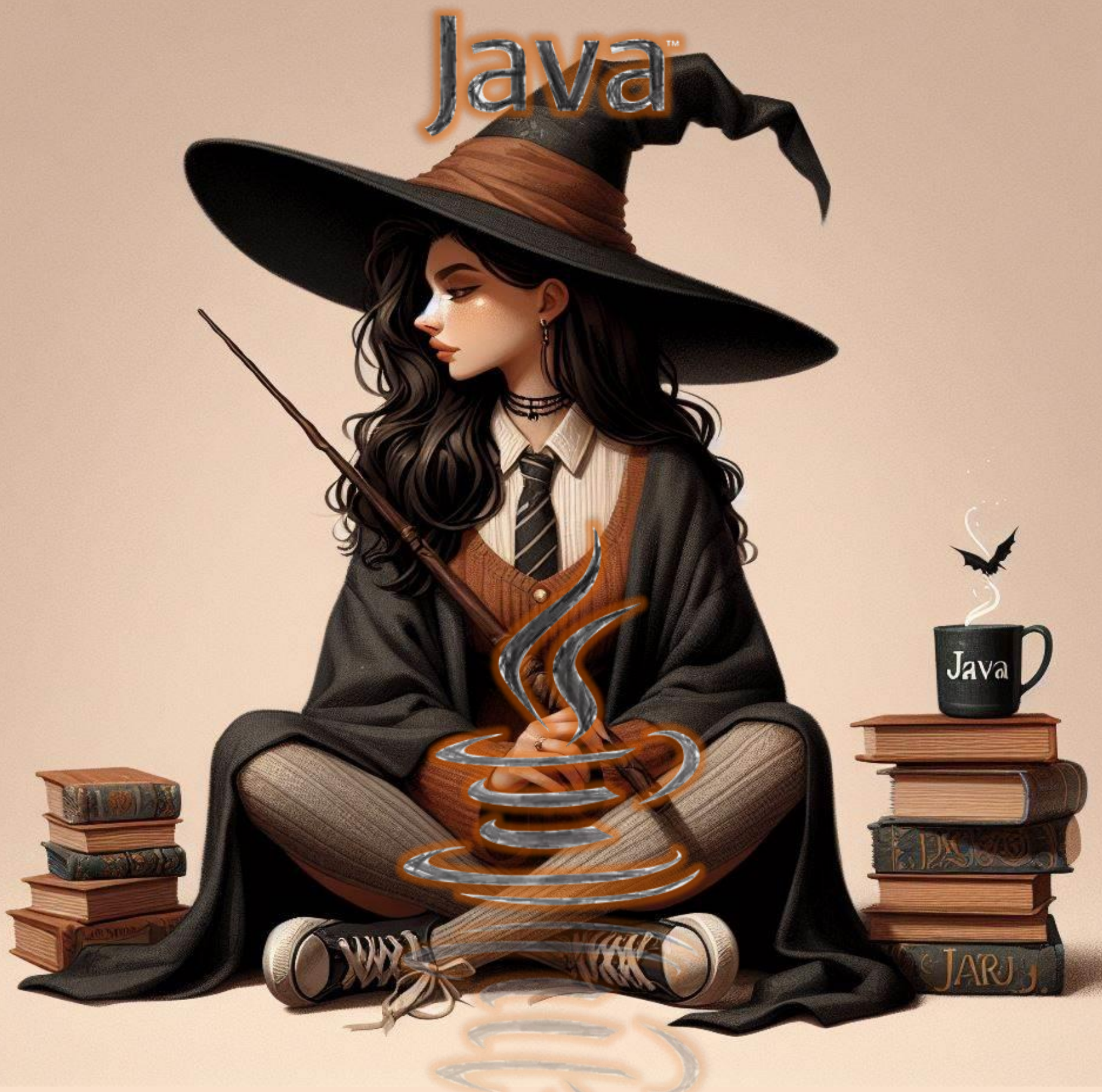
Java e a Ordem dos Desenvolvedores



Flavia Ribeiro

Índice

01

A Pedra Filosofal da Programação

Introdução ao Java

Java é uma linguagem de programação robusta e de propósito geral, originalmente desenvolvida pela Sun Microsystems e atualmente mantida pela Oracle. Com sua capacidade de "escrever uma vez, rodar em qualquer lugar" (WORA), Java é amplamente utilizada em aplicações corporativas, desenvolvimento web, dispositivos móveis e muito mais.

