



Missão Prática - Nível 1 Mundo 3

Campus : POLO COPACABANA

Curso : DESENVOLVIMENTO FULL STACK

Disciplina : RPG0014 INICIANDO O CAMINHO PELO JAVA

Turma : 9003

Semestre: 2023.3 FLEX

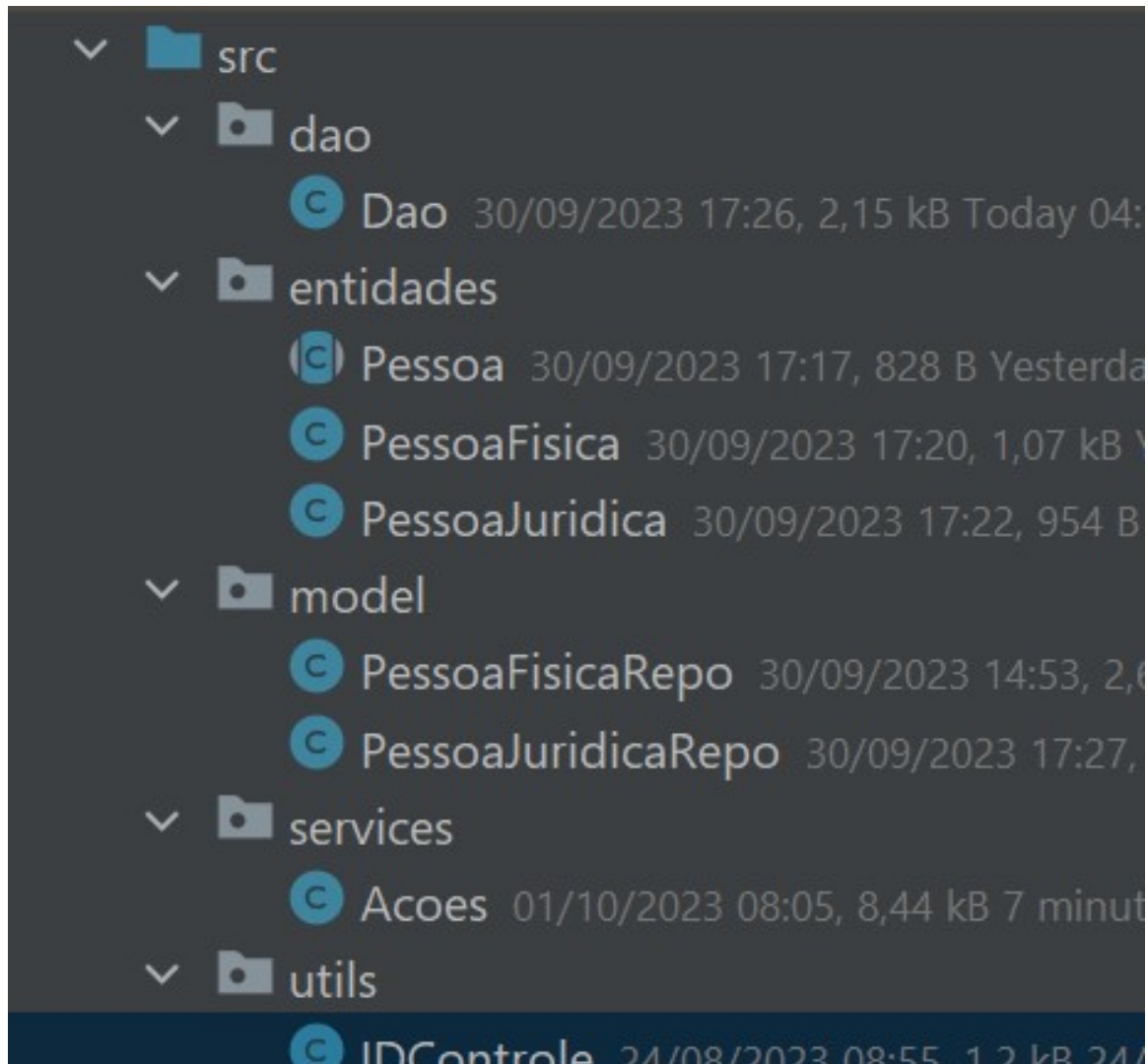
Nome : MARCO SERGIO ALBINO VITTORIO BAROZZI

2º Procedimento | Criação do Cadastro em Modo Texto

Objetivos:

1. Utilizar herança e polimorfismo na definição de entidades.
2. Utilizar persistência de objetos em arquivos binários
3. Implementar uma interface cadastral em modo texto.
4. Utilizar o controle de exceções da plataforma Java.
5. No final do projeto, o aluno terá implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

Códigos desenvolvidos :



