FILA FLEXÍVEL

QUESTÃO 1

public class Celula {

    public String elemento;

    public Celula prox;

    public Celula() {

        this(" ");

    }

    public Celula(String x) {

        this.elemento = x;

        this.prox = null;

    }

}

public class FilaFlexivel {

    private Celula primeiro, ultimo;

    public FilaFlexivel() {

        primeiro = new Celula();

        ultimo = primeiro;

    }

    public void inserir(String x) {

        ultimo.prox = new Celula(x);

        ultimo = ultimo.prox;

    }

    public String remover() throws Exception {

        if (primeiro == ultimo) {

            throw new Exception("Erro! A Fila Está vazia");

        }

        Celula tmp = primeiro;

        primeiro = primeiro.prox;

        String elemento = primeiro.elemento;

        tmp.prox = null;

        tmp = null;

        return elemento;

    }

    public void mostrar() {

        System.out.print("[");

        for (Celula i = primeiro.prox; i != null; i = i.prox) {

            System.out.print(i.elemento + " ");

        }

        System.out.print("]");

    }

    public boolean pesquisar(String x) {

        for (Celula i = primeiro.prox; i != null; i = i.prox) {

            if (i.elemento.equals(x)) {

                return true;

            }

        }

        return false;

    }

    public boolean isVazia() {

        return primeiro == ultimo;

    }

}

import java.util.\*;

public class Teste {

    public static void main(String[] args) throws Exception {

        Scanner sc = new Scanner(System.in);

        FilaFlexivel fi = new FilaFlexivel();

        int op = 0;

        String nomes;

        do {

            System.out.println("\nMenu:");

            System.out.println("1) Inserir cliente na fila de espera");

            System.out.println("2) Remover um cliente da fila de espera");

            System.out.println("3) Listar os nomes dos clientes que estão na fila de espera");

            System.out.println("4) Pesquisar se o cliente está na fila de espera");

            System.out.println("5) Verificar se a fila está vazia");

            System.out.println("6) Encerrar o programa");

            System.out.print("Opção:");

            op = sc.nextInt();

            switch (op) {

                case 1:

                    System.out.println("\nDigite o nome");

                    nomes = sc.next();

                    fi.inserir(nomes);

                    break;

                case 2:

                    fi.remover();

                    break;

                case 3:

                    fi.mostrar();

                    break;

                case 4:

                    System.out.println("Digite o nome a pesquisar");

                    nomes = sc.next();

                    fi.pesquisar(nomes);

                    if (fi.pesquisar(nomes)) {

                        System.out.println("Esse nome está na lista");

                    } else {

                        System.out.println("Esse nome não existe na lista");

                    }

                    break;

                case 5:

                    if (fi.isVazia()) {

                        System.out.println("Fila Vazia");

                    } else {

                        System.out.println("Fila contém elementos");

                    }

                    break;

                default:

                    System.out.println("opção Inválida");

                    break;

            }

        } while (op != 6);

    }

}

Questão 2:

public class Celula {

    public Cliente elemento;

    public Celula prox;

    public Celula() {

        this(null);

    }

    public Celula(Cliente x) {

        this.elemento = x;

        this.prox = null;

    }

}

    public class Cliente {

        private String nome;

        private String cfp;

        private String endereco;

        private String telefone;

        public Cliente(String nome, String cpf, String endereço, String telefone) {

            this.nome = nome;

            this.cfp = cpf;

            this.endereco = endereço;

            this.telefone = telefone;

        }

        public void setNome(String nome) {

            this.nome = nome;

        }

        public String getNome() {

            return nome;

        }

        public void setCPF(String cpf) {

            this.cfp = cpf;

        }

        public String getCPF() {

            return cfp;

        }

        public void setEndereco(String endereço) {

            this.endereco = endereço;

        }

        public String getEndereco() {

            return endereco;

        }

        public void setTelefone(String telefone) {

            this.telefone = telefone;

        }

        public String getTelefone() {

            return telefone;

        }

        @Override

        public String toString() {

            return "Cliente [nome=" + nome + ", cfp=" + cfp + ", endereco=" + endereco + ", telefone=" + telefone + "]" + "\n";

        }

    }

public class FilaFlexivel {

    private Celula primeiro, ultimo;

    public FilaFlexivel() {

        primeiro = new Celula();

        ultimo = primeiro;

    }

    public void inserir(Cliente x) {

        ultimo.prox = new Celula(x);

        ultimo = ultimo.prox;

    }

    public Cliente remover() throws Exception {

        if (primeiro == ultimo) {

            throw new Exception("Erro! A Fila Está vazia");

        }

        Celula tmp = primeiro;

        primeiro = primeiro.prox;

        Cliente elemento = primeiro.elemento;

        tmp.prox = null;

        tmp = null;

        return elemento;

    }

    public void mostrarFila() {

        for (Celula i = primeiro.prox; i != null; i = i.prox) {

            System.out.print(i.elemento + " ");

        }

    }

    public boolean pesquisar(String x) {

        for (Celula i = primeiro.prox; i != null; i = i.prox) {

            if (i.elemento.getNome().equals(x)) {

                return true;

            }

        }

        return false;

    }

    public boolean isVazia() {

        return primeiro == ultimo;

    }

}

import java.util.\*;

public class Teste {

    public static void main(String[] args) throws Exception {

        Scanner sc = new Scanner(System.in);

        int op;

        String nome, endereco, cpf, telefone;

        FilaFlexivel fi = new FilaFlexivel();

        do {

            System.out.println("\n Menu");

            System.out.println("1- Para pesquisar se existe o nome");

