Desenvolvimento Full Stack
Missão Prática 1 – Mundo 3

Aluno: Vinícius Silva de Lima – 2023 0707 6309

Campus – Polo Vilar dos Teles – São João de Meriti – RJ
RPG0014 – Iniciando o caminho pelo Java – 2024.3

1. Título da prática: “1º Procedimento | Criação das entidades e sistema de persistência.”

2.Objetivo da prática:

Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.

- Utilizar herança e polimorfismo na definição de entidades.
- Utilizar persistência de objetos em arquivos binários. Implementar uma interface cadastral em modo texto.
- Utilizar o controle de exceções da plataforma Java.
- Implementar um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.
-

3. Códigos do projeto:

CadastroPoo:

```
package cadastropoo;

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

/**
 *
 * @author Vinícius
 */
public class CadastroPOO {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        try {
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            repo1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));
            repo1.inserir(new PessoaFisica(2, "Carlos", "22222222222", 52));
            repo1.persistir("pessoasFisicas.dat");
            System.out.println("Dados de Pessoa Fisica Armazenados.");

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar("pessoasFisicas.dat");
            System.out.println("Dados de Pessoa Fisica Recuperados.");
            for (PessoaFisica pf : repo2.obterTodos()) {
                pf.exibir();
        }
```

```
            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "33333333333333"));
            repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "44444444444444"));
            repo3.persistir("pessoasJuridicas.dat");
            System.out.println("Dados de Pessoa Juridica Armazenados.");

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
            repo4.recuperar("pessoasJuridicas.dat");
            System.out.println("Dados de Pessoa Juridica Recuperados.");
            for (PessoaJuridica pj : repo4.obterTodos()) {
                pj.exibir();
            }

        } catch (Exception e){
            e.printStackTrace();
        }

        // TODO code application logic here
    }

}
```

Pessoa:

```
package model;

import java.io.Serializable;

/**
 *
 * @author Vinícius
 */
public class Pessoa implements Serializable{
    private int id;
    private String nome;

    public Pessoa(){

    }

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

    public void exibir() {
        System.out.println("ID: " + id + ", Nome: " + nome);
    }

    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;
    }

}
```

PessoaFisica:

```
package model;

import java.io.Serializable;


/**
 *
 * @author Vinícius
 */
public class PessoaFisica extends Pessoa implements Serializable{
    private String cpf;
```

```
    private int idade;

    public PessoaFisica() {
    }

    public PessoaFisica(int id, String nome, String cpf, int idade) {
        super(id, nome);
        this.cpf = cpf;
        this.idade = idade;
    }

    @Override
    public void exibir(){
        super.exibir();
        System.out.println("CPF: " + cpf + ",Idade: " + idade);
    }

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

    public int getIdade() {
        return idade;
    }

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

PessoaJuridica:

```
package model;

import java.io.Serializable;


/**
 *
 * @author Vinícius
 */
public class PessoaJuridica extends Pessoa implements Serializable{
    private String cnpj;

    public PessoaJuridica(){
    }

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

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }

    public String getCnpj(){
        return cnpj;
    }

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

}
```

PessoaFisicaRepo:

```
package model;

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

/**
```

```
 *
 * @author Vinícius
 */

public class PessoaFisicaRepo{
    private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();

    public void inserir(PessoaFisica pessoaFisica) {
        pessoasFisicas.add(pessoaFisica);
    }

   public void alterar(PessoaFisica pessoaFisica) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(i).getId() == pessoaFisica.getId()) {
                pessoasFisicas.set(i, pessoaFisica);
                return;
            }

    }
}

public void excluir(int id) {
        pessoasFisicas.removeIf(p -> p.getId() == id);
    }

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

public List<PessoaFisica> obterTodos() {
        return new ArrayList<>(pessoasFisicas);
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            out.writeObject(pessoasFisicas);
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasFisicas = (ArrayList<PessoaFisica>) in.readObject();
        }
    }
}
```

PessoaJuridicaRepo:

```
package model;

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

/**
 *
 * @author Vinícius
 */
public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

    public void inserir(PessoaJuridica pessoaJuridica){
        pessoasJuridicas.add(pessoaJuridica);
    }

    public void alterar(PessoaJuridica pessoaJuridica) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            if (pessoasJuridicas.get(i).getId() == pessoaJuridica.getId()) {
                pessoasJuridicas.set(i, pessoaJuridica);
                return;
            }
    }
}

public void excluir(int id) {
        pessoasJuridicas.removeIf(p -> p.getId() == id);
```