Classe MainPart2.java (Apresentação do Menu)

```
import services.Acoes;
import java.util.Scanner;

public class MainPart2 {

    static Scanner scanner = new Scanner(System.in);
    static String opcaoPessoa = "";
    static String opcaoAcao = "";
```

```

public static void main(String[] args) {
    while (true) {
        menuAcoes();
        menuPessoa();
        executarAcao();
    }

}

private static void executarAcao() {
    Acoes acoes = new Acoes(opcaoPessoa, opcaoAcao);
    acoes.executandoAcoes();
}

private static void menuPessoa() {
    System.out.println("F - Pessoa Fisica | J - Pessoa
Juridica ");

    while (true) {
        String pessoa = scanner.next();
        if (pessoa.contains("f") || pessoa.contains("F") ||
pessoa.contains("j") || pessoa.contains("J")) {
            if (pessoa.contains("f") ||
pessoa.contains("F")) {
                //dados = new PessoaFisica();
                opcaoPessoa = "f";
            } else {
                //dados = new PessoaJuridica();
                opcaoPessoa = "j";
            }
            break;
        } else {
            System.out.println("Pessoa deve ser (F)isica ou
(J)uridica");
        }
    }

}

private static void menuAcoes() {
    while (true) {
        System.out.println("");
        System.out.println("");
    }

    System.out.println("=====");
}

```

```

        System.out.println("1 - Incluir Pessoa");
        System.out.println("2 - Alterar Pessoa");
        System.out.println("3 - Excluir Pessoa");
        System.out.println("4 - Buscar pelo Id");
        System.out.println("5 - Exibir todos");
        System.out.println("6 - Persistir Dados");
        System.out.println("7 - Recuperar Dados");
        System.out.println("0 - Finalizar Programa");

System.out.println("=====");
;

        System.out.print("opção : ");
        int escolha = -1;
        try {
            escolha = scanner.nextInt();

        } catch (RuntimeException e) {
            escolha = -1;
            scanner.nextLine(); // Limpa o buffer de
entrada
        }

        switch (escolha) {
            case 1:
                opcaoAcao = "I";
                //Incluir;
                return;
            case 2:
                opcaoAcao = "A";
                //Alterar
                return;
            case 3:
                opcaoAcao = "R";
                //excluir (Remove);
                return;
            case 4:
                opcaoAcao = "B";
                //buscarPeloId
                return;
            case 5:
                opcaoAcao = "S";
                //exibirTodos (Show all)
                return;
            case 6:
                opcaoAcao = "P";
                //persistirDados
                return;

```



```

        this.id= IDControle.getID();
        this.individualId=id;

        this.nome = nome;
    }

    public Integer getId() {
        return individualId;
    }

    public String getNome() {
        return nome;
    }

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

    public abstract String exhibir();
}

```

Classe PessoaFisica.java

```

package entidades;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable
{
    private static final long serialVersionUID = 1L;

    private Integer idade;
    private String cpf;
    public PessoaFisica() {
    }
    public PessoaFisica(Integer idade, String cpf) {
        this.idade = idade;
        this.cpf = cpf;
    }

    public PessoaFisica( String nome, Integer idade, String cpf)
    {
        super( nome);
    }
}

```

```

        this.idade = idade;
        this.cpf = cpf;

    }
    public Integer getIdade() {
        return idade;
    }

    public void setIdade(Integer idade) {
        this.idade = idade;
    }

    public String getCpf() {
        return cpf;
    }
    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    @Override
    public String exhibir() {
        return
            "id : " + getId() + "\n" +
            "nome : " + getNome() + "\n" +
            "idade : " + idade + "\n" +
            "cpf : " + cpf + "\n" ;
    }
}

```

Classe PessoaJuridica.java

```

package entidades;

import java.io.Serializable;

import utils.IDControle;

public class PessoaJuridica extends Pessoa implements
Serializable {
    private static final long serialVersionUID = 1L;

    private String cnpj;
}

```

```

public PessoaJuridica() {

}

public PessoaJuridica(String cnpj) {
    this.cnpj = cnpj;
}

public PessoaJuridica(String nome, String cnpj) {
    super(nome);
    this.cnpj = cnpj;
}

public PessoaJuridica(Integer id, String nome, String cnpj)
{
    super(nome);
    this.cnpj = cnpj;
}

public String getCnpj() {
    return cnpj;
}

public void setCnpj(String cnpj) {
    this.cnpj = cnpj;
}

@Override
public String exibir() {
    return
        "id : " + getId() + "\n" +
        "empresa : " + getNome() + "\n" +
        "cnpj : " + getCnpj() + "\n";
}

}

```

Classe PessoaFisicaRepo.java

```

package model;

import entidades.PessoaFisica;

import java.io.*;
import java.util.*;