            System.out.println("2- Mostrar Filas de Objetos");

            System.out.println("3- Para Inserir Objeto");

            System.out.println("4- Para remover Objeto");

            System.out.println("5- Encerrar o programa");

            System.out.print("Opção:");

            op = sc.nextInt();

            switch (op) {

                case 1:

                    System.out.println("\n Digite o nome");

                    nome = sc.next();

                    if (fi.pesquisar(nome)) {

                        System.out.println("\n Objeto com nome " + nome + " encontrado!");

                    } else {

                        System.out.println("Esse Objeto não Existe!");

                    }

                    break;

                case 2:

                    fi.mostrarFila();

                    break;

                case 3:

                    sc.nextLine();

                    System.out.print("Nome:");

                    nome = sc.nextLine();

                    System.out.print("CPF:");

                    cpf = sc.nextLine();

                    System.out.print("Endereço:");

                    endereco = sc.nextLine();

                    System.out.print("Telefone:");

                    telefone = sc.nextLine();

                    Cliente clienteNovo = new Cliente(nome, cpf, endereco, telefone);

                    fi.inserir(clienteNovo);

                    break;

                case 4:

                    fi.remover();

                    break;

                default:

                    System.out.println("Opção Inválida");

                    break;

            }

        } while (op != 5);

        System.out.println("Fim");

        sc.close();

    }

}

Questão 3:

public class Aviao {

    private String nome;

    private int id;

    public Aviao(String nome, int id) {

        this.nome = nome;

        this.id = id;

    }

    public String getNome() {

        return nome;

    }

    public void setNome(String nome) {

        this.nome = nome;

    }

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

}

public class Celula {

    public Aviao elemento;

    public Celula prox;

    public Celula() {

    }

    public Celula(Aviao x) {

        this.elemento = x;

        this.prox = null;

    }

}

class FilaFlexivel {

    private Celula primeiro, ultimo;

    public FilaFlexivel() {

        primeiro = new Celula();

        ultimo = primeiro;

    }

    public void inserir(Aviao x) {

        ultimo.prox = new Celula(x);

        ultimo = ultimo.prox;

    }

    public Aviao remover() throws Exception {

        if (primeiro == ultimo)

            throw new Exception("Erro!");

        Celula tmp = primeiro;

        primeiro = primeiro.prox;

        Aviao elemento = primeiro.elemento;

        tmp.prox = null;

        tmp = null;

        return elemento;

    }

    public void listarTodos() {

        System.out.print("[ ");

        for (Celula i = primeiro.prox; i != null; i = i.prox) {

            System.out.print("Nome: " + i.elemento.getNome() + " Id: " + i.elemento.getId() + ", ");

        }

        System.out.println("]");

    }

    public void listarPrimeiro() {

        System.out.println("[Nome: " + primeiro.elemento.getNome() + " Id: " + primeiro.elemento.getId() + "]");

    }

    public int length() {

        int t = 0;

        for (Celula i = primeiro; i != ultimo; i = i.prox) {

            t++;

        }

        return t;

    }

}

import java.util.Scanner;

public class Teste {

    public static void main(String[] args) throws Exception {

        Scanner sc = new Scanner(System.in);

        int op, id;

        String nome;

        FilaFlexivel f = new FilaFlexivel();

        do {

            System.out.println("Menu de opções");

            System.out.println("1) Listar quantidade de aviões que estão aguardando na fila de decolagem");

            System.out.println("2) Autorizar a decolagem do primeiro avião da fila de decolagem");

            System.out.println("3) Adicionar um avião à fila de decolagem");

            System.out.println("4) Listar o identificador de todos os aviões da fila de decolagem");

            System.out.println("5) Lista o nome e o identificador do primeiro avião da fila de colagem");

            System.out.println("6) Encerrar o programa");

            op = sc.nextInt();

            sc.nextLine();

            switch (op) {

                case 1:

                    System.out.println(f.length());

                    break;

                case 2:

                    f.remover();

                    System.out.println("Avião autorizado para decolar!");

                    break;

                case 3:

                    System.out.println("Digite o nome do avião: ");

                    nome = sc.nextLine();

                    System.out.println("Digite o id do avião: ");

                    id = sc.nextInt();

                    Aviao a = new Aviao(nome, id);

                    f.inserir(a);

                    break;

                case 4:

                    f.listarTodos();

                    break;

                case 5:

                    f.listarPrimeiro();

                    break;

                case 6:

                    System.out.println("Programa encerrado.");

                    break;

                default:

                    System.out.println("Opção inválida.");

                    break;

            }

        } while (op != 6);

        sc.close();

    }

}

Lista Flexível Simples

Questão 1:

public class Celula {

    public double elemento;

    public Celula prox;

    public Celula() {

        this(0);

    }

    public Celula(double x) {

        this.elemento = x;

        this.prox = null;

    }

}

public class ListaFlexivel {

    private Celula primeiro, ultimo;

    public ListaFlexivel() {

        primeiro = new Celula();

        ultimo = primeiro;

    }

    public void inserirInicio(double x) {

        Celula tmp = new Celula(x);

        tmp.prox = primeiro.prox;

        primeiro.prox = tmp;

        if (primeiro == ultimo) {

            ultimo = tmp;

        }

        tmp = null;

    }

    public void inserirFim(double x) {

        ultimo.prox = new Celula(x);

        ultimo = ultimo.prox;

    }

    public double removerInicio() throws Exception {

        if (primeiro == ultimo)

            throw new Exception("Erro!");