```
    }

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

    public List<PessoaJuridica> obterTodos() {
        return new ArrayList<>(pessoasJuridicas);
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            out.writeObject(pessoasJuridicas);
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            @SuppressWarnings("unchecked")
            ArrayList<PessoaJuridica> temp = (ArrayList<PessoaJuridica>) in.readObject();
            this.pessoasJuridicas = temp;
        }
    }

}
```

CadastroPooParte2:

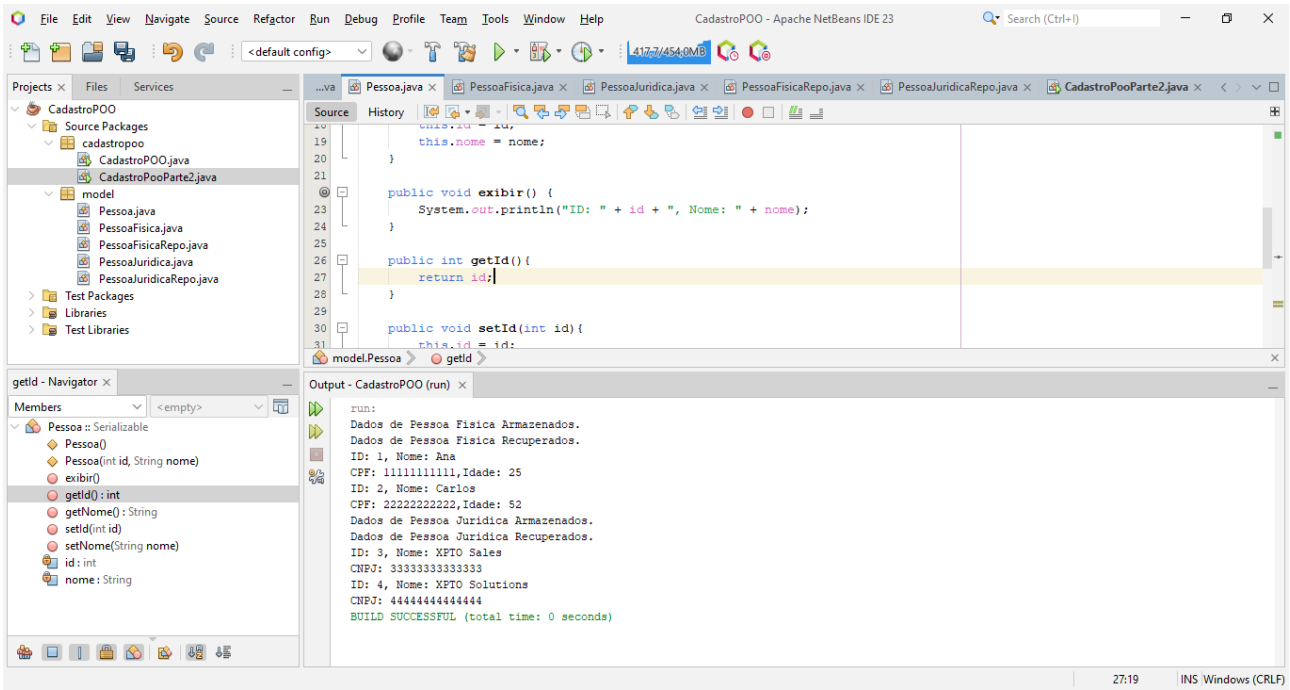
```
package cadastropoo;

import java.io.Serializable;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;
import java.io.IOException;
import java.util.Scanner;


/**
 *
 * @author Vinícius
 */
public class CadastroPooParte2 {

    private static PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
    private static PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        int opcao;

        do {
            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("Escolha uma opção: ");
            opcao = scanner.nextInt();
            scanner.nextLine();

            switch (opcao) {
                case 1:
                    incluirPessoa();
                    break;
                case 2:
                    alterarPessoa();
                    break;
                case 3:
                    excluirPessoa();
```