```



```

public class PessoaFisicaRepo {

    private List<PessoaFisica> listaPessoasFisicas;

    // Construtor
    public PessoaFisicaRepo() {

        this.listaPessoasFisicas = new ArrayList<>();
    }

    // Método para inserir uma pessoa física
    public void inserir(PessoaFisica pessoaFisica) {

        listaPessoasFisicas.add(pessoaFisica);
    }

    // Método para alterar uma pessoa física

    public boolean alterar(PessoaFisica pessoaFisica) {
        for (int i = 0; i < listaPessoasFisicas.size(); i++) {
            if (listaPessoasFisicas.get(i).getId() ==
pessoaFisica.getId()) {
                listaPessoasFisicas.set(i, pessoaFisica);
                return true; // Retorna true indicando sucesso
na alteração
            }
        }

        return false; // Retorna false se a pessoa física não
foi encontrada na lista
    }

    // Método para excluir uma pessoa física por ID

    public boolean excluir(int id) {
        for (PessoaFisica p:listaPessoasFisicas) {
            if(p.getId()==id){
                listaPessoasFisicas.remove(p);
                return true;
            }
        }
        return false; // Retorna false se a pessoa física não
foi encontrada na lista
    }
}

```

```

    public PessoaFisica obter(int id) {
        return listaPessoasFisicas.stream()
            .filter(pessoaFisica -> pessoaFisica.getId() ==
id)
            .findFirst()
            .orElse(null);
    }

    // Método para obter todas as pessoas físicas

    public List<PessoaFisica> obterTodos() {
        return listaPessoasFisicas ;
    }

    public void persistir(String nomeArquivo) {

        if(listaPessoasFisicas.isEmpty()){
            System.out.println("Não existem registros a serem
persistidos");
            return;
        }

        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            out.writeObject(listaPessoasFisicas);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void recuperar(String nomeArquivo) {
        try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            listaPessoasFisicas = (ArrayList<PessoaFisica>)
in.readObject();

        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    public void ListarTodas(){
        for (PessoaFisica pf: listaPessoasFisicas) {
            System.out.println(pf.exibir());
        }
    }

```

```

    }

    }

}

```

Classe PessoaJuridicaRepo.java

```

package model;

import entidades.PessoaJuridica;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaRepo {

    private List<PessoaJuridica> listaPessoasJuridicas;

    // Construtor
    public PessoaJuridicaRepo() {

        this.listaPessoasJuridicas = new
ArrayList<PessoaJuridica>();
    }

    // Método para inserir uma pessoa física
    public void inserir(PessoaJuridica pessoaJuridica) {
        listaPessoasJuridicas.add(pessoaJuridica);
    }

    // Método para alterar uma pessoa física

    public boolean alterar(PessoaJuridica pessoaJuridica) {
        for (int i = 0; i < listaPessoasJuridicas.size(); i++) {
            if (listaPessoasJuridicas.get(i).getId() ==
pessoaJuridica.getId()) {
                listaPessoasJuridicas.set(i, pessoaJuridica);
                return true; // Retorna true indicando sucesso
na alteração
            }
        }
    }
}

```

```

        return false; // Retorna false se a pessoa não foi
encontrada na lista
    }

    // Método para excluir uma pessoa física por ID
    public boolean excluir(int id) {
        for (PessoaJuridica p: listaPessoasJuridicas) {
            if(p.getId()==id){
                listaPessoasJuridicas.remove(p);
                return true;
            }
        }
        return false; // Retorna false se a pessoa física não
foi encontrada na lista
    }

    public PessoaJuridica obter(int id) {
        return listaPessoasJuridicas.stream()
            .filter(pessoaJuridica -> pessoaJuridica.getId()
== id)
            .findFirst()
            .orElse(null);
    }

    // Método para obter todas as pessoas físicas
    public ArrayList<PessoaJuridica> obterTodos_1() {
        return new ArrayList<>(listaPessoasJuridicas);
    }

    public List<PessoaJuridica> obterTodos() {
        return listaPessoasJuridicas;
    }

    public void persistir(String nomeArquivo) {
        if(listaPessoasJuridicas.isEmpty()){
            System.out.println("Não existem registros a serem
persistidos");
            return;
        }
        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            out.writeObject(listaPessoasJuridicas);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

        public void recuperar(String nomeArquivo) {
            try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
                listaPessoasJuridicas = (ArrayList<PessoaJuridica>)
in.readObject();

                } catch (IOException | ClassNotFoundException e) {
                    e.printStackTrace();
                }
            }

            public void ListarTodas() {
                for (PessoaJuridica pf: listaPessoasJuridicas) {
                    System.out.println(pf.exibir());
                }
            }
        }
    }
}

```

Classe Acoes.java (tratamento da opção selecionada)

```

package services;

import java.util.Scanner;

import dao.Dao;
import entidades.Pessoa;
import entidades.PessoaFisica;
import entidades.PessoaJuridica;

import utils.Util;

public class Acoes {
    private PessoaFisica pessoaFisica;
    private PessoaJuridica pessoaJuridica;
    private Scanner scanner = new Scanner(System.in);
    private String opcaoPessoa;
    private String opcaoAcao;
    private String prefixo="";
    static Integer idBusca = 0;
    private Util utl;
    private Dao dao;
    private String nome = "";
    private String cpf = "";
    private String cnpj = "";
    private Integer idade = 0;
    public Acoes(String opcaoPessoa, String opcaoAcao) {

