

Estrutura de Dados



Java

Nicole Ellen M. Silvestre

Estrutura de Dados

aprendendo com Java

Sumário

Capítulo 1: Arrays e Coleções

- Arrays unidimensionais e multidimensionais
- Introdução às coleções em Java: ArrayList, LinkedList, HashSet, HashMap
- Operações básicas em arrays e coleções (inserção, remoção, busca)
- Comparação entre arrays e coleções

Capítulo 2: Pilhas e Filas

- Arrays unidimensionais e multidimensionais
- Introdução às coleções em Java: ArrayList, LinkedList, HashSet, HashMap
- Operações básicas em arrays e coleções (inserção, remoção, busca)
- Comparação entre arrays e coleções

Capítulo 3: Listas Encadeadas

- Introdução às listas encadeadas
- Implementação de listas encadeadas simples e duplamente encadeadas
- Operações básicas em listas encadeadas (inserção, remoção, busca)
- Comparação entre listas encadeadas e arrays

Capítulo 4: Árvores e Árvores Binárias

- Conceito de árvores e árvores binárias
- Tipos de Árvores
- Implementação de árvores binárias de busca (BST) em Java
- Operações básicas em árvores binárias (inserção, remoção, busca)
- Traversals em árvores binárias (pré-ordem, em ordem, pós-ordem)
- Exercícios práticos de implementação e manipulação de árvores binárias

Capítulo 5: Grafos

- Introdução aos grafos e representações em Java
- Algoritmos de busca em grafos: BFS (Busca em Largura) e DFS (Busca em Profundidade)
- Aplicações práticas de grafos em algoritmos e problemas do mundo real
- Exercícios práticos de implementação e manipulação de grafos

01

Arrays e Colecções

Start

Neste capítulo, vamos mergulhar nos conceitos de arrays e coleções em Java. Essas estruturas são essenciais para organizar e manipular conjuntos de dados em programas Java. O que são Arrays e Coleções?

- **Arrays:** Imagine que você tem uma caixa de sapatos onde você pode colocar vários itens de uma vez e cada item tem uma etiqueta com um número. Um array em Java é como essa caixa de sapatos - é uma estrutura de dados que armazena uma coleção ordenada de itens do mesmo tipo. Cada item no array é acessado por meio de um índice numérico.
- **Coleções:** Agora, pense em uma prateleira na sua cozinha onde você pode colocar uma variedade de potes, cada um contendo um ingrediente diferente. As coleções em Java são semelhantes a essa prateleira - são estruturas de dados que podem armazenar um número variável de elementos, cada um associado a uma chave ou índice único.

Para que são usados?

- **Arrays:** Os arrays são úteis quando você precisa armazenar uma coleção fixa de elementos do mesmo tipo, como notas de um aluno, dias da semana ou coordenadas em um plano cartesiano.
- **Coleções:** As coleções são especialmente úteis quando você precisa manipular conjuntos de dados dinâmicos, onde o número de elementos pode mudar ao longo do tempo. Elas são utilizadas em uma variedade de situações, como gerenciamento de listas de contatos, registros de produtos em um inventário ou resultados de uma pesquisa.

Como funcionam?

- **Arrays:** Em um array, cada elemento é acessado por meio de um índice numérico. O primeiro elemento está no índice 0, o segundo no índice 1 e assim por diante. Os arrays têm um tamanho fixo, o que significa que você precisa especificar o número de elementos que ele pode conter no momento da criação.
- **Coleções:** As coleções em Java são implementadas através de classes como ArrayList, LinkedList, HashSet, entre outras. Elas oferecem métodos para adicionar, remover e acessar elementos, e podem crescer ou diminuir dinamicamente conforme necessário.

