**Aula 4 – Pilha**

**Aplicando Conhecimento**

**Template**

Nome do Aluno: Rebeca Abreu de Almeida

RA: 10233223

Ao realizar o solicitado, publique os códigos fontes criados neste *template* com os testes realizados. Depois, entregar este *template* com a sua solução e testes junto ao código fonte em Java completo, todos em um único arquivo compactado.

**Resolução e Testes**

Publique aqui SOMENTE os códigos fontes desenvolvidos e os testes realizados.

    public *int* search(*int* *e*) {

        for (*int* i = topo; i >= 0; i--) {

            if (*this*.e[i] == *e*) {

                return topo - i;

            }

        }

        return -1;

    }

    // realiza a inversão do conteúdo da Pilha

    // o elemento do topo deve ficar na base e

    // o da base deve se tornar o do topo

    public *void* inverts() throws Exception {

*int*[] temp = **new** *int*[*this*.size()];

*int* i = 0;

        // Desempilha todos os elementos em um array temporário

        while (!*this*.isEmpty()) {

            temp[i++] = *this*.pop();

        }

        // Reempilha os elementos em ordem inversa

        for (*int* j = 0; j < temp.length; j++) {

*this*.push(temp[j]);

        }

    }

    // remove e retorna os elementos ímpares (type = 1) ou

    // pares (type = 2) da Pilha, mantendo os seus outros elementos

    // na ordem original. Caso não seja fornecido o

    // type = 1 ou 2, retorna uma exceção com a mensagem

    // "The parameter to the popEvenOdd method must be 1 for odd and 2 for even"

    public *void* popEvenOdd(*int* *type*) throws Exception {

        if (*type* != 1 && *type* != 2) {

            throw **new** Exception("The parameter to the popEvenOdd method must be 1 for odd and 2 for even");

        }

        Pilha temp = **new** Pilha(*this*.size());

        // Remove os elementos e filtra pares ou ímpares

        while (!*this*.isEmpty()) {

*int* elem = *this*.pop();

            if (*type* == 1 && elem % 2 == 0) {

                temp.push(elem);

            } else if (*type* == 2 && elem % 2 != 0) {

                temp.push(elem);

            }

        }

        // Reempilha os elementos que sobraram

        while (!temp.isEmpty()) {

*this*.push(temp.pop());

        }

    }

    // remove os elementos da Pilha que

    // são múltiplos de um certo número (nro)

    // passado como parâmetro, deixando os

    // outros na ordem original.

    public *void* popMultiple(*int* *nro*) throws Exception {

        Pilha temp = **new** Pilha(*this*.size());

        // Remove os elementos e filtra os múltiplos de 'nro'

        while (!*this*.isEmpty()) {

*int* elem = *this*.pop();

            if (elem % *nro* != 0) {

                temp.push(elem);

            }

        }

        // Reempilha os elementos que sobraram

        while (!temp.isEmpty()) {

*this*.push(temp.pop());

        }

    }

Texto branco sobre fundo preto

Descrição gerada automaticamente