```

```

        this.opcaoPessoa = opcaoPessoa;
        this.opcaoAcao = opcaoAcao;
        this.utl = new Util(opcaoPessoa, opcaoAcao);
        this.dao = new Dao(opcaoPessoa);
    }

    public void executandoAcoes() {
        String tpPessoa = (opcaoPessoa == "f") ? "Pessoa fisica"
: "Pessoa juridica";
        switch (opcaoAcao) {
            case "I":
                novo();
                break;
            case "A":
                System.out.println("");
                editar();
                break;
            case "R":
                excluir();
                break;
            case "B":
                System.out.println("buscar dados by id " +
tpPessoa);
                mostrarItemPorId();
                break;
            case "S":
                System.out.println("exibir todos dados " +
tpPessoa);
                mostrarTodos();
                break;
            case "P":
                System.out.println("persisitir dados " +
tpPessoa);
                System.out.println("");
                persistirDadosBinarios();
                break;
            case "G":
                //recuperarDados(Get)
                System.out.println("recuperar dados " +
tpPessoa);
                System.out.println("");
                recuperarDadosBinarios();
                break;
            case "E":
                //sair(exit)
                System.out.println("sair !");

```

```

        scanner.close();
        System.exit(0);
        break;
    default:
        System.out.println("Escolha inválida. Tente
novamente.");
        utl.clickMe();
    }
}

private void mostrarItemPorId() {
    obterId();
    obterDadosById();
}

private void editar() {
    obterId();
    alterarDados();
}

private void novo() {
    preencheDadosPessoa();
    insercaoListasPessoa();
}

private void excluir() {
    obterId();
    obterDadosById();

    boolean inloop=true;

    while(inloop){
        System.out.println("Confirma a exclusão do item
(S/N) ?");
        String resp = scanner.nextLine();

        if(resp.equals("N")||resp.equals("n")){
            return;
        }else if(resp.equals("s")||resp.equals("S")){
            inloop=false;
            dao.excluir(idBusca);
            System.out.println("excluindo dados " +
((opcaoPessoa=="f") ? "Pessoa Fisica": "Pessoa Juridica"));
        }
    }

}

private void mostrarTodos() {
    dao.obterTodos();
    utl.clickMe();
}

```

```

    }
    private void recuperarDadosBinarios() {
        prefixo = utl.inputDadosText("Digite o prefixo do
arquivo",
                                "o prefixo precisa ser
preenchido",
                                "");
        dao.recuperar(prefixo);
        utl.clickMe();
    }
    private void persistirDadosBinarios() {
        prefixo = utl.inputDadosText("Digite o prefixo do
arquivo",
                                "o prefixo precisa ser
preenchido", "");
        dao.persistir(prefixo);
    }
    private void insercaoListasPessoa() {
        if (opcaoPessoa.equals("f")) {
            dao.inserirDados(pessoaFisica);
        } else {
            dao.inserirDados(pessoaJuridica);
        }
    }
    private void edicaoListasPessoa() {
        if (opcaoPessoa.equals("f")) {
            dao.alterar(pessoaFisica);
        } else {
            dao.alterar(pessoaJuridica);
        }
    }
    private void preencheDadosPessoa() {
        String nomeNovo = "";
        String cpfNovo = "";
        String cnpjNovo = "";
        Integer idadeNovo = 0;

        // nome
        if (opcaoAcao.equals("A")) {
            scanner.nextLine();
        }

        nomeNovo = utl.inputDadosText("Digite o nome",
                                "nome precisa ser preenchido", nome);

        if (opcaoPessoa.equals("f")) {

```



```

        // idade
        idadeNovo= utl.inputDadosNum("Digite o idade",
                                     "insira a idade entre 18 e 99
anos"
                                     ,idade);

        // cpf
        cpfNovo = utl.inputDadosText("Digite o cpf",
                                     "o cpf precisa ser
definido",
                                     cpf );

        utl.clickMe();

concluiEntradaDeDadosPessoa (cpfNovo,nomeNovo,idadeNovo);

    } else {
        // cnpj
        cnpjNovo= utl.inputDadosText("Digite o cnpj",
                                     "o cnpj precisa ser
definido",
                                     cnpj);

        utl.clickMe();

        concluiEntradaDeDadosPessoa (cnpjNovo,nomeNovo);

    }

}

private void concluiEntradaDeDadosPessoa(String cnpj, String
nome) {
    if (opcaoAcao.equals("A")) {
        pessoaJuridica.setCnpj (cnpj);
        pessoaJuridica.setNome (nome);
    }else {
        atualizarInstancia(new PessoaJuridica(nome, cnpj));
    }
}

private void concluiEntradaDeDadosPessoa(String cpf, String
nome, Integer idade) {

    if (opcaoAcao.equals("A")) {
        pessoaFisica.setCpf (cpf);
        pessoaFisica.setNome (nome);
        pessoaFisica.setIdade (idade);
    }
}