        Celula tmp = primeiro;

        primeiro = primeiro.prox;

        double elemento = primeiro.elemento;

        tmp.prox = null;

        tmp = null;

        return elemento;

    }

    public double removerFim() throws Exception {

        if (primeiro == ultimo)

            throw new Exception("Erro!");

        Celula i;

        for (i = primeiro; i.prox != ultimo; i = i.prox)

            ;

        double elemento = ultimo.elemento;

        ultimo = i;

        ultimo.prox = null;

        i = null;

        return elemento;

    }

    public void inserir(double x, int pos) throws Exception {

        int tamanho = tamanho();

        if (pos < 0 || pos > tamanho) {

            throw new Exception("Erro!");

        } else if (pos == 0) {

            inserirInicio(x);

        } else if (pos == tamanho) {

            inserirFim(x);

        } else {

            Celula i = primeiro;

            for (int j = 0; j < pos; j++, i = i.prox)

                ;

            Celula tmp = new Celula(x);

            tmp.prox = i.prox;

            i.prox = tmp;

            tmp = null;

            i = null;

        }

    }

    public double remover(double pos) throws Exception {

        double elemento, tamanho = tamanho();

        if (primeiro == ultimo || pos < 0 || pos >= tamanho) {

            throw new Exception("Erro!");

        } else if (pos == 0) {

            elemento = removerInicio();

        } else if (pos == tamanho - 1) {

            elemento = removerFim();

        } else {

            Celula i = primeiro;

            for (int j = 0; j < pos; j++, i = i.prox)

                ;

            Celula tmp = i.prox;

            elemento = tmp.elemento;

            i.prox = tmp.prox;

            tmp.prox = null;

            tmp = null;

            i = null;

        }

        return elemento;

    }

    public int tamanho() {

        int tamanho = 0;

        for (Celula i = primeiro; i != ultimo; i = i.prox)

            tamanho++;

        return tamanho;

    }

    public void mostrar() {

        System.out.print("[");

        for (Celula i = primeiro.prox; i != null; i = i.prox) {

            System.out.print(i.elemento + " - ");

        }

        System.out.print("]");

    }

    public int pesquisar(double temperatura) {

        int quantidade = 0;

        for (Celula i = primeiro.prox; i != null; i = i.prox) {

            if (i.elemento == temperatura) {

                quantidade++;

            }

        }

        return quantidade;

    }

    public boolean isVazia() {

        return primeiro == ultimo;

    }

}

import java.util.Scanner;

public class Teste{

    public static void main(String[] args) throws Exception {

        Scanner sc = new Scanner(System.in);

        ListaFlexivel lista = new ListaFlexivel();

        int opcao = 0;

        do {

            System.out.println("----- Menu de Opções -----");

            System.out.println("1 - Inserir temperatura no início");

            System.out.println("2 - Inserir temperatura no final");

            System.out.println("3 - Inserir temperatura numa posição específica");

            System.out.println("4 - Remover a primeira temperatura");

            System.out.println("5 - Remover a última temperatura");

            System.out.println("6 - Remover de uma posição específica");

            System.out.println("7 - Mostrar todas temperaturas");

            System.out.println("8 - Mostrar o número de dias que uma temperatura específica foi registrada");

            System.out.println("9 - Verificar se a lista está vazia");

            System.out.println("10 - Encerrar o programa");

            System.out.print("Digite a opção desejada: ");

            opcao = sc.nextInt();

            switch (opcao) {

                case 1:

                    System.out.print("Digite a temperatura: ");

                    double tempInicio = sc.nextDouble();

                    lista.inserirInicio(tempInicio);

                    break;

                case 2:

                    System.out.print("Digite a temperatura: ");

                    double tempFim = sc.nextDouble();

                    lista.inserirFim(tempFim);

                    break;

                case 3:

                    System.out.print("Digite a temperatura: ");

                    double tempPos = sc.nextDouble();

                    System.out.print("Digite a posição: ");

                    int pos = sc.nextInt();

                    lista.inserir(tempPos, pos);

                    break;

                case 4:

                    double tempRemovidoInicio = lista.removerInicio();

                    System.out.println("Temperatura removida: " + tempRemovidoInicio);

                    break;

                case 5:

                    double tempRemovidoFim = lista.removerFim();

                    System.out.println("Temperatura removida: " + tempRemovidoFim);

                    break;

                case 6:

                    System.out.print("Digite a posição: ");

                    int posRemover = sc.nextInt();

                    double tempRemovido = lista.remover(posRemover);

                    System.out.println("Temperatura removida: " + tempRemovido);

                    break;

                case 7:

                    System.out.print("Temperaturas registradas: ");

                    lista.mostrar();

                    System.out.println();

                    break;

                case 8:

                    System.out.print("Digite a temperatura: ");

                    double tempPesquisar = sc.nextDouble();

                    int dias = lista.pesquisar(tempPesquisar);

                    System.out.println("A temperatura " + tempPesquisar + " foi registrada " + dias + " dias.");

                    break;

                case 9:

                    if (lista.isVazia()) {

                        System.out.println("A lista está vazia.");

                    } else {

                        System.out.println("A lista não está vazia.");

                    }

                    break;

                case 10:

                    System.out.println("Programa encerrado.");

                    break;

                default:

                    System.out.println("Opção inválida. Digite novamente.");

                    break;

            }

        } while (opcao != 10);

        sc.close();

    }

}

Questão 2

public class Celula {

    public Site elemento;

    public Celula prox;

    public Celula() {

        this(null);

    }

    public Celula(Site x) {

        this.elemento = x;

        this.prox = null;