Exemplos Práticos:

- **Arrays:** Os arrays são úteis quando você precisa armazenar uma coleção fixa de elementos do mesmo tipo, como notas de um aluno, dias da semana ou coordenadas em um plano cartesiano.
- **Coleções:** As coleções são especialmente úteis quando você precisa manipular conjuntos de dados dinâmicos, onde o número de elementos pode mudar ao longo do tempo. Elas são utilizadas em uma variedade de situações, como gerenciamento de listas de contatos, registros de produtos em um inventário ou resultados de uma pesquisa.

```
// Declarando e inicializando um array de notas  
double[] notas = {7.5, 8.0, 9.2, 6.8, 7.0};
```

Ex: array

```
// Importando a classe ArrayList  
import java.util.ArrayList;  
  
// Criando uma lista de produtos  
ArrayList<String> produtos = new ArrayList<String>();  
  
// Adicionando produtos à lista  
produtos.add("Camiseta");  
produtos.add("Calça");  
produtos.add("Tênis");
```

Ex: coleção

Entender como usar arrays e coleções é fundamental para o desenvolvimento eficaz de programas Java. Pratique e experimente diferentes cenários para consolidar seu conhecimento.

02

Pilhas
e
Filas

Start

O que são Pilhas e Filas?

Pilhas: Imagine uma pilha de pratos na sua cozinha. Você só pode adicionar ou remover pratos no topo da pilha. Pilhas em Java funcionam da mesma maneira - você só pode adicionar ou remover elementos em uma extremidade da estrutura, chamada de topo.

Filas: Agora, pense em uma fila em um caixa de supermercado. As pessoas entram na fila na parte de trás e saem na parte da frente. Filas em Java funcionam de maneira semelhante - você pode adicionar elementos no final da fila e remover elementos do início.

Exemplos Práticos:

Pilhas: Um exemplo comum de pilha é o histórico de navegação em um navegador da web. Você pode voltar para páginas visitadas anteriormente, uma por vez, como se estivesse removendo os pratos da pilha na ordem em que foram adicionados.



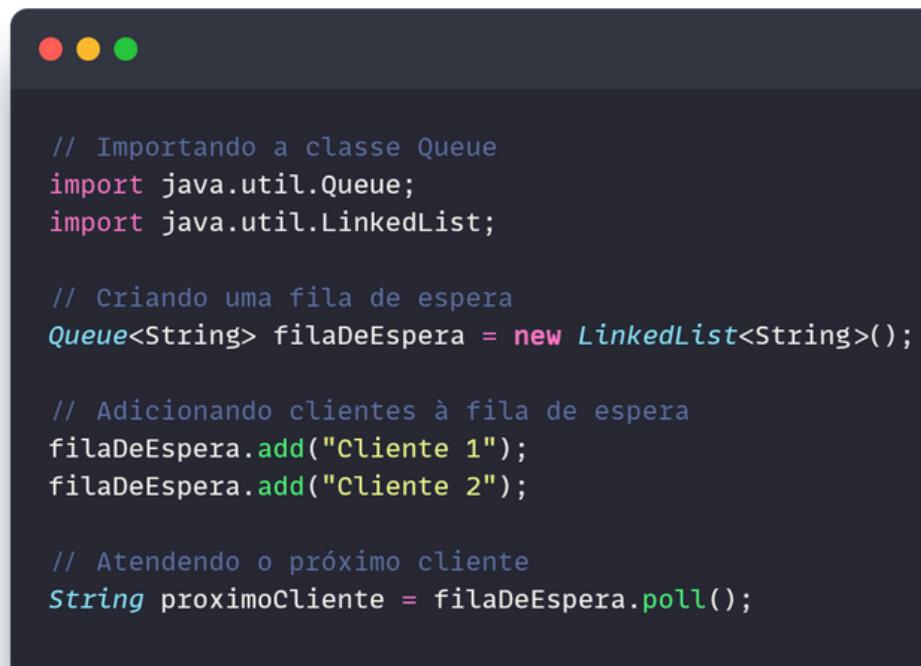
```
// Importando a classe Stack
import java.util.Stack;

// Criando uma pilha
Stack<String> historico = new Stack<String>();

// Adicionando páginas visitadas à pilha
historico.push("Página Inicial");
historico.push("Página de Notícias");

// Voltando para a página anterior
String paginaAnterior = historico.pop();
```