```

```

        }else{
            atualizarInstancia(new PessoaFisica(nome, idade,
cpf));
        }
    }
    private void atualizarInstancia(Pessoa cls) {
        if (opcaoPessoa == "f") {
            pessoaFisica = (PessoaFisica) cls;

        } else {
            pessoaJuridica = (PessoaJuridica) cls;

        }
    }
    private void obterId() {
        //somente valido para alteracao/exclusão/busca
        while (true) {
            try {
                System.out.println("Digite o id");
                idBusca = scanner.nextInt();
                scanner.nextLine();
                return;
            } catch (RuntimeException e) {
                System.out.println("Id inválido, valor deve ser
numerico. Tente novamente");
                utl.clickMe();
            }
        }
    }
    private void alterarDados() {

        if (obtemDadosById()) {

System.out.println("=====
=====");
            System.out.println("Caso deseja manter o valor
original, tecla [enter] para seguir o proximo item");

System.out.println("=====
=====");
            utl.clickMe();
            preencheDadosPessoa();
            edicaoListasPessoa();
        }

    }
    private boolean obtemDadosById() {

```

```

boolean pessoaValida;
Integer id=idBusca;

if (opcaoPessoa == "f") {
    pessoaFisica = (PessoaFisica) dao.obterPessoa(id);
    pessoaValida = utl.verificarInstancia(pessoaFisica);
} else {
    pessoaJuridica = (PessoaJuridica)
dao.obterPessoa(id);
    pessoaValida =
utl.verificarInstancia(pessoaJuridica);
}

if (pessoaValida) {
    if(opcaoPessoa == "f"){
        nome =pessoaFisica.getNome();
        cpf =pessoaFisica.getCpf();
        idade =pessoaFisica.getIdade();
    }else{
        nome =pessoaJuridica.getNome();
        cnpj =pessoaJuridica.getCnpj();
    }
    return true;
} else {
    nome="";
    idade=0;
    cpf="";
    cnpj="";
    return false;
}

}

}

```

Classe Dao.java (Operações de persistencia)

```
package dao;

import entidades.Pessoa;
import entidades.PessoaFisica;
import entidades.PessoaJuridica;
import model.PessoaFisicaRepo;
import model.PessoaJuridicaRepo;

import java.util.List;

public class Dao {
    private static PessoaFisicaRepo repo1 = new
PessoaFisicaRepo();
    private static PessoaJuridicaRepo repo2 = new
PessoaJuridicaRepo();
    private String opcaoPessoa;

    public Dao(String opcaoPessoa) {
        this.opcaoPessoa = opcaoPessoa;
    }
    public void inserirDados(Pessoa pessoa) {
        if (opcaoPessoa == "f") {
            repo1.inserir((PessoaFisica) pessoa);

        } else {
            repo2.inserir((PessoaJuridica) pessoa);
        }
    }

    public void excluir(Integer id) {
        repo1.excluir(id);
    }

    public Pessoa obterPessoa(Integer id) {
        if (opcaoPessoa == "f") {
            return repo1.obter(id);
        } else {
            return repo2.obter(id);
        }
    }

    public void obterTodos() {
        if (opcaoPessoa == "f") {
            List<PessoaFisica> lPessoaFisica =
repo1.obterTodos();
```

```

        imprimeTodos(lPessoaFisica);
    } else {
        List<PessoaJuridica> lPessoaJuridica =
repo2.obterTodos();
        imprimeTodos(lPessoaJuridica);
    }

}

public void alterar(Pessoa pessoa) {
    if (opcaoPessoa == "f") {
        repo1.alterar((PessoaFisica) pessoa);
    } else {
        repo2.alterar((PessoaJuridica) pessoa);
    }
}

private void imprimeTodos(List<? extends Pessoa> lista) {
    for (Pessoa p : lista) {
        System.out.println(p.exibir());
    }
}

public void recuperar(String prefixo) {
    if (opcaoPessoa.equals("f")) {
        repo1.recuperar(prefixo + ".fisica.bin");
    } else {
        repo2.recuperar(prefixo + ".juridica.bin");
    }
}

public void persistir(String prefixo) {
    if (opcaoPessoa.equals("f")) {
        repo1.persistir(prefixo + ".fisica.bin");
    } else {
        repo2.persistir(prefixo + ".juridica.bin");
    }
}
}

```

Classe IdControle.java (controle do ID)

```

package utils;

import java.io.*;

public class IDControle {