    }

}

public class ListaFlexivel {

    private Celula primeiro, ultimo;

    public ListaFlexivel() {

        primeiro = new Celula();

        ultimo = primeiro;

    }

    public void inserirInicio(Site x) {

        Celula tmp = new Celula(x);

        tmp.prox = primeiro.prox;

        primeiro.prox = tmp;

        if (primeiro == ultimo) {

            ultimo = tmp;

        }

        tmp = null;

    }

    public void inserirFim(Site x) {

        ultimo.prox = new Celula(x);

        ultimo = ultimo.prox;

    }

    public Site removerInicio() throws Exception {

        if (primeiro == ultimo)

            throw new Exception("Erro!");

        Celula tmp = primeiro;

        primeiro = primeiro.prox;

        Site elemento = primeiro.elemento;

        tmp.prox = null;

        tmp = null;

        return elemento;

    }

    public Site removerFim() throws Exception {

        if (primeiro == ultimo)

            throw new Exception("Erro!");

        Celula i;

        for (i = primeiro; i.prox != ultimo; i = i.prox)

            ;

        Site elemento = ultimo.elemento;

        ultimo = i;

        ultimo.prox = null;

        i = null;

        return elemento;

    }

    public void inserir(Site x, int pos) throws Exception {

        int tamanho = tamanho();

        if (pos < 0 || pos > tamanho) {

            throw new Exception("Erro!");

        } else if (pos == 0) {

            inserirInicio(x);

        } else if (pos == tamanho) {

            inserirFim(x);

        } else {

            Celula i = primeiro;

            for (int j = 0; j < pos; j++, i = i.prox)

                ;

            Celula tmp = new Celula(x);

            tmp.prox = i.prox;

            i.prox = tmp;

            tmp = null;

            i = null;

        }

    }

    public Site remover(double pos) throws Exception {

        Site elemento;

        int tamanho = tamanho();

        if (primeiro == ultimo || pos < 0 || pos >= tamanho) {

            throw new Exception("Erro!");

        } else if (pos == 0) {

            elemento = removerInicio();

        } else if (pos == tamanho - 1) {

            elemento = removerFim();

        } else {

            Celula i = primeiro;

            for (int j = 0; j < pos; j++, i = i.prox)

                ;

            Celula tmp = i.prox;

            elemento = tmp.elemento;

            i.prox = tmp.prox;

            tmp.prox = null;

            tmp = null;

            i = null;

        }

        return elemento;

    }

    public int tamanho() {

        int tamanho = 0;

        for (Celula i = primeiro; i != ultimo; i = i.prox)

            tamanho++;

        return tamanho;

    }

    public void mostrar() {

        System.out.print("[");

        for (Celula i = primeiro.prox; i != null; i = i.prox) {

            System.out.print(i.elemento + " ");

        }

        System.out.print("]");

    }

    public boolean isVazia() {

        return primeiro == ultimo;

    }

    public String pesquisarLink(String nomeSite) {

        Celula i = primeiro;

 while (i != ultimo) {

            if (i.elemento.getNome().equalsIgnoreCase(nomeSite)) {

                String link = i.elemento.getLink();

                inserirInicio(i.elemento);

                return link;

            }

            i = i.prox;

        }

        return "Site não encontrado na lista!";

    }

}

public class Site {

    private String nome;

    private String link;

    public Site(String nome, String link) {

        this.nome = nome;

        this.link = link;

    }

    public String getNome() {

        return nome;

    }

    public String getLink() {

        return link;

    }

    public void setNome(String nome) {

        this.nome = nome;

    }

    public void setLink(String link) {

        this.link = link;

    }

    @Override

    public String toString() {

        return "Site [nome=" + nome + ", link=" + link + "] - ";

    }

}

import java.util.Scanner;

public class Teste {

public static void main(String[] args) throws Exception {

Scanner sc = new Scanner(System.in);

ListaFlexivel lista = new ListaFlexivel();

int op = 0;

do {

System.out.println("Menu de Opções:");

System.out.println("1. Inserir um Site no início da lista");

System.out.println("2. Inserir um Site no final da lista");

System.out.println("3. Inserir um Site numa posição específica da lista");

System.out.println("4. Remover o primeiro Site da lista (Imprimir o nome do site removido)");

System.out.println("5. Remover o último Site da lista (Imprimir o nome do site removido)");

System.out.println("6. Remover um Site de uma posição específica da lista (Imprimir o nome do site removido)");

System.out.println("7. Mostrar o nome e o link de todos os sites da lista");

System.out.println("8. Pesquisar o link de um site");

System.out.println("9. Encerrar o programa");

System.out.print("Opção:");

op = sc.nextInt();

switch (op) {

case 1:

System.out.println("Digite o nome do site:");

String nome = sc.next();

System.out.println("Digite o link do site:");

String link = sc.next();

Site siteInicio = new Site(nome, link);

lista.inserirInicio(siteInicio);

break;

case 2:

System.out.println("Digite o nome do site:");

String nome2 = sc.next();

System.out.println("Digite o link do site:");

String link2 = sc.next();

Site siteFim = new Site(nome2, link2);

lista.inserirFim(siteFim);

break;

case 3:

System.out.println("Digite o nome do site:");

String nome3 = sc.next();

System.out.println("Digite o link do site:");

String link3 = sc.next();

System.out.println("Digite a posição onde deseja inserir o site:");

int posicao = sc.nextInt();

Site sitePosicao = new Site(nome3, link3);

lista.inserir(sitePosicao, posicao);

break;

case 4:

try {

Site removidoInicio = lista.removerInicio();

