**ATIVIDADE 5**

**ATENÇÃO:**

* **Esta Atividade deverá ser feita em GRUPO DE PELO MENOS 04 ALUNOS E DE NO MÁXIMO 08 ALUNOS embora a entrega deverá ser feita INDIVIDUALMENTE no Classroom.**
* **Atividades feitas individualmente ou entregues com atraso NÃO SERÃO CONSIDERADAS.**

**Grupo**

Rafael Rossetto Guitarrari RA : 823158602

Andrey de Freitas Souza RA : 823217536

Gabriel Farah De lima RA: 822231424

Fabrício de Barros Narbon RA:822227166

Bianca Alves Ribeiro RA: 8222240261

Luiz Gustavo França de Abreu RA: 823210075

Gabrielle Garcia Paz  RA: 823126085

Webster Diógenes Rodrigues RA:8222242764

Leia o **Tópico 10.5** do Livro Lógica de Programação e Estrutura de Dados com aplicações em Java, disponível em [**https://jorgestreet.com.br/wp-content/uploads/2019/04/logica\_de\_programacao\_estrutura\_dados\_java.pdf**](https://jorgestreet.com.br/wp-content/uploads/2019/04/logica_de_programacao_estrutura_dados_java.pdf). Em seguida, implementar **em Java e em Python** os exercícios abaixo.

1) Desenvolva um programa para inverter a posição dos elementos de uma pilha P.

2. Desenvolva um programa para testar se uma pilha P1 tem mais elementos que uma pilha P2.

3) Desenvolva um programa para testar se duas pilhas P1 e P2 são iguais.

4) Faça um programa que permita o usuário digitar 7 números e que serão colocados em uma pilha P. Em seguida, o usuário deverá digitar mais um número x. Procure se o número x está na pilha e imprima a resposta.

5) Faça um programa que permita ordenar os elementos de uma pilha em ordem crescente e decrescente.

6) Faça um programa que, dado duas pilhas com 4 elementos cada uma, form uma terceira pilha com 8 elementos empilhando os elementos em ordem decrescente.

**RESPOSTAS**

import java.util.\*;

public class PilhaExercicios {

    // 1) Inverter a posição dos elementos de uma pilha

    public static <T> Stack<T> inverterPilha(Stack<T> pilha) {

        Stack<T> pilhaAux = **new** Stack<>();

        while (!pilha.isEmpty()) {

            pilhaAux.push(pilha.pop());

        }

        return pilhaAux;

    }

    // 2) Testar se uma pilha P1 tem mais elementos que uma pilha P2

    public static boolean pilhaMaior(Stack<?> p1, Stack<?> p2) {

        return p1.size() > p2.size();

    }

    // 3) Testar se duas pilhas P1 e P2 são iguais

    public static boolean pilhasIguais(Stack<?> p1, Stack<?> p2) {

        return p1.equals(p2);

    }

    // 4) Procurar um número x em uma pilha

    public static boolean procurarNumero(Stack<Integer> pilha, int x) {

        return pilha.contains(x);

    }

    // 5) Ordenar os elementos de uma pilha em ordem crescente e decrescente

    public static Stack<Integer> ordenarPilha(Stack<Integer> pilha, boolean crescente) {

        List<Integer> lista = **new** ArrayList<>(pilha);

        Collections.sort(lista);

        if (!crescente) {

            Collections.reverse(lista);

        }

        Stack<Integer> pilhaOrdenada = **new** Stack<>();

        for (Integer num : lista) {

            pilhaOrdenada.push(num);

        }

        return pilhaOrdenada;

    }

    // 6) Formar uma terceira pilha com 8 elementos em ordem decrescente

    public static Stack<Integer> combinarPilhasDecrescente(Stack<Integer> p1, Stack<Integer> p2) {

        Stack<Integer> combinada = **new** Stack<>();

        combinada.addAll(p1);

        combinada.addAll(p2);

        return ordenarPilha(combinada, false);

    }

    public static void main(String[] args) {

        Scanner scanner = **new** Scanner(System.in);

        Stack<Integer> pilhaP = **new** Stack<>();

        System.out.println("Digite 7 números para empilhar:");

        for (int i = 0; i < 7; i++) {

            pilhaP.push(scanner.nextInt());

        }

        System.out.print("Digite um número para procurar na pilha: ");

        int x = scanner.nextInt();

        boolean encontrado = procurarNumero(pilhaP, x);

        System.out.println("Número " + x + (encontrado ? " está " : " não está ") + "na pilha.");

        Stack<Integer> pilhaCrescente = ordenarPilha(pilhaP, true);

        Stack<Integer> pilhaDecrescente = ordenarPilha(pilhaP, false);

        System.out.println("Pilha em ordem crescente: " + pilhaCrescente);

        System.out.println("Pilha em ordem decrescente: " + pilhaDecrescente);

        Stack<Integer> pilha1 = **new** Stack<>();

        Stack<Integer> pilha2 = **new** Stack<>();

        System.out.println("Digite 4 números para a primeira pilha:");

        for (int i = 0; i < 4; i++) {

            pilha1.push(scanner.nextInt());

        }

        System.out.println("Digite 4 números para a segunda pilha:");

        for (int i = 0; i < 4; i++) {

            pilha2.push(scanner.nextInt());

        }

        Stack<Integer> pilhaCombinada = combinarPilhasDecrescente(pilha1, pilha2);

        System.out.println("Pilha combinada em ordem decrescente: " + pilhaCombinada);

        scanner.close();

    }

}