```

```

private static final String FILE_ID = "id.bin";

static {

    // Inicializar o arquivo com o ID 0, se ele ainda não
existir.
    File file = new File(FILE_ID);
    if (!file.exists()) {
        try {
            saveID(0);
        } catch (IOException e) {
            System.err.println("Erro ao inicializar o
arquivo de ID: " + e.getMessage());
        }
    }
}

public static int getID() {
    int proximoID=1;

    try{
        int atualID = readID();
        proximoID = atualID + 1;
        saveID(proximoID);
    }catch (IOException e){

    }finally {
        return proximoID;
    }

}

private static int readID() throws IOException {
    try (DataInputStream in = new DataInputStream(new
FileInputStream(FILE_ID))) {
        return in.readInt();
    }
}

private static void saveID(int id) throws IOException {
    try (DataOutputStream out = new DataOutputStream(new
FileOutputStream(FILE_ID))) {
        out.writeInt(id);
    }
}

```

```
}
```

Classe Util.java (tratamento dos inputs)

```
package utils;

import entidades.Pessoa;

import java.io.IOException;
import java.util.Scanner;

public class Util {
    private String opcaoAcao;
    private String opcaoPessoa;
    private Scanner scanner = new Scanner(System.in);
    public Util(String opcaoPessoa, String opcaoAcao) {

        this.opcaoPessoa=opcaoPessoa;
        this.opcaoAcao=opcaoAcao;

    }

    // validação idade
    public boolean campoIdadeValida(String valor, String msg) {
        try {
            Integer idade = Integer.parseInt(valor);
            if ((idade < 18) || (idade > 99)) {
                if (!opcaoAcao.equals("A")) {
                    System.out.println(msg);
                    clickMe();
                }
                return false;
            }
        } catch (NumberFormatException e) {
            System.out.println("Por favor, insira um número inteiro válido para a idade.");
            return false;
        }

        return true;
    }

    // validação nome
    public boolean campoValido(String valor, String msg) {

        if (valor.isEmpty()) {
```

```

        if (!opcaoAcao.equals("A")) {
            System.out.println(msg);
            clickMe();
        }
        return false;
    }
    return true;
}

public boolean verificarInstancia(Pessoa pessoa) {

    String msg;
    if (opcaoPessoa.equals("f")) {
        msg = "Pessoa Fisica";
    } else {
        msg = "Pessoa Juridica";
    }
    if (pessoa == null) {
        System.out.println("Id da " + msg + " não foi encontrado, tente novamente");
        return false;
    } else {
        System.out.println("=====");
        System.out.println(pessoa.exibir());
        clickMe();
        return true;
    }

}

// mensagem de espera para apresentação de dados
public void clickMe() {

    System.out.println("");
    System.out.println("tecle qualquer tecla para continuar.. ");
    try {
        System.in.read();
    } catch (Exception e) {
        e.printStackTrace();
    }

}

```



```

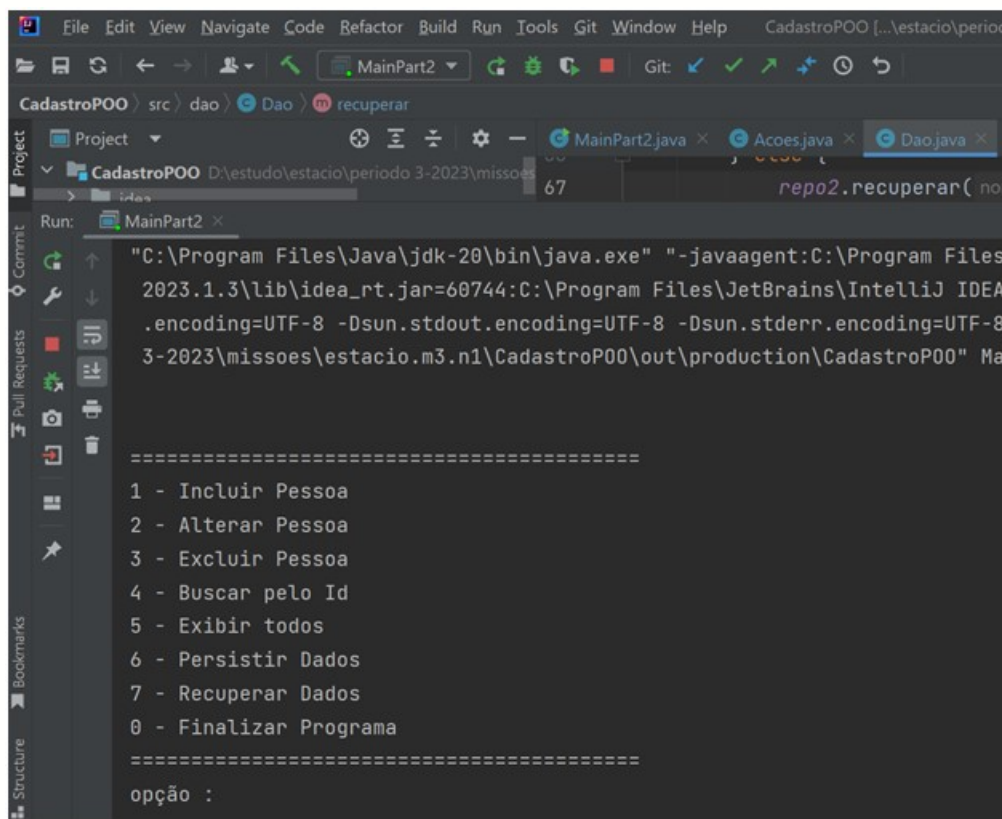
    public String inputDadosText(String msgTitulo,String
msgErr,String valorOriginal) {
        boolean inloop = true;
        String sReturn="";
        while (inloop) {
            System.out.println(msgTitulo);
            sReturn = scanner.nextLine();
            if (!campoValido(sReturn, msgTitulo)) {
                if (opcaoAcao.equals("A")) {
                    sReturn =valorOriginal;
                    inloop = false;
                }
            } else {
                inloop = false;
            }
        }
        return sReturn;
    }

    public Integer inputDadosNum(String msgTitulo,String
msgErr,Integer valorOriginal) {
        boolean inloop = true;
        Integer sReturn=0;
        while (inloop) {
            System.out.println(msgTitulo);
            String entrada = scanner.nextLine();
            if (!campoValido(entrada, msgTitulo)) {
                if (opcaoAcao.equals("A")) {
                    sReturn =valorOriginal;
                    inloop = false;
                }
            } else {
                try {
                    sReturn =Integer.parseInt(entrada);
                    inloop = false;
                }catch (Exception e){
                    System.out.println("valor invalido, tente
novamente");
                }
            }
        }
        return sReturn;
    }
}