Filas: Um exemplo prático de fila é uma fila de espera em um restaurante. As pessoas entram na fila na ordem em que chegaram e são atendidas uma por vez, começando pela pessoa que está na frente da fila.



```
// Importando a classe Queue
import java.util.Queue;
import java.util.LinkedList;

// Criando uma fila de espera
Queue<String> filaDeEspera = new LinkedList<String>();

// Adicionando clientes à fila de espera
filaDeEspera.add("Cliente 1");
filaDeEspera.add("Cliente 2");

// Atendendo o próximo cliente
String proximoCliente = filaDeEspera.poll();
```

03

listas Encadeadas

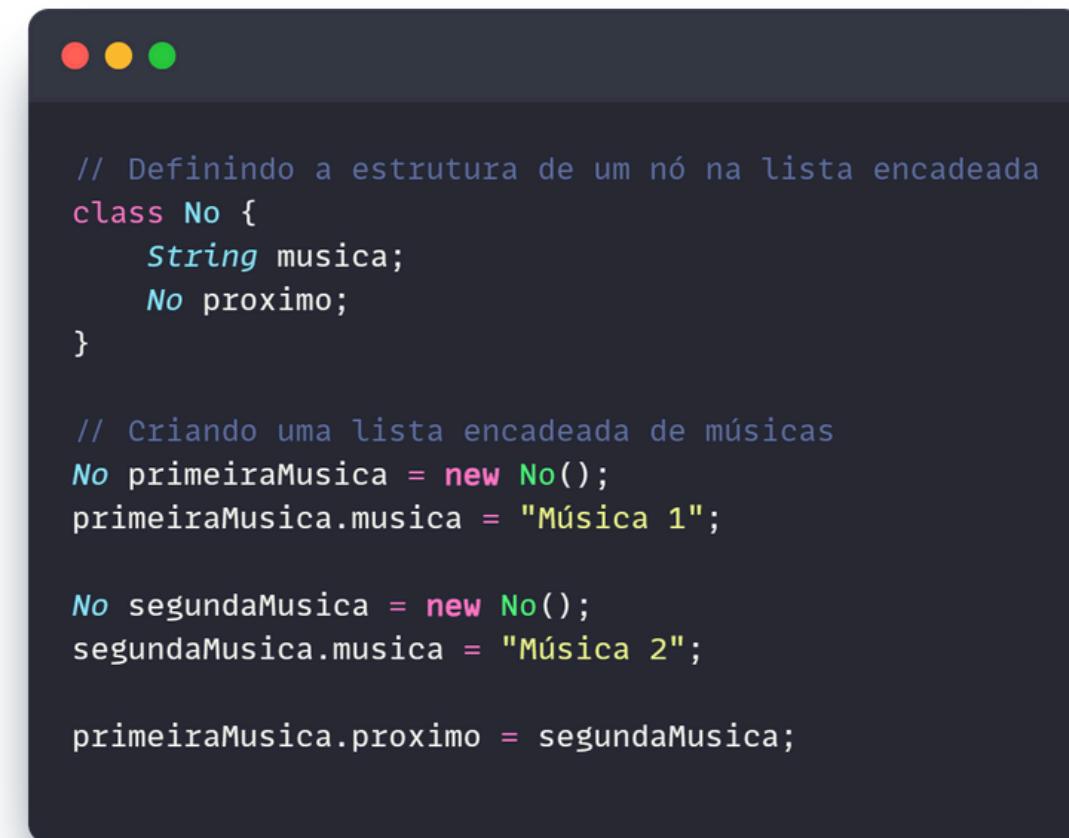
Start

O que são Listas Encadeadas?

- Listas Encadeadas: Imagine uma corrente de elos onde cada elo tem uma conexão com o próximo. Listas encadeadas em Java funcionam de maneira semelhante - cada elemento na lista tem uma referência para o próximo elemento na sequência.