System.out.println("Site removido: " + removidoInicio.getNome());

} catch (Exception e) {

System.out.println(e.getMessage());

}

break;

case 5:

try {

Site removidoFim = lista.removerFim();

System.out.println("Site removido: " + removidoFim.getNome());

} catch (Exception e) {

System.out.println(e.getMessage());

}

break;

case 6:

System.out.println("Digite a posição do site que deseja remover:");

int posicaoRemover = sc.nextInt();

try {

Site removidoPosicao = lista.remover(posicaoRemover);

System.out.println("Site removido: " + removidoPosicao.getNome());

} catch (Exception e) {

System.out.println(e.getMessage());

}

break;

case 7:

System.out.println("Lista de Sites:");

lista.mostrar();

System.out.println();

break;

case 8:

System.out.println("Digite o nome do site que deseja pesquisar:");

String nomePesquisa = sc.next();

String linkPesquisa = lista.pesquisarLink(nomePesquisa);

System.out.println(linkPesquisa);

break;

case 9:

System.out.println("Encerrando o programa.");

break;

default:

System.out.println("Opção inválida.");

break;

}

} while (op != 9);

}

}

Questão 3

public class Celula {

    public int elemento;

    public Celula prox;

    public Celula() {

        this(0);

    }

    public Celula(int x) {

        this.elemento = x;

        this.prox = null;

    }

}

public class ListaFlexivel {

    private Celula primeiro, ultimo;

    public ListaFlexivel() {

        primeiro = new Celula();

        ultimo = primeiro;

    }

    public void inserirFim(int x) {

        ultimo.prox = new Celula(x);

        ultimo = ultimo.prox;

    }

    public boolean removerPeloID(int codigoAluno) {

        Celula anterior = primeiro;

        Celula atual = primeiro.prox;

        while (atual != null) {

            if (atual.elemento == codigoAluno) {

                anterior.prox = atual.prox;

                if (atual == ultimo) {

                    ultimo = anterior;

                }

                atual.prox = null;

                atual = null;

                return true;

            }

            anterior = atual;

            atual = atual.prox;

        }

        return false;

    }

    public void mostrar() {

        System.out.print("[");

        for (Celula i = primeiro.prox; i != null; i = i.prox) {

            System.out.print(i.elemento + " ");

        }

        System.out.print("]");

    }

    public boolean pesquisar(int id) {

        boolean teste = false;

        for (Celula i = primeiro.prox; i != null; i = i.prox) {

            if (i.elemento == id) {

                teste = true;

            }

        }

        return teste;

    }

}

import java.util.Scanner;

public class Teste {

    public static void main(String[] args) throws Exception {

        Scanner sc = new Scanner(System.in);

        int op = 0;

        ListaFlexivel alunosMestrandos = new ListaFlexivel();

        ListaFlexivel alunosDoutorandos = new ListaFlexivel();

        do {

            System.out.println("Menu:");

            System.out.println("1) Inserir um aluno no final da lista de mestrandos");

            System.out.println("2) Inserir um aluno no final da lista de doutorandos");

            System.out.println("3) Remover um aluno específico da lista de mestrandos");

            System.out.println("4) Remover um aluno específico da lista de doutorandos");

            System.out.println("5) Listar os códigos dos alunos que estão na lista de mestrandos");

            System.out.println("6) Listar os códigos dos alunos que estão na lista de doutorandos");

            System.out.println("7) Pesquisar se um aluno específico está na lista de mestrandos");

            System.out.println("8) Pesquisar se um aluno específico está na lista de doutorandos");

            System.out.println("9) Encerrar o programa");

            op = sc.nextInt();

            switch (op) {

                case 1:

                    System.out.println("Digite o código do aluno de mestrado: ");

                    int codigoMestrado = sc.nextInt();

                    alunosMestrandos.inserirFim(codigoMestrado);

                    System.out.println("Aluno adicionado na lista de mestrado.");

                    break;

                case 2:

                    System.out.println("Digite o código do aluno de doutorado: ");

                    int codigoDoutorado = sc.nextInt();

                    alunosDoutorandos.inserirFim(codigoDoutorado);

                    System.out.println("Aluno adicionado na lista de doutorado.");

                    break;

                case 3:

                    System.out.println("Digite o código do aluno de mestrado a ser removido: ");

                    int codigoMestradoRemover = sc.nextInt();

                    boolean mestradoRemovido = alunosMestrandos.removerPeloID(codigoMestradoRemover);

                    if (mestradoRemovido) {

                        System.out.println("Aluno removido da lista de mestrado.");

                    } else {

                        System.out.println("Aluno não encontrado na lista de mestrado.");

                    }

                    break;

                case 4:

                    System.out.println("Digite o código do aluno de doutorado a ser removido: ");

                    int codigoDoutoradoRemover = sc.nextInt();

                    boolean doutoradoRemovido = alunosDoutorandos.removerPeloID(codigoDoutoradoRemover);

                    if (doutoradoRemovido) {

                        System.out.println("Aluno removido da lista de doutorado.");

                    } else {

                        System.out.println("Aluno não encontrado na lista de doutorado.");

                    }

                    break;

                case 5:

                    System.out.println("Lista de códigos dos alunos de mestrado:");

                    alunosMestrandos.mostrar();

                    System.out.println();

                    break;

                case 6:

                    System.out.println("Lista de códigos dos alunos de doutorado:");

                    alunosDoutorandos.mostrar();

                    System.out.println();

                    break;

                case 7:

