Estrutura de Dados I
Pilhas

Matheus Gabriel

Agosto 2024

1 Pilhas

1.1 Introdução

A pilha **(stack)** é uma estrutura de dados que define o acesso de dados usando o principio LIFO:

Definição 1.1: LIFO

LIFO (Último a Entrar, Primeiro a Sair): Um método de gerenciamento de dados onde o último elemento adicionado é o primeiro a ser removido. Comumente utilizado em estruturas de dados do tipo pilha, onde as operações são realizadas na ordem último a entrar, primeiro a sair.

- **Operação Push:** Adiciona o elemento x ao topo da pilha.
- **Operação Pop:** Remove o elemento do topo da pilha, que é x.

1.2 Métodos comuns para pilhas

Método	Uso	Exemplo
`create()`	Cria uma nova pilha	`Stack<Integer> stack = new Stack<>();`
`push(E e)`	Adiciona elemento ao topo	`stack.push(10);`
`pop()`	Remove e retorna o topo	`int item = stack.pop();`
`peek()`	Retorna o topo sem remover	`int top = stack.peek();`
`isEmpty()`	Verifica se está vazio	`boolean empty = stack.isEmpty();`
`isFull()`	Verifica se está cheio	`boolean full = stack.isFull();`
`size()`	Retorna o número de elementos	`int size = stack.size();`
`count()`	Contagem de elementos (alternativa)	`int count = stack.count();`
`clear()`	Remove todos os elementos	`stack.clear();`

1.3 Implementação do zero

Essa implementação provavelmente está incompleta.

```java
// Stack.java
public class Stack {
    private int[] data;
    private int count;

    // Construtor para inicializar a pilha com um tamanho específico
    public Stack(int size) {
        data = new int[size];
        count = 0;
    }

    // Adiciona um elemento ao topo da pilha
    public void push(int value) {
        if (count == data.length) {
            System.out.println("Erro: pilha cheia.");
            return;
        }
        data[count++] = value;
    }

    // Remove e retorna o elemento do topo da pilha
    public int pop() {
        if (isEmpty()) {
            System.out.println("Erro: pilha vazia.");
            throw new RuntimeException("Pilha vazia");
        }
        return data[--count];
    }

    // Retorna o elemento do topo da pilha sem removê-lo
    public int peek() {
        if (isEmpty()) {
            System.out.println("Erro: pilha vazia.");
            throw new RuntimeException("Pilha vazia");
        }
        return data[count - 1];
    }

    // Verifica se a pilha está vazia
    public boolean isEmpty() {
        return count == 0;
    }
```

```java
// Retorna o número de elementos na pilha
public int size() {
    return count;
}

// Remove todos os elementos da pilha
public void clear() {
    count = 0;
}

// Representa a pilha como uma string
@Override
public String toString() {
    StringBuilder sb = new StringBuilder("Pilha: [");
    for (int i = 0; i < count; i++) {
        sb.append(data[i]);
        if (i < count - 1) {
            sb.append(", ");
        }
    }
    sb.append("]");
    return sb.toString();
}

// Método principal para teste
public static void main(String[] args) {
    Stack stack = new Stack(5);

    // Adiciona elementos à pilha
    stack.push(1);
    stack.push(2);
    stack.push(3);

    // Exibe a pilha
    System.out.println(stack);

    // Exibe o topo da pilha
    System.out.println("Topo da pilha: " + stack.peek());

    // Remove e exibe elementos da pilha
    while (!stack.isEmpty()) {
        System.out.println("Removido: " + stack.pop());
    }

    // Exibe se a pilha está vazia
    System.out.println("A pilha está vazia? " + stack.isEmpty());
```

```
        // Exibe a pilha após limpar
        stack.clear();
        System.out.println(stack);
    }
}
```