação foram feitas por mim, com foco em tornar o conteúdo didático e acessível para iniciantes em Java.

O objetivo deste eBook é compartilhar conhecimento e praticar habilidades técnicas e criativas dentro da programação, comunicação e organização de ideias.

https://github.com/Dev-LucasJ