                    System.out.println("Digite o código do aluno de mestrado a ser pesquisado: ");

                    int codigoMestradoPesquisar = sc.nextInt();

                    boolean mestradoEncontrado = alunosMestrandos.pesquisar(codigoMestradoPesquisar);

                    if (mestradoEncontrado) {

                        System.out.println("Aluno encontrado na lista de mestrado.");

                    } else {

                        System.out.println("Aluno não encontrado na lista de mestrado.");

                    }

                    break;

                case 8:

                    System.out.println("Digite o código do aluno de doutorado a ser pesquisado: ");

                    int codigoDoutoradoPesquisar = sc.nextInt();

                    boolean doutoradoEncontrado = alunosDoutorandos.pesquisar(codigoDoutoradoPesquisar);

                    if (doutoradoEncontrado) {

                        System.out.println("Aluno encontrado na lista de doutorado.");

                    } else {

                        System.out.println("Aluno não encontrado na lista de doutorado.");

                    }

                    break;

                default:

                    break;

            }

        } while (op != 9);

    }

}

Lista flexível duplamente encadeada

Questão 1

Método "removerInicio":

a. Verifica se a lista contém pelo menos dois elementos, caso contrário, lança uma exceção.

b. Cria uma nova referência para a primeira célula da lista.

c. Ajusta a referência "primeiro" para a próxima célula da lista.

d. Recupera o elemento da nova primeira célula e armazena em "elemento".

e. Remove a referência "tmp" para a antiga primeira célula.

f. Retorna o elemento removido.

Método "removerFim":

a. Verifica se a lista contém pelo menos dois elementos, caso contrário, lança uma exceção.

b. Recupera o elemento da última célula da lista e armazena em "elemento".

c. Ajusta a referência "ultimo" para a célula anterior da lista.

d. Remove a referência para a última célula da lista.

e. Retorna o elemento removido.

Método "inserir":

a. Verifica se a posição inserida está dentro dos limites válidos (de 0 até o tamanho da lista).

b. Caso a posição seja 0, chama o método "inserirInicio".

c. Caso a posição seja o tamanho da lista, chama o método "inserirFim".

d. Caso contrário, percorre a lista até a posição desejada e cria uma nova célula com o elemento "x".

e. Ajusta as referências para que a nova célula seja inserida na posição correta.

Método "remover":

a. Verifica se a lista contém pelo menos dois elementos, caso contrário, lança uma exceção.

b. Verifica se a posição removida está dentro dos limites válidos (de 0 até o tamanho da lista -1).

c. Caso a posição seja 0, chama o método "removerInicio".

d. Caso a posição seja a última célula, chama o método "removerFim".

e. Caso contrário, percorre a lista até a posição desejada e remove a célula correspondente.

f. Retorna o elemento removido.

Questão 2 :

public class CelulaDupla {

    public String elemento;

    public CelulaDupla prox;

    public CelulaDupla ant;

    public CelulaDupla() {

        this("");

    }

    public CelulaDupla(String x) {

        this.elemento = x;

        this.prox = null;

        this.ant = null;

    }

}

public class ListaDupla {

    private CelulaDupla primeiro, ultimo;

    public ListaDupla() {

        primeiro = new CelulaDupla();

        ultimo = primeiro;

    }

    public void inserirInicio(String x) {

        CelulaDupla tmp = new CelulaDupla(x);

        tmp.ant = primeiro;

        tmp.prox = primeiro.prox;

        primeiro.prox = tmp;

        if (primeiro == ultimo) {

            ultimo = tmp;

        } else {

            tmp.prox.ant = tmp;

        }

        tmp = null;

    }

 public void inserirFim(String x) {

        ultimo.prox = new CelulaDupla(x);

        ultimo.prox.ant = ultimo;

        ultimo = ultimo.prox;

    }

    public String removerInicio() throws Exception {

        if (primeiro == ultimo)

            throw new Exception("Erro!");

        CelulaDupla tmp = primeiro;

        primeiro = primeiro.prox;

        String elemento = primeiro.elemento;

        tmp.prox = null;

        primeiro.ant = null;

        tmp = null;

        return elemento;

    }

    public String removerFim() throws Exception {

        if (primeiro == ultimo)

            throw new Exception("Erro");

        String elemento = ultimo.elemento;

        ultimo = ultimo.ant;

        ultimo.prox.ant = null;

        ultimo.prox = null;

        return elemento;

    }

    public void inserir(String x, int pos) throws Exception {

        int tamanho = tamanho();

        if (pos < 0 || pos > tamanho) {

            throw new Exception("Erro!");

        } else if (pos == 0) {

            inserirInicio(x);

        } else if (pos == tamanho) {

            inserirFim(x);

        } else {

            CelulaDupla i = primeiro;

            for (int j = 0; j < pos; j++, i = i.prox)

                ;

            CelulaDupla tmp = new CelulaDupla(x);

            tmp.ant = i;

            tmp.prox = i.prox;

            tmp.ant.prox = tmp;

            tmp.prox.ant = tmp;

            tmp = i = null;

        }

    }

    public int tamanho() {

        int tamanho = 0;

        CelulaDupla i = primeiro.prox;

        while (i != null) {

            tamanho++;

            i = i.prox;

        }

        return tamanho;

    }

    public String removerPorPosicao(int pos) throws Exception {

        String elemento;

        int tamanho = tamanho();

        if (primeiro == ultimo) {

            throw new Exception("Erro!");

        } else if (pos < 0 || pos >= tamanho) {

            throw new Exception("Erro!");

        } else if (pos == 0) {

            elemento = removerInicio();