```

Resultados da execução dos códigos

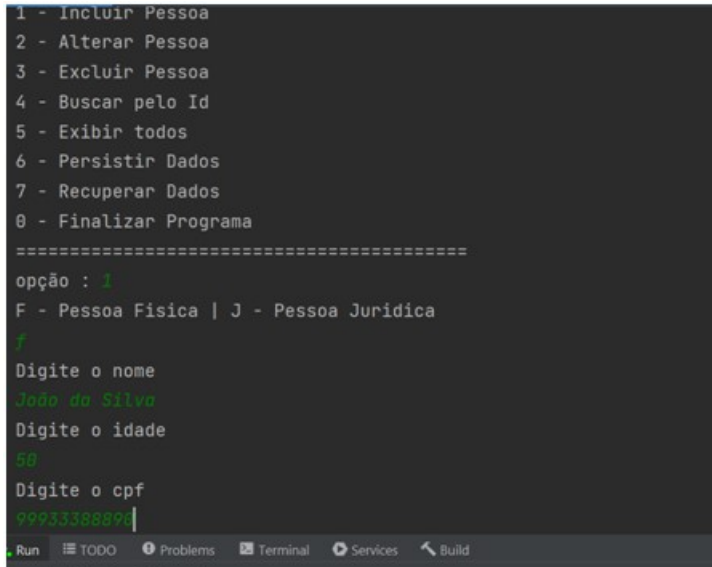
1 - Apresentar as opções do programa para o usuário, sendo 1 para incluir, 2 para alterar, 3 para excluir, 4 para exibir pelo id, 5 para exibir todos, 6 para salvar dados, 7 para recuperar dados e 0 para finalizar a execução.



```
Run: MainPart2 x
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\
2023.1.3\lib\idea_rt.jar=60744:C:\Program Files\JetBrains\IntelliJ IDEA
.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8
3-2023\missoes\estacio.m3.n1\CadastroPOO\out\production\CadastroPOO" Ma

=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
opção :
```

2 - Selecionada a opção incluir, escolher o tipo (Física ou Jurídica), receber os dados a partir do teclado e adicionar no repositório correto.



```
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir todos
6 - Persistir Dados
7 - Recuperar Dados
8 - Finalizar Programa
=====
opção : 1
F - Pessoa Física | J - Pessoa Juridica
f
Digite o nome
João da Silva
Digite o idade
50
Digite o cpf
99933388899
```

3 -Selecionar a opção excluir, escolher o tipo Física, receber o id a partir do teclado e remover do repositório correto.

obs: caso se não se conheça o id pode estar se usando a opção 5 (exibir todos) ou se digitar diretamente pois antes da remoção será apresentado os dados referentes ao id assim como uma confirmação

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
opção : 5
F - Pessoa Fisica | J - Pessoa Juridica
f
```

Agora com a listagem verifico o id

```
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
opção : 5
F - Pessoa Fisica | J - Pessoa Juridica
f
exibir todos dados Pessoa fisica
id : 1
nome : João da Silva
idade : 50
cpf : '99933388890'
```

Continuando

```
tecle qualquer tecla para continuar..
```

```
Confirma a exclusão do item (S/N) ?
```

```
S
```

```
excluindo dados Pessoa Fisica
```

```
=====
```

- 1 - Incluir Pessoa
- 2 - Alterar Pessoa
- 3 - Excluir Pessoa
- 4 - Buscar pelo Id
- 5 - Exibir todos
- 6 - Persistir Dados
- 7 - Recuperar Dados

Agora realizo a opção excluir para o id