Instalação e Configuração do Ambiente

Antes de começar a programar em Java, precisamos configurar nosso ambiente de desenvolvimento:

Instalar o JDK (Java Development Kit)

Baixe o JDK mais recente do site oficial da Oracle.
Siga as instruções de instalação específicas para o seu sistema operacional.
Configurar Variáveis de Ambiente

Adicione o caminho do JDK à variável de ambiente PATH.
Verifique a instalação abrindo o terminal e digitando java -version.

Primeiro Programa em Java: "Olá, Mundo!“

Vamos escrever nosso primeiro programa em Java para nos familiarizarmos com a sintaxe básica.

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

public class OlaMundo: Declara uma classe pública chamada OlaMundo.

public static void main(String[] args): O ponto de entrada do programa, onde a execução começa.

System.out.println("Olá, Mundo!"): Imprime "Olá, Mundo!" no console.

02

O Segredo da Herança

Estrutura Básica de um Programa Java

Um programa Java é composto por classes e métodos. Cada aplicação Java começa pela classe que contém o método main.

Tipos de Dados e Variáveis

Java possui vários tipos de dados primitivos como int, double, char, boolean, entre outros. Variáveis são usadas para armazenar dados, e cada variável deve ser declarada com um tipo específico.

```java
int numero = 10;
double preco = 29.99;
char letra = 'A';
boolean ativo = true;
```

Operadores e Expressões

Java suporta operadores aritméticos (+, -, *, /), relacionais (==, !=, <, >) e lógicos (&&, ||, !).

```java
int a = 5;
int b = 10;
int soma = a + b;
boolean resultado = (a < b) && (b > 5);
```

Estruturas de Controle de Fluxo

Estruturas de controle como if, else, switch, for, while e do-while controlam o fluxo de execução.

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

03

O Cálice da Coleção

Arrays e Coleções

Arrays são estruturas de dados que armazenam elementos do mesmo tipo. Coleções são mais flexíveis e podem armazenar diferentes tipos de dados.

```java
int[] numeros = {1, 2, 3, 4, 5};
ArrayList<String> nomes = new ArrayList<>();
nomes.add("Harry");
nomes.add("Hermione");
```

Framework de Coleções do Java

O Java Collections Framework inclui classes como ArrayList, HashSet e HashMap para gerenciar coleções de objetos.

```java
HashMap<String, Integer> mapa = new HashMap<>();
mapa.put("Harry", 20);
mapa.put("Hermione", 19);
```

Manipulação de Arrays e Coleções

Arrays são estruturas de dados que armazenam elementos do mesmo tipo. Coleções são mais flexíveis e podem armazenar diferentes tipos de dados.

```java
for (int numero : numeros) {
    System.out.println(numero);
}
```

04

A Ordem da Orientação a Objetos

Conceitos de Orientação a Objetos

Orientação a Objetos é um paradigma de programação baseado em objetos que possuem atributos e comportamentos.

Classes e Objetos

Uma classe é um modelo para objetos. Um objeto é uma instância de uma classe.

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

    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

    public void mostrarInfo() {
        System.out.println(nome + " tem " + idade + "
anos.");
    }
}

public class Main {
    public static void main(String[] args) {
        Pessoa harry = new Pessoa("Harry", 20);
        harry.mostrarInfo();
    }
}
```

Herança, Polimorfismo e Encapsulamento

Herança permite que classes derivadas herdem atributos e métodos. Polimorfismo permite que métodos se comportem de maneira diferente dependendo do objeto. Encapsulamento esconde detalhes internos da classe.

```java
public class Estudante extends Pessoa {
    int ano;

    public Estudante(String nome, int idade, int ano) {
        super(nome, idade);
        this.ano = ano;
    }

    @Override
    public void mostrarInfo() {
        super.mostrarInfo();
        System.out.println("Ano: " + ano);
    }
}
```

Interfaces e Classes Abstratas

Interfaces definem métodos que devem ser implementados por classes. Classes abstratas podem ter métodos abstratos e concretos.

