# JAVA BOTS

## MOLDANDO O FUTURO DA PROGRAMAÇÃO COM JAVA


ÍRIS FERREIRA

# CONSTRUINDO COM OBJETIVOS

## Entendendo a Orientação a Objetos

No vasto universo da programação, Java brilha como uma estrela guia. Sua versatilidade e robustez o tornam uma escolha popular entre desenvolvedores de todo o mundo. Neste ebook, vamos explorar algumas das principais características e suas aplicações, ajudando você a dominar o poder do


# ORIENTAÇÃO A OBJETOS

## CONSTRUINDO COM OBJETIVOS: ENTENDENDO A ORIENTAÇÃO A OBJETOS

# CONSTRUINDO COM OBJETIVOS: ENTENDENDO A ORIENTAÇÃO A OBJETOS

Java é uma linguagem de programação orientada a objetos, o que significa que ela segue os princípios da programação orientada a objetos (POO). Na POO, os programas são organizados em torno de objetos, que representam entidades do mundo real com características (atributos) e comportamentos (métodos). Os objetos são instâncias de classes, que são como moldes ou blueprints para criar objetos. Uma classe define o comportamento e as propriedades que seus objetos terão.
Em Java, tudo é considerado um objeto, exceto os tipos primitivos de dados, como inteiros e booleanos Vejamos um exemplo simples de uma classe `carro ´:

JavaBots - Íris Ferreira

```java
public class Carro {
    private String marca;
    private String modelo;

    public Carro(String marca, String modelo) {
        this.marca = marca;
        this.modelo = modelo;
    }

    public void ligar() {
        System.out.println("O carro está ligado.");
    }
}
```

02

# PILARES

**OS 4 PILARES DA ORIENTAÇÃO A OBJETOS**

# OS 4 PILARES DA ORIENTAÇÃO A OBJETOS

A orientação a objetos em Java é caracterizada por quatro pilares principais:

1. **Abstração**: Abstração é o processo de identificar as características essenciais de um objeto do mundo real e representá-las em um modelo de classe. Em Java, uma classe define uma abstração de um objeto, especificando seus atributos e métodos, mas ocultando os detalhes de implementação.

2. **Encapsulamento:** Encapsulamento é o princípio de esconder os detalhes internos de um objeto e expor apenas as operações que podem ser realizadas sobre ele. Em Java, isso é alcançado usando modificadores de acesso (como public, private e protected) para controlar o acesso aos membros de uma classe.

3. **Herança:** Herança é a capacidade de uma classe herdar características (atributos e métodos) de outra classe. Em Java, isso é feito usando a palavra-chave extends. A classe que herda é chamada de subclasse, e a classe que é herdada é chamada de superclasse. A herança permite reutilizar o código e estabelecer relações entre classes.

4. **Polimorfismo:** Polimorfismo é a capacidade de objetos de classes diferentes responderem ao mesmo método de maneiras diferentes. Em Java, isso pode ser alcançado através do sobrescrito de métodos (override) e da sobrecarga de métodos (overload). O polimorfismo permite escrever código mais genérico e flexível.

Java é uma linguagem orientada a objetos, o que significa que tudo é tratado como um objeto. Vejamos um exemplo simples de uma classe **Carro**:


# PORTABILIDADE

**NAVEGANDO POR PLATAFORMAS: JAVA E SUA PORTABILIDADE:**

# NAVEGANDO POR PLATAFORMAS: JAVA E SUA PORTABILIDADE

Uma das características marcantes do Java é sua portabilidade, graças à JVM (Java Virtual Machine). Um programa Java pode ser executado em qualquer dispositivo que tenha uma JVM instalada. Por exemplo, este programa "Hello World" pode ser executado em qualquer sistema operacional:

JavaBots - Íris Ferreira

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

04

# SEGURANÇA

## PROTEGENDO SEU CÓDIGO: JAVA E SUA ABORDAGEM SEGURA

# PROTEGENDO SEU CÓDIGO: JAVA E SUA ABORDAGEM SEGURA

Java é conhecido por sua abordagem robusta à segurança. Por exemplo, o uso do modificador `private´ em Java garante que certos dados só possam ser acessados pela própria classe. Veja como isso funciona:

JavaBots - Íris Ferreira

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

    public ContaBancaria(double saldoInicial)
        this.saldo = saldoInicial;
    }

    public double getSaldo() {
        return saldo;
    }
}
```

05

# MULTITHREADING

## EXECUTANDO EM PARALELO: JAVA E SUA CAPACIDADE MULTITHREADING

# EXECUTANDO EM PARALELO: JAVA E SUA CAPACIDADE MULTITHREADING

Java suporta a execução de múltiplas threads simultaneamente, permitindo a criação de programas que podem realizar várias tarefas ao mesmo tempo. Por exemplo, podemos criar uma thread simples que imprime números de 1 a 5:

JavaBots - Íris Ferreira

```java
public class Main {
    public static void main(String[] args) {
        Thread thread = new Thread(() -> {
            for (int i = 1; i <= 5; i++) {
                System.out.println(i);
                try {
                    Thread.sleep(1000); // Aguarda 1 segundo entre cada número
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
        thread.start();
    }
}
```

06

# SIMPLICIDADE E CLAREZA

**CODIFICANDO COM ELEGÂNCIA: JAVA E SUA SINTAXE SIMPLES**

# CODIFICANDO COM ELEGÂNCIA: JAVA E SUA SINTAXE SIMPLES

Java é projetado para ser uma linguagem simples e fácil de entender. Por exemplo, podemos criar um método simples para somar dois números:

JavaBots - Íris Ferreira

```java
public class Calculadora {
    public static int somar(int a, int b) {
        return a + b;
    }
}
```

# OBRIGADO