```
                break;
            case 4:
                buscarPeloId();
                break;
            case 5:
                exibirTodos();
                break;
            case 6:
                persistirDados();
                break;
            case 7:
                recuperarDados();
                break;
            case 0:
                System.out.println("Finalizando programa...");
                break;
            default:
                System.out.println("Opção inválida. Tente novamente.");
        }
    } while (opcao != 0);

    scanner.close();
}

private static void excluirPessoa() {
    System.out.println("Excluir Pessoa: [F] - Física | [J] - Jurídica");
    String tipo = scanner.nextLine().toUpperCase();
    System.out.print("Digite o id da pessoa: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    if (tipo.equals("F")) {
        repoFisica.excluir(id);
        System.out.println("Pessoa Física excluída.");
    } else if (tipo.equals("J")) {
        repoJuridica.excluir(id);
        System.out.println("Pessoa Jurídica excluída.");
    } else {
        System.out.println("Tipo inválido.");
    }
}

private static void buscarPeloId() {
    System.out.println("Buscar Pessoa pelo ID: [F] - Física | [J] - Jurídica");
    String tipo = scanner.nextLine().toUpperCase();
    System.out.print("Digite o id da pessoa: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    if (tipo.equals("F")) {
        PessoaFisica pf = repoFisica.obter(id);
        if (pf != null) {
            pf.exibir();
        } else {
            System.out.println("Pessoa Física não encontrada.");
        }
    } else if (tipo.equals("J")) {
        PessoaJuridica pj = repoJuridica.obter(id);
        if (pj != null) {
            pj.exibir();
        } else {
            System.out.println("Pessoa Jurídica não encontrada.");
        }
    } else {
        System.out.println("Tipo inválido.");
    }
}

private static void exibirTodos() {
    System.out.println("Exibir Todos: [F] - Física | [J] - Jurídica");
    String tipo = scanner.nextLine().toUpperCase();

    if (tipo.equals("F")) {
        System.out.println("Pessoas Físicas:");
        for (PessoaFisica pf : repoFisica.obterTodos()) {
            pf.exibir();
        }
    } else if (tipo.equals("J")) {
        System.out.println("Pessoas Jurídicas:");
```

```
            for (PessoaJuridica pj : repoJuridica.obterTodos()) {
                pj.exibir();
            }
        } else {
            System.out.println("Tipo inválido.");
        }
    }

    private static void persistirDados() {
        System.out.print("Digite o prefixo para os arquivos de persistência: ");
        String prefixo = scanner.nextLine();
        String arquivoFisica = prefixo + ".fisica.bin";
        String arquivoJuridica = prefixo + ".juridica.bin";

        try {
            repoFisica.persistir(arquivoFisica);
            repoJuridica.persistir(arquivoJuridica);
            System.out.println("Dados persistidos com sucesso.");
        } catch (IOException e) {
            System.err.println("Erro ao persistir os dados: " + e.getMessage());
        }
    }

    private static void recuperarDados() {
        System.out.print("Digite o prefixo para os arquivos de recuperação: ");
        String prefixo = scanner.nextLine();
        String arquivoFisica = prefixo + ".fisica.bin";
        String arquivoJuridica = prefixo + ".juridica.bin";

        try {
            repoFisica.recuperar(arquivoFisica);
            repoJuridica.recuperar(arquivoJuridica);
            System.out.println("Dados recuperados com sucesso.");
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Erro ao recuperar os dados: " + e.getMessage());
        }
    }

    private static void incluirPessoa() {
        System.out.println("Incluir Pessoa: [F] - Física | [J] - Jurídica");
        String tipo = scanner.nextLine().toUpperCase();

        while (!tipo.equals("F") && !tipo.equals("J")) {
            System.out.println("Tipo inválido. Por favor, digite F para Pessoa Física ou J para Pessoa Jurídica.");
            System.out.println("Incluir Pessoa: [F] - Física | [J] - Jurídica");
            tipo = scanner.nextLine().toUpperCase();

        }

        System.out.print("Digite o id da pessoa: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Digite o nome da pessoa: ");
        String nome = scanner.nextLine();

        if (tipo.equals("F")) {
            System.out.print("Digite o CPF da pessoa: ");
            String cpf = scanner.nextLine();

            System.out.print("Digite a idade da pessoa: ");
            int idade = scanner.nextInt();
            scanner.nextLine();

            PessoaFisica pf = new PessoaFisica(id, nome, cpf, idade);
            repoFisica.inserir(pf);
            System.out.println("Pessoa Física incluída com sucesso.");
        } else {
            System.out.print("Digite o CNPJ da pessoa: ");
            String cnpj = scanner.nextLine();