```java
public interface Animal {
    void emitirSom();
}

public class Cachorro implements Animal {
    public void emitirSom() {
        System.out.println("Au Au");
    }
}
```

05

As Relíquias do Desenvolvimento e Persistência

Tratamento de Exceções

Exceções são eventos que ocorrem durante a execução e interrompem o fluxo normal. O tratamento de exceções é feito usando try, catch e finally.

```java
try {
    int resultado = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Erro: Divisão por zero.");
} finally {
    System.out.println("Bloco finally executado.");
}
```

Programação Multithreaded

Exceções são eventos que ocorrem durante a execução e interrompem o fluxo normal. O tratamento de exceções é feito usando try, catch e finally.

```java
public class MinhaThread extends Thread {
    public void run() {
        System.out.println("Thread em execução.");
    }
}

public class TesteThread {
    public static void main(String[] args) {
        MinhaThread thread = new MinhaThread();
        thread.start();
    }
}
```

JDBC e Conexão com Banco de Dados

Exceções são eventos que ocorrem durante a execução e interrompem o fluxo normal. O tratamento de exceções é feito usando try, catch e finally.

```java
import java.sql.*;

public class ConexaoBD {
    public static void main(String[] args) {
        try {
            Connection conexao =
DriverManager.getConnection("jdbc:mysql://localhost:3306/
            meubanco", "usuario", "senha");
            Statement stmt = conexao.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM
minha_tabela");

            while (rs.next()) {
                System.out.println(rs.getString("coluna"));

            }

            conexao.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

06

Os Fundamentos da Web com Servlets e JSP

Arquitetura de Aplicações Web

Java é frequentemente utilizado para criar aplicações web robustas e escaláveis, utilizando o padrão MVC (Model-View-Controller).

Servlets e JSP

Servlets processam requisições e respostas HTTP:

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class OlaServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1>Olá, Mundo Servlet!</h1>");
    }
}
```

JavaServer Pages (JSP) permitem a criação de conteúdo HTML dinâmico, embutindo código Java diretamente nas páginas HTML.

07

Desenvolvendo Aplicações Web com Spring MVC

Introdução ao Spring MVC

Spring MVC é um módulo do Spring Framework que facilita a construção de aplicações web seguindo o padrão MVC.

Configuração do Projeto Spring Boot

Criar um projeto Spring Boot usando Spring Initializr.
Adicionar dependências no pom.xml:

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Controladores em Spring MVC

Os controladores gerenciam as requisições HTTP e mapeiam-nas para métodos específicos.

```java
import org.springframework.web.bind.annotation.*;

@RestController
public class OlaController {
    @RequestMapping("/ola")
    public String ola() {
        return "Olá, Spring MVC!";
    }
}
```

Thymeleaf como Motor de Template

Adicionar dependência do Thymeleaf no pom.xml:

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

Criar uma página Thymeleaf (ola.html):

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Olá</title>
</head>
<body>
    <h1 th:text="${mensagem}">Mensagem aqui</h1>
</body>
</html>
```

08

Persistência e Web Services RESTful com Spring Boot

Integração com Banco de Dados usando Spring Data JPA

Adicionar dependência do Spring Data JPA:

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Configurar a conexão com o banco de dados no application.properties:

```properties
spring.datasource.url=jdbc:mysql://localhost:3306/meubanco
spring.datasource.username=usuario
spring.datasource.password=senha
spring.jpa.hibernate.ddl-auto=update
```

Criar uma entidade e um repositório:

```java
import javax.persistence.*;

@Entity
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nome;
    private String email;
    // Getters e Setters
}
```

```java
import
org.springframework.data.jpa.repository.JpaRepository;

public interface UsuarioRepository extends
JpaRepository<Usuario, Long> {
}
```

RESTful Web Services com Spring Boot

Criar um controlador REST:

```java
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api")
public class UsuarioController {
    private final UsuarioRepository usuarioRepository;

    public UsuarioController(UsuarioRepository
usuarioRepository) {
        this.usuarioRepository = usuarioRepository;
    }

    @GetMapping("/usuarios")
    public List<Usuario> listarUsuarios() {
        return usuarioRepository.findAll();
    }

    @PostMapping("/usuarios")
    public Usuario criarUsuario(@RequestBody Usuario
usuario) {
        return usuarioRepository.save(usuario);
    }
}
```

Agradecimentos

OBRIGADA POR LER ATÉ AQUI