Exemplos Práticos:

- Um exemplo prático de lista encadeada é uma lista de reprodução de músicas. Cada música na lista está conectada à próxima, permitindo que você reproduza as músicas em ordem.



```
// Definindo a estrutura de um nó na lista encadeada
class No {
    String musica;
    No proximo;
}

// criando uma lista encadeada de músicas
No primeiraMusica = new No();
primeiraMusica.musica = "Música 1";

No segundaMusica = new No();
segundaMusica.musica = "Música 2";

primeiraMusica.proximo = segundaMusica;
```

04

Árvores e Árvores Binárias

Start

O que são Árvores e Árvores Binárias?

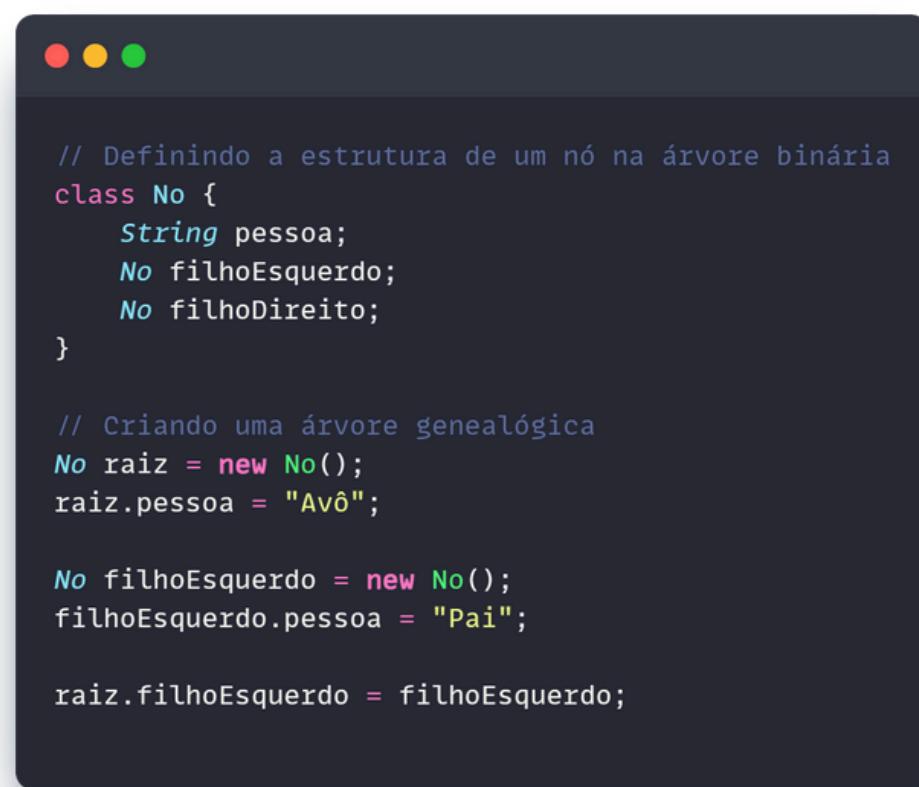
- Árvores: Imagine uma estrutura hierárquica de informações onde cada elemento tem conexões com outros elementos abaixo dele. Árvores em Java funcionam de maneira semelhante - cada nó na árvore tem zero ou mais nós filhos. As árvores são usadas para representar relacionamentos hierárquicos entre dados, como a estrutura de arquivos em um sistema operacional ou a organização de uma empresa.
- Árvores Binárias: Árvores binárias são um tipo especial de árvore onde cada nó tem no máximo dois filhos: um filho esquerdo e um filho direito. Essa estrutura de árvore é amplamente utilizada em algoritmos de busca e ordenação. Existem vários tipos de árvores, cada uma com suas características e aplicações específicas, incluindo árvores de busca binária, árvores AVL, árvores rubro-negras, entre outras.

