idade PSC Lista 9

a) Crie a classe BlocoDeNotas que possui como atributo um ArrayList<String> chamado notas. Crie métodos para inserir, remover e buscar notas. Crie um método que imprima todas as notas.

```
import java.util.ArrayList;

public class BlocoDeNotas {
    private ArrayList<String> notas;

    public BlocoDeNotas() {
        notas = new ArrayList<>();
    }

    public void inserirNota(String nota) {
        notas.add(nota);
    }

    public boolean removerNota(int indice) {
        if (indice >= 0 && indice < notas.size()) {
            notas.remove(indice);
            return true;
        }
        return false;
    }

    public String buscarNota(int indice) {
        if (indice >= 0 && indice < notas.size()) {
            return notas.get(indice);
        }
        return "Nota não encontrada!";
    }

    public void listarNotas() {
        System.out.println("Notas:");
        for (int i = 0; i < notas.size(); i++) {
            System.out.println((i + 1) + ". " + notas.get(i));
        }
    }
}
```

b) Crie a classe AppBloco, com um método main, e um menu que 1) Insira uma nota, 2) Remova uma nota, 3) Altere uma nota, 4) Listar todas as notas e 5) Saia do sistema.

```
import javax.swing.JOptionPane;

public class AppBloco {
    public static void main(String[] args) {
        BlocoDeNotas bloco = new BlocoDeNotas();
        int opcao;

        do {
            opcao = Integer.parseInt(JOptionPane.showInputDialog(
                "1. Inserir nota\n" +
                "2. Remover nota\n" +
                "3. Alterar nota\n" +
                "4. Listar todas as notas\n" +
                "5. Sair"));

            switch (opcao) {
                case 1:
                    String nota = JOptionPane.showInputDialog("Digite a nota:");
                    bloco.inserirNota(nota);
                    break;

                case 2:
                    int removerIndice = Integer.parseInt(JOptionPane.showInputDialog("Digite o índice da nota
para remover:")) - 1;
                    if (bloco.removerNota(removerIndice)) {
                        JOptionPane.showMessageDialog(null, "Nota removida.");
                    } else {
                        JOptionPane.showMessageDialog(null, "Índice inválido.");
                    }
                    break;

                case 3:
                    int alterarIndice = Integer.parseInt(JOptionPane.showInputDialog("Digite o índice da nota
para alterar:")) - 1;
                    String novaNota = JOptionPane.showInputDialog("Digite a nova nota:");
                    if (bloco.removerNota(alterarIndice)) {
                        bloco.inserirNota(novaNota);
                    } else {
                        JOptionPane.showMessageDialog(null, "Índice inválido.");
                    }
                    break;

                case 4:
                    bloco.listarNotas();
                    break;

                case 5:
                    JOptionPane.showMessageDialog(null, "Saindo...");
                    break;

                default:
                    JOptionPane.showMessageDialog(null, "Opção inválida!");
            }
        } while (opcao != 5);
    }
```

}

Problemas Propostos: Exercício 2
a) Você vai gerenciar um depósito e resolveu criar um sistema para isso. Para isso criou uma classe chamada Caixa, com os atributos corredor (String), posicao(int), peso(double) e dono(String), que armazena o nome do dono da caixa. Respeitou o encapsulamento e criou os métodos de acesso e os modificadores.

```
public class Caixa {
    private String corredor;
    private int posicao;
    private double peso;
    private String dono;

    public Caixa(String corredor, int posicao, double peso, String dono) {
        this.corredor = corredor;
        this.posicao = posicao;
        this.peso = peso;
        this.dono = dono;
    }

    public String getCorredor() {
        return corredor;
    }

    public void setCorredor(String corredor) {
        this.corredor = corredor;
    }

    public int getPosicao() {
        return posicao;
    }

    public void setPosicao(int posicao) {
        this.posicao = posicao;
    }

    public double getPeso() {
        return peso;
    }

    public void setPeso(double peso) {
        this.peso = peso;
    }

    public String getDono() {
        return dono;
```

```
    }

    @Override
    public String toString() {
        return "Caixa{" +
                "corredor='" + corredor + '\'' +
                ", posicao=" + posicao +
                ", peso=" + peso +
                ", dono='" + dono + '\'' +
                '}';
    }
}
```