            PessoaJuridica pj = new PessoaJuridica(id, nome, cnpj);
            repoJuridica.inserir(pj);
            System.out.println("Pessoa Jurídica incluída com sucesso.");
        }
    }

    private static void alterarPessoa() {
        System.out.println("Alterar Pessoa: [F] - Física | [J] - Jurídica");
```

```
        String tipo = scanner.nextLine().toUpperCase();

        System.out.print("Digite o id da pessoa para alteração: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        if (tipo.equals("F")) {
            PessoaFisica pf = repoFisica.obter(id);
            if (pf != null) {
                System.out.print("Digite o novo nome da pessoa Física (ou deixe em branco para não alterar): ");
                String nome = scanner.nextLine();
                if (!nome.isEmpty())
                    pf.setNome(nome);

                System.out.print("Digite o novo CPF da pessoa Física (ou deixe em branco para não alterar): ");
                String cpf = scanner.nextLine();
                if (!cpf.isEmpty())
                    pf.setCpf(cpf);

                System.out.print("Digite a nova idade da pessoa Física (ou 0 para não alterar): ");
                int idade = scanner.nextInt();
                if (idade != 0)
                    pf.setIdade(idade);

                repoFisica.alterar(pf);
                System.out.println("Pessoa Física alterada com sucesso.");
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipo.equals("J")) {
            PessoaJuridica pj = repoJuridica.obter(id);
            if (pj != null) {
                System.out.print("Digite o novo nome da pessoa Jurídica (ou deixe em branco para não alterar): ");
                String nome = scanner.nextLine();
                if (!nome.isEmpty())
                    pj.setNome(nome);

                System.out.print("Digite o novo CNPJ da pessoa Jurídica (ou deixe em branco para não alterar): ");
                String cnpj = scanner.nextLine();
                if (!cnpj.isEmpty())
                    pj.setCnpj(cnpj);

                repoJuridica.alterar(pj);
                System.out.println("Pessoa Jurídica alterada com sucesso.");
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Tipo inválido.");
        }
    }

}
```

4. Resultado da execução:



5. Análise e conclusão:

a. Quais as vantagens e desvantagens do uso de herança?
As vantagens da herança em Java incluem a reutilização de código, facilitando a manutenção e a extensão de funcionalidades. Isso permite que você crie uma hierarquia de classes organizada. Por outro lado, as desvantagens incluem a complexidade que pode surgir com múltiplas camadas de herança e a dificuldade em entender a estrutura. E tem o fato que não existe o mecanismo de herança múltipla.

b. Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?
A interface Serializable é necessária porque permite que objetos sejam convertidos em um formato que pode ser facilmente salvo em arquivos binários. Sem ela, o Java não saberia como "transformar" os objetos em bytes para armazená-los e depois recuperá-los. É como ter um passaporte que permite que seu objeto viaje para o armazenamento e volte em seguida.

c. Como o paradigma funcional é utilizado pela API stream no Java?
O paradigma funcional na API Stream do Java permite que você processe coleções de dados de forma declarativa, usando funções em vez de loops tradicionais. Isso significa que você pode aplicar operações como map, filter e reduce para transformar e combinar dados de maneira mais clara e concisa, quase como uma conversa entre funções.

d. Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?
Em Java, o padrão mais comum para persistência de dados em arquivos é o **DAO** (Data Access Object). Esse padrão separa a lógica de acesso a dados da lógica de negócio. Assim, você pode gerenciar como os dados são salvos ou recuperados de forma organizada, facilitando manutenção e testes.

5.2. Análise e conclusão:

a. O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?
Elementos estáticos em Java são aqueles que pertencem à classe em si, não a instâncias específicas da classe. O método `main` é estático porque é o ponto de entrada da aplicação e precisa ser chamado pelo Java sem criar uma instância da classe.

b. Para que serve a classe Scanner?
A classe Scanner em Java serve para ler a entrada de dados, como texto ou números, de forma simples e direta.

c. Como o uso de classes de repositório impactou na organização do código?
O uso de classes de repositório ajuda a organizar o código separando a lógica de acesso a dados da lógica de negócio. Isso torna o código mais limpo e fácil de entender, facilita a manutenção e os testes, e permite que você troque a forma como os dados são armazenados sem afetar o resto da aplicação.