Tipos de Árvores:

1. Árvores de Busca Binária (BST): Cada nó tem no máximo dois filhos, sendo que o filho esquerdo é menor que o nó pai e o filho direito é maior. São úteis para implementar algoritmos de busca eficientes, como a busca binária.
2. Árvores AVL: Uma árvore de busca binária balanceada, onde a diferença de altura entre as subárvores esquerda e direita de cada nó (fator de平衡amento) é mantida dentro de limites pré-definidos. Isso garante tempos de busca e inserção mais eficientes.
3. Árvores Rubro-Negras: Outro tipo de árvore de busca binária balanceada, onde cada nó tem uma cor (vermelho ou preto) atribuída a ele. Essas árvores têm regras especiais de平衡amento que garantem operações eficientes de inserção, exclusão e busca.

Exemplos Práticos:

- Um exemplo prático de árvore binária é uma árvore genealógica, onde cada pessoa na árvore tem zero, um ou dois pais (filhos esquerdo e direito).



```
// Definindo a estrutura de um nó na árvore binária
class No {
    String pessoa;
    No filhoEsquerdo;
    No filhoDireito;
}

// Criando uma árvore genealógica
No raiz = new No();
raiz.pessoa = "Avô";

No filhoEsquerdo = new No();
filhoEsquerdo.pessoa = "Pai";

raiz.filhoEsquerdo = filhoEsquerdo;
```

05

Grafos

Start

O que são Grafos?

- Grafos: Imagine uma rede complexa de conexões entre diferentes pontos. Grafos em Java representam essa estrutura de dados, onde os pontos são chamados de vértices e as conexões entre eles são chamadas de arestas.

Exemplos Práticos:

- Um exemplo prático de grafo é uma rede social, onde cada pessoa é um vértice e a conexão entre elas representa uma amizade ou relacionamento.



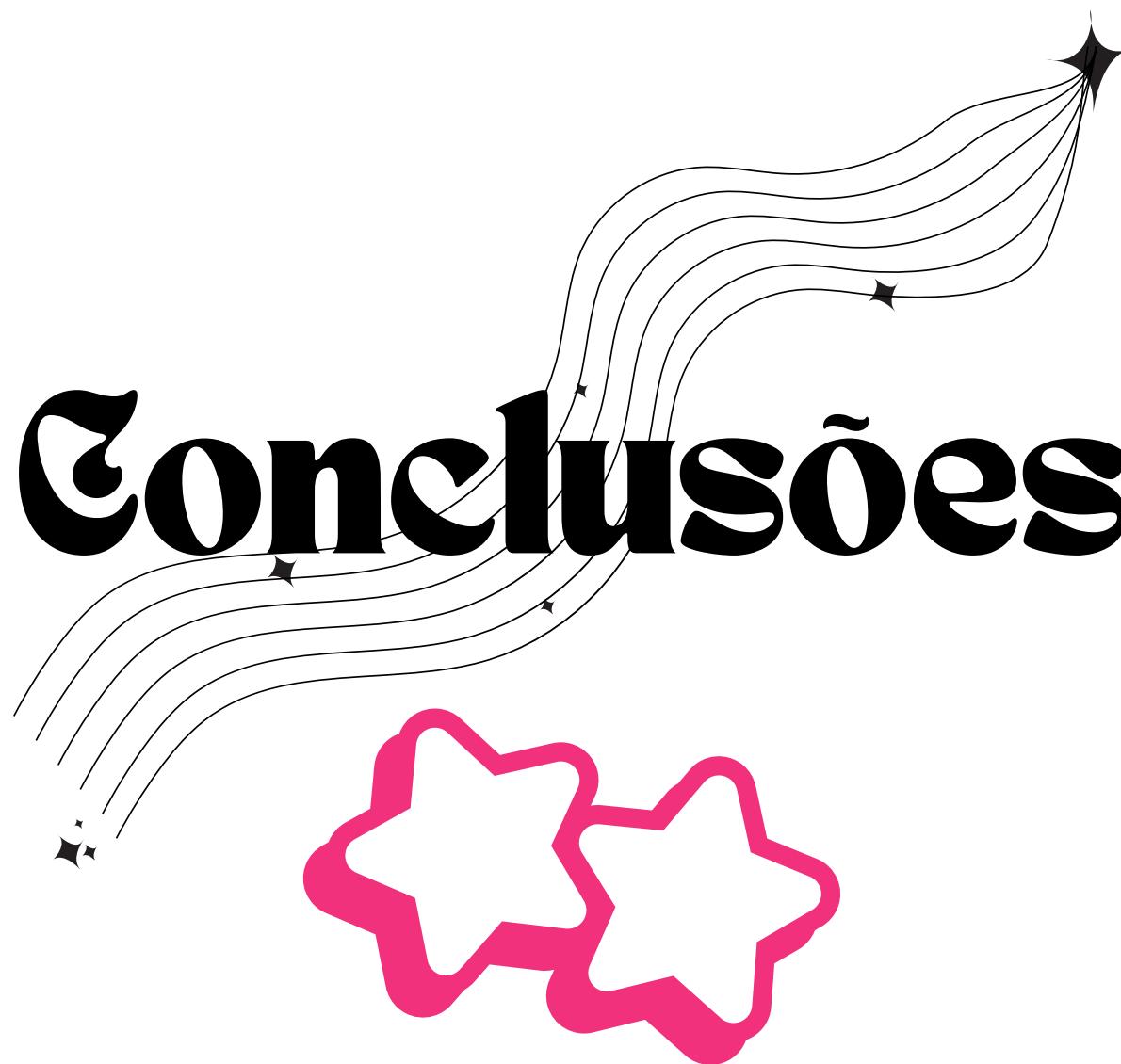
```
// Importando a classe Graph
import java.util.HashMap;

// Criando um grafo representando uma rede social
HashMap<String, ArrayList<String>> redeSocial = new HashMap<String, ArrayList<String>>();

// Adicionando amigos de cada pessoa à rede social
ArrayList<String> amigosJoao = new ArrayList<String>();
amigosJoao.add("Maria");
amigosJoao.add("Pedro");

redeSocial.put("João", amigosJoao);
```