b) Depois criou a classe Deposito, que contém um ArrayList de caixas. Fez um método para adicionar caixas e um para remover (pelo dono). Fez um método que encontra uma caixa pelo dono, retornando sua posição no arraylist (ou -1 se não achar). E um método para mudar o corredor e a posição de uma caixa, que encontra a caixa pelo dono e altera seu atributos. Ele fez também um método que retorna um vetor com a(s) caixa(s) que pesam mais do que um valor passado por parâmetro.

```
import java.util.ArrayList;

public class Deposito {
    private ArrayList<Caixa> caixas;

    public Deposito() {
        caixas = new ArrayList<>();
    }

    public void adicionarCaixa(Caixa caixa) {
        caixas.add(caixa);
    }

    public boolean removerCaixa(String dono) {
        return caixas.removeIf(caixa -> caixa.getDono().equals(dono));
    }

    public int procurarCaixa(String dono) {
        for (int i = 0; i < caixas.size(); i++) {
            if (caixas.get(i).getDono().equals(dono)) {
                return i;
            }
        }
        return -1;
    }

    public boolean alterarCaixa(String dono, String novoCorredor, int novaPosicao) {
        int indice = procurarCaixa(dono);
        if (indice != -1) {
```

```
            Caixa caixa = caixas.get(indice);
            caixa.setCorredor(novoCorredor);
            caixa.setPosicao(novaPosicao);
            return true;
        }
        return false;
    }

    public ArrayList<Caixa> listarCaixasPesadas(double pesoMinimo) {
        ArrayList<Caixa> caixasPesadas = new ArrayList<>();
        for (Caixa caixa : caixas) {
            if (caixa.getPeso() > pesoMinimo) {
                caixasPesadas.add(caixa);
            }
        }
        return caixasPesadas;
    }
}
```

c) Para testar seu sistema fez uma classe Teste com o método main que, usando o JOptionPane, possui um loop com as opções 1. adiciona caixa, 2. remove caixa, 3. procura caixa, 4. muda caixa, 5. lista mais pesadas que 10.0 e 6. sair.

```
import javax.swing.JOptionPane;

public class TesteDeposito {
    public static void main(String[] args) {
        Deposito deposito = new Deposito();
        int opcao;

        do {
            opcao = Integer.parseInt(JOptionPane.showInputDialog(
                "1. Adicionar caixa\n" +
                "2. Remover caixa\n" +
                "3. Procurar caixa\n" +
                "4. Alterar caixa\n" +
                "5. Listar caixas pesadas (> 10.0kg)\n" +
                "6. Sair"));

            switch (opcao) {
                case 1:
                    String corredor = JOptionPane.showInputDialog("Corredor:");
                    int posicao = Integer.parseInt(JOptionPane.showInputDialog("Posição:"));
                    double peso = Double.parseDouble(JOptionPane.showInputDialog("Peso:"));
                    String dono = JOptionPane.showInputDialog("Dono:");
                    deposito.adicionarCaixa(new Caixa(corredor, posicao, peso, dono));
                    break;

                case 2:
```

```
                String removerDono = JOptionPane.showInputDialog("Digite o nome do dono para
remover:");
                if (deposito.removerCaixa(removerDono)) {
                    JOptionPane.showMessageDialog(null, "Caixa removida.");
                } else {
                    JOptionPane.showMessageDialog(null, "Caixa não encontrada.");
                }
                break;

            case 3:
                String buscarDono = JOptionPane.showInputDialog("Digite o nome do dono para
procurar:");
                int indice = deposito.procurarCaixa(buscarDono);
                if (indice != -1) {
                    JOptionPane.showMessageDialog(null, "Caixa encontrada: " +
deposito.procurarCaixa(buscarDono));
                } else {
                    JOptionPane.showMessageDialog(null, "Caixa não encontrada.");
                }
                break;

            case 4:
                String alterarDono = JOptionPane.showInputDialog("Digite o nome do dono para alterar:");
                String novoCorredor = JOptionPane.showInputDialog("Novo corredor:");
                int novaPosicao = Integer.parseInt(JOptionPane.showInputDialog("Nova posição:"));
                if (deposito.alterarCaixa(alterarDono, novoCorredor, novaPosicao)) {
                    JOptionPane.showMessageDialog(null, "Caixa alterada.");
                } else {
                    JOptionPane.showMessageDialog(null, "Caixa não encontrada.");
                }
                break;

            case 5:
                double pesoMinimo = 10.0;
                var caixasPesadas = deposito.listarCaixasPesadas(pesoMinimo);
                if (caixasPesadas.isEmpty()) {
                    JOptionPane.showMessageDialog(null, "Nenhuma caixa pesada encontrada.");
                } else {
                    JOptionPane.showMessageDialog(null, "Caixas pesadas:\n" + caixasPesadas);
                }
                break;

            case 6:
                JOptionPane.showMessageDialog(null, "Saindo...");
                break;

            default:
                JOptionPane.showMessageDialog(null, "Opção inválida!");
        }
    } while (opcao != 6);
  }
}
```