```
=====
```

- 1 - Incluir Pessoa
- 2 - Alterar Pessoa
- 3 - Excluir Pessoa
- 4 - Buscar pelo Id
- 5 - Exibir todos
- 6 - Persistir Dados
- 7 - Recuperar Dados
- 0 - Finalizar Programa

```
=====
```

```
opção : 3
```

```
F - Pessoa Fisica | J - Pessoa Juridica
```

```
f
```

```
Digite o id
```

```
1
```

```
Run Debug TODO Problems Terminal Services Build  
are up-to-date (3 minutes ago)
```

Serão novamente apresentados os dados

```
7 - Recuperar Dados
0 - Finalizar Programa
=====
opção : 3
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o id
1
=====
id : 1
nome : João da Silva
idade : 50
cpf : '99933388890
teste qual quer de 1a para continuar
```

4- Incluir nova pessoa Juridica

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
opção : 1
F - Pessoa Fisica | J - Pessoa Juridica
j
Digite o nome
Material de construção do Zé
Digite o cnpj
666677778888
```

5 - Selecionada a opção alterar, escolher o tipo Jurídica, receber o id a partir do teclado, apresentar os dados atuais, solicitar os novos dados e alterar no repositório correto.

```

5 - Exibir todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
opção : 2
F - Pessoa Fisica | J - Pessoa Juridica
j
Digite o id
4
=====
id : 4
empresa : Material de construção do Zé
cnpj : '666677778888

tecle qualquer tecla para continuar..
|

```

```

cnpj : '666677778888

tecle qualquer tecla para continuar..

=====
Caso deseja manter o valor original, tecla [enter] para seguir o proximo item
=====

tecle qualquer tecla para continuar..

Digite o nome

Digite o cnpj
222200008888

tecle qualquer tecla para continuar..

```

Agora confirmando a alteração através da listagem todos

```

=====
opção : 5
F - Pessoa Fisica | J - Pessoa Juridica
j
exibir todos dados Pessoa juridica
id : 4
empresa : Material de construção do Zé
cnpj : '222200008888

tecle qualquer tecla para continuar..
|

```

6 salvar pessoa jurídica em arquivo binário

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
opção : 6
F - Pessoa Fisica | J - Pessoa Juridica
j
persistir dados Pessoa juridica

Digite o prefixo do arquivo
bin|
```

7 incluir duas pessoas físicas em persistir

```
opção : 1
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o nome
Maria Aparecida
Digite o idade
22
Digite o cpf
88811155509|
```



```
opção : 1
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o nome
Marlene Maria
Digite o idade
34
Digite o cpf
33322266680

tecle qualquer tecla para continuar..
```

8 Persistir pessoas físicas

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
opção : 6
F - Pessoa Fisica | J - Pessoa Juridica
f
persistir dados Pessoa fisica

Digite o prefixo do arquivo
bin2
```

9 Recuperar Pessoa Jurídica

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
opção : 7
F - Pessoa Fisica | J - Pessoa Juridica
j
recuperar dados Pessoa juridica

Digite o prefixo do arquivo
bin3
```

9 Visualizar Pessoa Jurídica

```
opção : 5
F - Pessoa Fisica | J - Pessoa Juridica
j
exibir todos dados Pessoa juridica
id : 4
empresa : Material de construção do Zé
cnpj : '222200008888

tecle qualquer tecla para continuar..
|
```

10 Recuperar Pessoa Fisica

```
opção : 7
F - Pessoa Fisica | J - Pessoa Jur
f
recuperar dados Pessoa fisica

Digite o prefixo do arquivo
```

11 obter pessoa física por id

```
=====
opção : 4
F - Pessoa Fisica | J - Pessoa Juridica
f
buscar dados by id Pessoa fisica
Digite o id
6
=====
id : 6
nome : Marlene Maria
idade : 34
cpf : '33322266680'

tecle qualquer tecla para continuar..
.
```

Análise e Conclusão:

Quais as vantagens e desvantagens do uso de herança?

r) Reutilização de código, polimorfismo no sentido de utilizar classes como base para outras, extensibilidade ou seja adicionar funcionalidades ou modificar alguma existente, encapsulamento, relacionamentos etc...

Por que a interface `Serializable` é necessária ao efetuar persistência em arquivos binários?

r) Devido a interface indicar que a classe em questão poderá ter uma serialização e desserialização, no caso de recuperação de dados em um arquivo binário.

Como o paradigma funcional é utilizado pela API stream no Java?

r) Através de operação de alto nível como map, filter e reduce para processar coleções de dados.
Tambem permitem "Expressões Lambda"

Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?

r) Acredito que sejam os ORMs (Object-Relational Mapping) como Hibernate ou JPA são mais associados a bancos de dados relacionais, eles também podem ser usados para persistir dados em arquivos, especialmente quando bancos de dados integrados são usados.

Eles permitem que os desenvolvedores interajam com bancos de dados usando objetos Java em vez de SQL direto.