UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática
Prof. Dr. Ivan Carlos Alcântara de Oliveira
Estruturas de Dados – 3º ADS - EAD

Aula 4 – Pilha
Aplicando Conhecimento
Template

Nome do Aluno: João Pedro Lima Lustosa Amorim

RA: 10289920

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]);
        }
    }
```

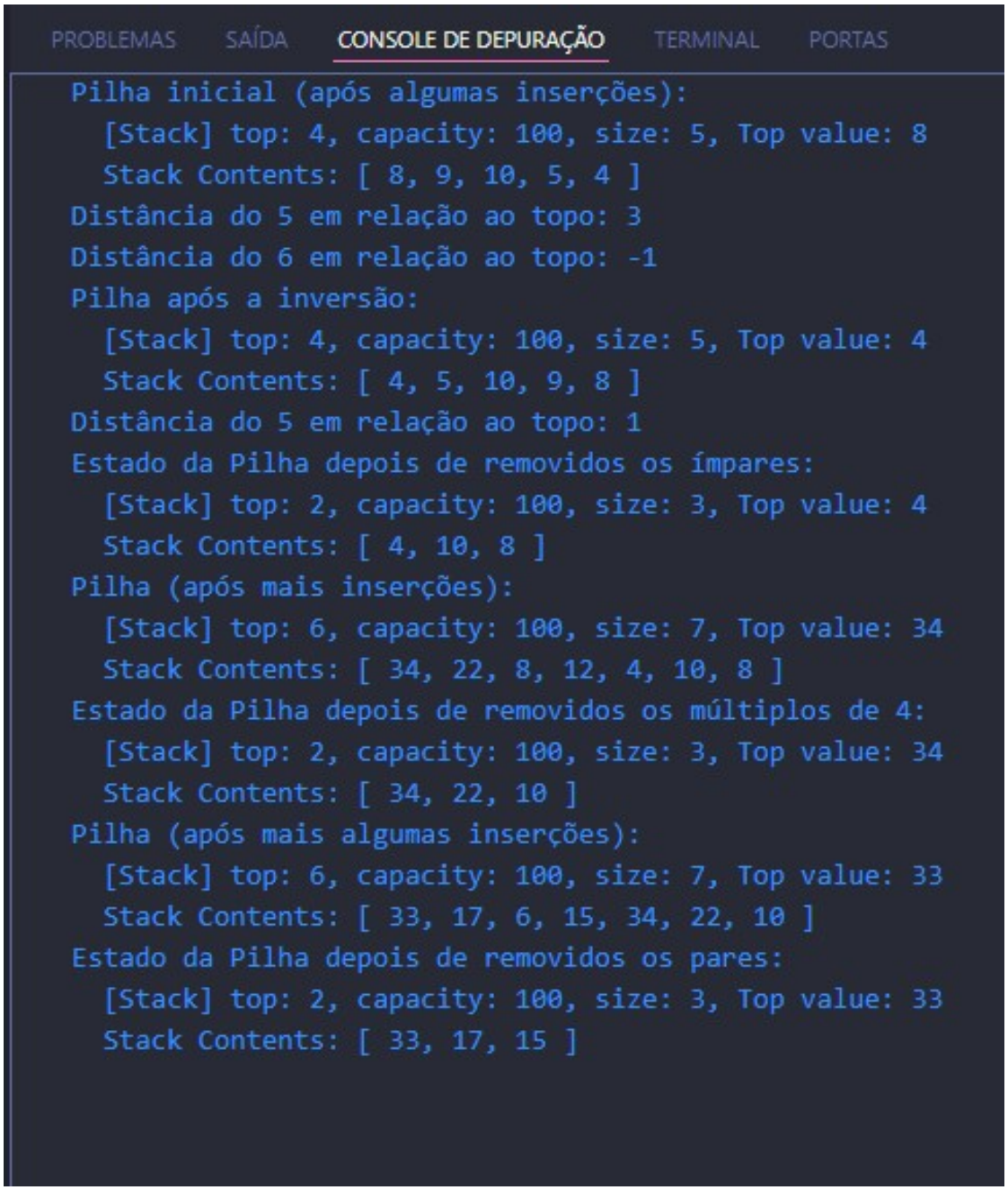
```
    // remove e retorna os elementos ímpares (type = 1) ou
    // pares (type = 2) da Pilha, mantendo os seus outros elementos
    // na ordem original. Caso não seja fornecido o
    // type = 1 ou 2, retorna uma exceção com a mensagem
    // "The parameter to the popEvenOdd method must be 1 for 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());
        }
    }
```


PROBLEMAS
SAÍDA
CONSOLE DE DEPURAÇÃO
TERMINAL
PORTAS
Pilha inicial (após algumas inserções):
[Stack] top: 4, capacity: 100, size: 5, Top value: 8
Stack Contents: [8, 9, 10, 5, 4]
Distância do 5 em relação ao topo: 3
Distância do 6 em relação ao topo: -1
Pilha após a inversão:
[Stack] top: 4, capacity: 100, size: 5, Top value: 4
Stack Contents: [4, 5, 10, 9, 8]
Distância do 5 em relação ao topo: 1
Estado da Pilha depois de removidos os ímpares:
[Stack] top: 2, capacity: 100, size: 3, Top value: 4
Stack Contents: [4, 10, 8]
Pilha (após mais inserções):
[Stack] top: 6, capacity: 100, size: 7, Top value: 34
Stack Contents: [34, 22, 8, 12, 4, 10, 8]
Estado da Pilha depois de removidos os múltiplos de 4:
[Stack] top: 2, capacity: 100, size: 3, Top value: 34
Stack Contents: [34, 22, 10]
Pilha (após mais algumas inserções):
[Stack] top: 6, capacity: 100, size: 7, Top value: 33
Stack Contents: [33, 17, 6, 15, 34, 22, 10]
Estado da Pilha depois de removidos os pares:
[Stack] top: 2, capacity: 100, size: 3, Top value: 33
Stack Contents: [33, 17, 15]