Problemas Propostos: Exercício 3

a) Crie a classe Cliente com os atributos privados do tipo String nome e fone e com o atributo inteiro id. Crie um construtor que receba valores para os atributos como parâmetros e os métodos de acesso e modificadores.

```
public class Cliente {
    private String nome;
    private String fone;
    private int id;

    public Cliente(int id, String nome, String fone) {
        this.id = id;
        this.nome = nome;
        this.fone = fone;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getFone() {
        return fone;
    }

    public void setFone(String fone) {
        this.fone = fone;
    }

    @Override
    public String toString() {
        return "Cliente{" +
                "ID=" + id +
                ", Nome='" + nome + '\'' +
                ", Telefone='" + fone + '\'' +
                '}';
    }
}
```

b) Crie a classe BancoDeClientes com um atributo privado do tipo ArrayList<Cliente> chamado clientes. Crie métodos para inserir um cliente, remover um cliente, alterar um cliente, listar os dados de um cliente e listar os dados de todos os clientes.

```
import java.util.ArrayList;

public class BancoDeClientes {
    private ArrayList<Cliente> clientes;

    public BancoDeClientes() {
        clientes = new ArrayList<>();
    }

    public void inserirCliente(Cliente cliente) {
        clientes.add(cliente);
    }

    public boolean removerCliente(int id) {
        return clientes.removeIf(cliente -> cliente.getId() == id);
    }

    public boolean alterarCliente(int id, String novoNome, String novoFone) {
        for (Cliente cliente : clientes) {
            if (cliente.getId() == id) {
                cliente.setNome(novoNome);
                cliente.setFone(novoFone);
                return true;
            }
        }
        return false;
    }

    public Cliente buscarCliente(int id) {
        for (Cliente cliente : clientes) {
            if (cliente.getId() == id) {
                return cliente;
            }
        }
        return null;
    }

    public void listarTodosClientes() {
        if (clientes.isEmpty()) {
            System.out.println("Nenhum cliente cadastrado.");
        } else {
            System.out.println("Lista de Clientes:");
            for (Cliente cliente : clientes) {
                System.out.println(cliente);
            }
        }
    }
}
```

c) Crie a classe CadastroApp, com o método main, e que tenha um menu que insira um cliente, remova um cliente, altere um cliente, liste os dados de um cliente e liste os dados de todos os clientes.

```
import javax.swing.JOptionPane;

public class CadastroApp {
    public static void main(String[] args) {
        BancoDeClientes banco = new BancoDeClientes();
        int opcao;

        do {
            opcao = Integer.parseInt(JOptionPane.showInputDialog(
                "1. Inserir cliente\n" +
                "2. Remover cliente\n" +
                "3. Alterar cliente\n" +
                "4. Listar dados de um cliente\n" +
                "5. Listar todos os clientes\n" +
                "6. Sair"));

            switch (opcao) {
                case 1:
                    int id = Integer.parseInt(JOptionPane.showInputDialog("ID do cliente:"));
                    String nome = JOptionPane.showInputDialog("Nome do cliente:");
                    String fone = JOptionPane.showInputDialog("Telefone do cliente:");
                    banco.inserirCliente(new Cliente(id, nome, fone));
                    JOptionPane.showMessageDialog(null, "Cliente adicionado com sucesso!");
                    break;

                case 2:
                    int idRemover = Integer.parseInt(JOptionPane.showInputDialog("Digite o ID do cliente para
remover:"));
                    if (banco.removerCliente(idRemover)) {
                        JOptionPane.showMessageDialog(null, "Cliente removido com sucesso!");
                    } else {
                        JOptionPane.showMessageDialog(null, "Cliente não encontrado.");
                    }
                    break;

                case 3:
                    int idAlterar = Integer.parseInt(JOptionPane.showInputDialog("Digite o ID do cliente para
alterar:"));
                    String novoNome = JOptionPane.showInputDialog("Novo nome do cliente:");
                    String novoFone = JOptionPane.showInputDialog("Novo telefone do cliente:");
                    if (banco.alterarCliente(idAlterar, novoNome, novoFone)) {
                        JOptionPane.showMessageDialog(null, "Cliente alterado com sucesso!");
                    } else {
                        JOptionPane.showMessageDialog(null, "Cliente não encontrado.");
```

```
                }
                break;

            case 4:
                int idBuscar = Integer.parseInt(JOptionPane.showInputDialog("Digite o ID do cliente para
buscar:"));
                Cliente cliente = banco.buscarCliente(idBuscar);
                if (cliente != null) {
                    JOptionPane.showMessageDialog(null, cliente.toString());
                } else {
                    JOptionPane.showMessageDialog(null, "Cliente não encontrado.");
                }
                break;

            case 5:
                StringBuilder clientesList = new StringBuilder();
                for (Cliente c : banco.clientes) {
                    clientesList.append(c).append("\n");
                }
                JOptionPane.showMessageDialog(null, clientesList.length() > 0 ? clientesList : "Nenhum
cliente cadastrado.");
                break;

            case 6:
                JOptionPane.showMessageDialog(null, "Saindo...");
                break;

            default:
                JOptionPane.showMessageDialog(null, "Opção inválida!");
        }
    } while (opcao != 6);
  }
}
```