Estruturas & Algoritmos:
A Odisseia Java

Raniery M. Goulart

Introdução

Introdução

Você já se perguntou como jogos armazenam inventários, como redes sociais recomendam amigos ou como apps organizam seus dados em tempo real? Tudo isso acontece graças a estruturas de dados — os verdadeiros heróis invisíveis por trás da eficiência dos sistemas que usamos todos os dias.

Neste eBook, vamos explorar o universo das estruturas de dados com Java de forma prática, enxuta e acessível. Nada de explicações complicadas ou fórmulas abstratas: aqui você vai ver exemplos reais, entender quando usar cada estrutura e como aplicá-las no seu código como um verdadeiro mestre Jedi da programação.

Se você é estudante, desenvolvedor iniciante ou até mesmo um dev experiente querendo revisar fundamentos, este material é para você. A cada capítulo, vamos destrinchar conceitos como listas, mapas, filas, pilhas e muito mais — tudo com código comentado, cenários do mundo real e foco em clareza.

01

Capítulo 1 – Arrays: A Base de Tudo

Arrays: A Base de Tudo

O que é um Array?

Um *array* é uma estrutura de dados que armazena uma coleção de elementos do mesmo tipo. Ele é ideal para armazenar valores fixos, como dias da semana, notas ou IDs.

Exemplo prático: médias de uma turma

A Odisseia Java

```java
public class MediaTurma {
    public static void main(String[] args) {
        double[] notas = {7.5, 8.0, 6.0, 9.2};
        double soma = 0;

        for (double nota : notas) {
            soma += nota;
        }

        double media = soma / notas.length;
        System.out.println("Média da turma: " + media);
    }
}
```

Por que usar?

- Acesso rápido por índice (*O(1)*)
- Estrutura leve e eficiente para leitura sequencial

02

Capítulo 2 – Listas: Flexibilidade em Movimento

Listas: Flexibilidade em Movimento

Por que usar ArrayList ao invés de arrays?

Quando você precisa de uma coleção que cresce dinamicamente, o *ArrayList* é uma ótima escolha.Exemplo prático: carrinho de compras

A Odisseia Java

```java
import java.util.ArrayList;

public class Carrinho {
    public static void main(String[] args) {
        ArrayList<String> carrinho = new ArrayList<>();

        carrinho.add("Notebook");
        carrinho.add("Mouse");
        carrinho.add("Teclado");

        carrinho.remove("Mouse");

        for (String item : carrinho) {
            System.out.println("Item: " + item);
        }
    }
}

```

Vantagens:

- Crescimento automático
- Métodos prontos como *add()*, *remove()*, *contains()*

03

Capítulo 3 – Mapas: Dados com Chave e Valor

Mapas: Dados com Chave e Valor

Quando usar *HashMap*?

Quando você precisa acessar rapidamente dados associados a uma chave, como perfis de usuários ou configurações.

Exemplo prático: cadastro de usuários

A Odisseia Java

```java
import java.util.HashMap;

public class CadastroUsuarios {
    public static void main(String[] args) {
        HashMap<String, String> usuarios = new HashMap<>();

        usuarios.put("joao123", "João da Silva");
        usuarios.put("ana456", "Ana Pereira");

        String nome = usuarios.get("ana456");
        System.out.println("Usuário encontrado: " + nome);
    }
}

```

Vantagens:

- Acesso rápido por chave (*O(1)* na média)
- Ideal para configurações, caches, e autenticação

04

Capítulo 4 – Pilhas: A Última que Entra é a Primeira a Sair

Pilhas: A Última que Entra é a Primeira a Sair

Entendendo *Stack*

Uma pilha segue o princípio LIFO (Last-In, First-Out). Perfeita para cenários como navegação de páginas, recuperação de estados ou desfazer ações.

Exemplo prático: voltar páginas

A Odisseia Java

```java
import java.util.Stack;

public class Navegador {
    public static void main(String[] args) {
        Stack<String> historico = new Stack<>();

        historico.push("Página 1");
        historico.push("Página 2");
        historico.push("Página 3");

        System.out.println("Voltando para: " + historico.pop());
    }
}

```

05

Capítulo 5 – Filas: Ordem de Chegada Importa

Filas: Ordem de Chegada Importa

***Queue:* Primeiro que entra, primeiro que sai (FIFO)**

Usada em filas de impressão, atendimento ou execução de tarefas.

Exemplo prático: fila de atendimento

A Odisseia Java

```java
import java.util.LinkedList;
import java.util.Queue;

public class FilaAtendimento {
    public static void main(String[] args) {
        Queue<String> fila = new LinkedList<>();

        fila.add("Cliente 1");
        fila.add("Cliente 2");

        System.out.println("Atendendo: " + fila.poll());
        System.out.println("Próximo: " + fila.peek());
    }
}

```

Agradecimentos

Obrigado por ler até aqui

Este ebook foi gerado por IA e diagramado por um humano.

Este conteúdo foi gerado com fins didáticos de construção, não foi realizado uma validação cuidadosa humana do conteúdo e pode conter erros gerados por uma IA.

https://github.com/Ranieeery