Conclusões



Parabéns por chegar ao final deste guia prático sobre estrutura de dados em Java! Espero que você tenha adquirido uma compreensão sólida dos conceitos fundamentais e esteja pronto para aplicar esse conhecimento em seus projetos de programação.

Ao longo deste guia, exploramos várias estruturas de dados, desde arrays e coleções até árvores e grafos. Cada uma dessas estruturas desempenha um papel importante no desenvolvimento de algoritmos eficientes e na resolução de uma ampla gama de problemas de programação.

Lembre-se de que a jornada de aprendizado em programação é contínua e sempre há mais para descobrir e explorar. Continue praticando, participando de projetos e procurando novas oportunidades de aprendizado para aprimorar suas habilidades e expandir seus horizontes.

Agradeço por dedicar seu tempo a este guia e espero que ele tenha sido útil e inspirador em sua jornada de aprendizado em programação. Se você tiver alguma dúvida ou comentário, não hesite em entrar em contato.

Agradecimentos

Gostaria de expressar meus sinceros agradecimentos a todas as pessoas que contribuíram para a criação deste guia:

- Aos desenvolvedores e educadores que compartilharam seu conhecimento e experiência para enriquecer este guia.
- Aos meus colegas e amigos que forneceram feedback valioso durante o processo de criação.
- Aos leitores como você, cujo interesse e dedicação tornam possível a criação de recursos educacionais como este.

Se você gostou deste guia, considere compartilhá-lo com outras pessoas interessadas em aprender sobre estrutura de dados em Java. Seu apoio é fundamental para ajudar a comunidade de programadores a crescer e prosperar.

Obrigado novamente e continue programando com paixão!

Atenciosamente, Nicole <3

Conecte-se comigo ;)



<https://github.com/Nicoleel336>



<https://www.linkedin.com/in/nicole-ellen-magalh%C3%A3es-silvestre-7b32712bb/>