        } else if (pos == tamanho - 1) {

            elemento = removerFim();

        } else {

            CelulaDupla i = primeiro.prox;

            for (int j = 0; j < pos; j++, i = i.prox)

                ;

            i.ant.prox = i.prox;

            i.prox.ant = i.ant;

            elemento = i.elemento;

            i.prox = null;

            i.ant = null;

            i = null;

        }

        return elemento;

    }

    public void mostrar() {

        System.out.print("[ ");

        for (CelulaDupla i = primeiro.prox; i != null; i = i.prox) {

            System.out.print(i.elemento + " ");

        }

        System.out.println("] ");

    }

    public boolean removerMusicaPorNome(String musica) {

        CelulaDupla i = primeiro.prox;

        while (i != null) {

            if (i.elemento.equals(musica)) {

                i.ant.prox = i.prox;

                if (i.prox != null) {

                    i.prox.ant = i.ant;

                } else {

                    ultimo = i.ant;

                }

                i = null;

                return true;

            }

            i = i.prox;

        }

        return false;

    }

    public void mostrarInverso() {

        System.out.print("[ ");

        for (CelulaDupla i = ultimo; i != primeiro; i = i.ant) {

            System.out.print(i.elemento + " ");

        }

        System.out.println("] ");

    }

    public boolean pesquisarMusica(String musica) {

        CelulaDupla i = primeiro.prox;

        while (i != null) {

            if (i.elemento.equals(musica)) {

                return true;

            }

            i = i.prox;

        }

        return false;

    }

    public String pesquisarAnterior(String musica) {

        CelulaDupla i = primeiro.prox;

        while (i != null) {

            if (i.elemento.equals(musica)) {

                if (i.ant == primeiro) {

                    return "";

                } else {

                    return i.ant.elemento;

                }

            }

            i = i.prox;

        }

        return "";

    }

    public String pesquisarPosterior(String musica) {

        CelulaDupla i = primeiro.prox;

        while (i != null) {

            if (i.elemento.equals(musica)) {

                if (i.prox == null) {

                    return "";

                } else {

                    return i.prox.elemento;

                }

            }

            i = i.prox;

        }

        return "";

    }

}

import java.util.Scanner;

public class Teste {

    public static void main(String[] args) {

        ListaDupla liDupla = new ListaDupla();

        Scanner sc = new Scanner(System.in);

        int op = 0;

        do {

            System.out.println("Menu:");

            System.out.println("1. Inserir uma música no final da lista");

            System.out.println("2. Inserir uma música no início da lista");

            System.out.println("3. Inserir uma música numa posição específica da lista");

            System.out.println("4. Remover a música do início da lista");

            System.out.println("5. Remover a música do final da lista");

            System.out.println("6. Remover uma música de uma posição específica da lista");

            System.out.println("7. Remover uma música específica");

            System.out.println("8. Listar todas as músicas da lista");

            System.out.println("9. Listar todas as músicas da lista na ordem inversa");

            System.out.println("10. Pesquisar uma música na lista");

            System.out.println("11. Pesquisar música anterior");

            System.out.println("12. Pesquisar música posterior");

            System.out.println("13. Encerrar o programa");

            System.out.print("\nDigite sua opção: ");

            op = sc.nextInt();

            switch (op) {

                case 1:

                    System.out.print("\nDigite o nome da música: ");

                    String nomeMusica = sc.next();

                    liDupla.inserirFim(nomeMusica);

                    break;

                case 2:

                    System.out.print("\nDigite o nome da música: ");

                    nomeMusica = sc.next();

                    liDupla.inserirInicio(nomeMusica);

                    break;

                case 3:

                    System.out.print("\nDigite o nome da música: ");

                    nomeMusica = sc.next();

                    System.out.print("\nDigite a posição: ");

                    int pos = sc.nextInt();

                    try {

                        liDupla.inserir(nomeMusica, pos);

                    } catch (Exception e) {

                        System.out.println(e.getMessage());

                    }

                    break;

                case 4:

                    try {

                        String nomeMusicaRemovida = liDupla.removerInicio();

                        System.out.println("\nMúsica removida: " + nomeMusicaRemovida);

                    } catch (Exception e) {

                        System.out.println(e.getMessage());

                    }

                    break;

                case 5:

                    try {

                        String nomeMusicaRemovida = liDupla.removerFim();

                        System.out.println("\nMúsica removida: " + nomeMusicaRemovida);

                    } catch (Exception e) {

                        System.out.println(e.getMessage());

                    }

                    break;

                case 6:

                    System.out.print("\nDigite a posição: ");

                    pos = sc.nextInt();

                    try {

                        String nomeMusicaRemovida = liDupla.removerPorPosicao(pos);

                        System.out.println("\nMúsica removida: " + nomeMusicaRemovida);

                    } catch (Exception e) {

                        System.out.println(e.getMessage());

                    }

                    break;

                case 7:

                    System.out.println("Digite a música");

                    nomeMusica = sc.next();

                    Boolean nomeMusicaRemovida = liDupla.removerMusicaPorNome(nomeMusica);

                    if (nomeMusicaRemovida) {

                        System.out.println("Música Removida com Sucesso");

                    } else {

                        System.out.println("Não foi posível remover essa musica");

                    }

                    break;

                case 8:

                    liDupla.mostrar();

                    break;

                case 9:

                    liDupla.mostrarInverso();

                    break;

                case 10:

                    System.out.println("Digite a música");

                    nomeMusica = sc.next();

                    liDupla.pesquisarMusica(nomeMusica);

                    break;

                case 11:

                    System.out.println("Digite a música");

                    nomeMusica = sc.next();

                    System.out.println("A Música anterior é " + liDupla.pesquisarAnterior(nomeMusica));

                    break;

                case 12:

                    System.out.println("Digite a música");

                    nomeMusica = sc.next();

                    System.out.println("A Música Posterior é " + liDupla.pesquisarPosterior(nomeMusica));

                    break;

            }

        } while (op != 13);

    }

}

Pilha Flexível:

Questão 1 :

public class Celula {

    public int elemento;

    public Celula prox;

    public Celula() {

        this(0);

    }

    public Celula(int x) {

        this.elemento = x;

        this.prox = null;

    }

}

public class PilhaFlexivel {

    private Celula topo;

    public PilhaFlexivel() {

        topo = null;

    }

    public void inserir(int x) { // Inserir(5)

        Celula tmp = new Celula(x);

        tmp.prox = topo;

        topo = tmp;

        tmp = null;

    }

    public int remover() throws Exception {

        if (topo == null)

            throw new Exception("Erro!");

        int elemento = topo.elemento;

        Celula tmp = topo;

        topo = topo.prox;

        tmp.prox = null;

        tmp = null;

        return elemento;

    }

    public void mostrar() {

        System.out.print("[ ");

        for (Celula i = topo; i != null; i = i.prox) {

            System.out.print(i.elemento + " ");

        }

        System.out.println("]");

    }

}

import java.util.Scanner;

public class Teste {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num1 = 1, num2 = 1, n;

System.out.print("Digite um valor inteiro positivo n: ");

n = sc.nextInt();

PilhaFlexivel p = new PilhaFlexivel();

p.inserir(1);

p.inserir(1);

for (int i = 2; i < n; i++) {

int fib = num1 + num2;

p.inserir(fib);

num1 = num2;

num2 = fib;

}

p.mostrar();

sc.close();

}

}

Questão 2 :

public class Celula {

    public double elemento;

    public Celula prox;

    public Celula() {

        this(0);

    }

    public Celula(double x) {

        this.elemento = x;

        this.prox = null;

    }

}

public class PilhaFlexivel {

    private Celula topo;

    public PilhaFlexivel() {

        topo = null;

    }

    public void inserir(double x) { // Inserir(5)

        Celula tmp = new Celula(x);

        tmp.prox = topo;

        topo = tmp;

        tmp = null;

    }

    public double remover() throws Exception {

        if (topo == null)

            throw new Exception("Erro!");

        double elemento = topo.elemento;

        Celula tmp = topo;

        topo = topo.prox;

        tmp.prox = null;

        tmp = null;

        return elemento;

    }

    public void mostrar() {

        System.out.print("[ ");

        for (Celula i = topo; i != null; i = i.prox) {

            System.out.print(i.elemento + " ");

        }

        System.out.println("]");

    }

}

import java.util.Scanner;

public class Teste {

    public static void main(String[] args) throws Exception {

        Scanner sc = new Scanner(System.in);

        String expressao;

        System.out.print("Digite a expressão em notação polonesa reversa: ");

        expressao = sc.nextLine();

        PilhaFlexivel pi = new PilhaFlexivel();

        for (int i = 0; i < expressao.length(); i++) {

            char c = expressao.charAt(i);

            if (c == '+' || c == '-' || c == '\*' || c == '/') {

                double a = pi.remover();

                double b = pi.remover();

                double resultado = 0;

                switch (c) {

                    case '+':

                        resultado = b + a;

                        break;

                    case '-':

                        resultado = b - a;

                        break;

                    case '\*':

                        resultado = b \* a;

                        break;

                    case '/':

                        resultado = b / a;

                        break;

                }

                pi.inserir(resultado);

            } else {

                int valorFinal = Character.getNumericValue(c);

                pi.inserir(valorFinal);

            }

        }

        pi.mostrar();

    }

}

Questão 3:

public class Celula {

    public char elemento;

    public Celula prox;

    public Celula(char x) {

        this.elemento = x;

        this.prox = null;

    }

}

public class PilhaFlexivel {

    private Celula topo;

    public PilhaFlexivel() {

        topo = null;

    }

    public void inserir(char x) {

        Celula tmp = new Celula(x);

        tmp.prox = topo;

        topo = tmp;

        tmp = null;

    }

    public char remover() throws Exception {

        if (topo == null)

            throw new Exception("Pilha vazia!");

        char elemento = topo.elemento;

        Celula tmp = topo;

        topo = topo.prox;

        tmp.prox = null;

        tmp = null;

        return elemento;

    }

    public boolean vazia() {

        return topo == null;

    }

}

import java.util.Scanner;

public class Teste {

    public static void main(String[] args) throws Exception {

        Scanner sc = new Scanner(System.in);

        System.out.print("Digite uma sequência de parênteses e colchetes: ");

        String sequencia = sc.nextLine();

        PilhaFlexivel pilha = new PilhaFlexivel();

        for (int i = 0; i < sequencia.length(); i++) {

            char caractere = sequencia.charAt(i);

            if (caractere == '(' || caractere == '[') {

                pilha.inserir(caractere);

            }

            else if (caractere == ')' || caractere == ']') {

                char topo = pilha.remover();

                if ((caractere == ')' && topo == '(') || (caractere == ']' && topo == '[')) {

                    continue;

                } else {

                    System.out.println("Sequência malformada!");

                    return;

                }

            }

        }

        if (pilha.vazia()) {

            System.out.println("Sequência bem-formada!");

        } else {

            System.out.println("Sequência malformada!");

        }

    }

}