Relatório de Prática

Universidade Estácio de Sá

Campus Gilberto Gil

Curso: Desenvolvimento Full Stack

Disciplina: Iniciando o Caminho Pelo Java

Turma: 202402908936

Semestre: 2025.1

Aluno: Mario Henrique Trentini Ely

Data: 03/04/2025

Título da Prática

Cadastro de Clientes em Modo Texto com Java

Objetivo da Prática

Desenvolver um sistema de cadastro em modo texto utilizando a linguagem Java, com foco na aplicação dos conceitos de Programação Orientada a Objetos (POO), herança, polimorfismo, controle de exceções e persistência de dados em arquivos binários.

Código-Fonte

Segue abaixo os principais trechos do código desenvolvido para esta prática.

PessoaJuridicaRepo.java

```
package model;


import java.util.ArrayList;
```

```
import java.io.*;

public class PessoaJuridicaRepo {
    private ArrayList<PessoaJuridica> lista = new ArrayList<>();

    public void inserir(PessoaJuridica pj) {
        lista.add(pj);
    }

    public void alterar(PessoaJuridica pj) {
        for (int i = 0; i < lista.size(); i++) {
            if (lista.get(i).getId() == pj.getId()) {
                lista.set(i, pj);
                break;
            }
        }
    }

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

    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pj : lista) {
            if (pj.getId() == id) return pj;
        }
        return null;
```

```
    }

    public ArrayList<PessoaJuridica> obterTodos() {
        return lista;
    }

    public void persistir(String nomeArquivo) throws Exception {
        FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(lista);
        oos.close();
        fos.close();
    }

    public void recuperar(String nomeArquivo) throws Exception {
        FileInputStream fis = new FileInputStream(nomeArquivo);
        ObjectInputStream ois = new ObjectInputStream(fis);
        lista = (ArrayList<PessoaJuridica>) ois.readObject();
        ois.close();
        fis.close();
    }
}
```

PessoaFisicaRepo.java

```
package model;

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

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

    public void inserir(PessoaFisica pf) {
        lista.add(pf);
    }

    public void alterar(PessoaFisica pf) {
        for (int i = 0; i < lista.size(); i++) {
            if (lista.get(i).getId() == pf.getId()) {
                lista.set(i, pf);
                break;
            }
        }
    }

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

    public PessoaFisica obter(int id) {
        for (PessoaFisica pf : lista) {
            if (pf.getId() == id) return pf;
        }
        return null;
    }
```

```
    public ArrayList<PessoaFisica> obterTodos() {
        return lista;
    }

    public void persistir(String nomeArquivo) throws Exception {
        FileOutputStream fos = new FileOutputStream(nomeArquivo);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(lista);
        oos.close();
        fos.close();
    }

    public void recuperar(String nomeArquivo) throws Exception {
        FileInputStream fis = new FileInputStream(nomeArquivo);
        ObjectInputStream ois = new ObjectInputStream(fis);
        lista = (ArrayList<PessoaFisica>) ois.readObject();
        ois.close();
        fis.close();
    }
}
```

Pessoa.java

```
package model;

import java.io.Serializable;

public class Pessoa implements Serializable {
```

```
protected int id;
protected String nome;

public Pessoa() {}

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

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

PessoaJuridica.java

```
package model;

public class PessoaJuridica extends Pessoa {
    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;
    }
}
```

Main.java

```
package main;

import model.*;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();
        int opcao;

        do {
            System.out.println("\n=== MENU ===");
            System.out.println("1 - Incluir");
            System.out.println("2 - Alterar");
            System.out.println("3 - Excluir");
            System.out.println("4 - Exibir por ID");
            System.out.println("5 - Exibir todos");
            System.out.println("6 - Salvar dados");
            System.out.println("7 - Recuperar dados");
```

```
System.out.println("0 - Sair");
System.out.print("Opção: ");
opcao = Integer.parseInt(sc.nextLine());

switch (opcao) {
    case 1:
        System.out.print("Tipo (F para Física, J para Jurídica): ");
        String tipo = sc.nextLine();
        System.out.print("ID: ");
        int id = Integer.parseInt(sc.nextLine());
        System.out.print("Nome: ");
        String nome = sc.nextLine();
        if (tipo.equalsIgnoreCase("F")) {
            System.out.print("CPF: ");
            String cpf = sc.nextLine();
            System.out.print("Idade: ");
            int idade = Integer.parseInt(sc.nextLine());
            repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
        } else {
            System.out.print("CNPJ: ");
            String cnpj = sc.nextLine();
            repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
        }
        break;
    case 2:
        System.out.print("Tipo (F ou J): ");
        tipo = sc.nextLine();
```

```
System.out.print("ID: ");
id = Integer.parseInt(sc.nextLine());
if (tipo.equalsIgnoreCase("F")) {
    PessoaFisica pf = repoFisica.obter(id);
    if (pf != null) {
        System.out.println("Dados atuais:");
        pf.exibir();
        System.out.print("Novo nome: ");
        pf.setNome(sc.nextLine());
        System.out.print("Novo CPF: ");
        pf.setCpf(sc.nextLine());
        System.out.print("Nova idade: ");
        pf.setIdade(Integer.parseInt(sc.nextLine()));
        repoFisica.alterar(pf);
    }
} else {
    PessoaJuridica pj = repoJuridica.obter(id);
    if (pj != null) {
        System.out.println("Dados atuais:");
        pj.exibir();
        System.out.print("Novo nome: ");
        pj.setNome(sc.nextLine());
        System.out.print("Novo CNPJ: ");
        pj.setCnpj(sc.nextLine());
        repoJuridica.alterar(pj);
    }
}
```

```
            break;
        case 3:
            System.out.print("Tipo (F ou J): ");
            tipo = sc.nextLine();
            System.out.print("ID para excluir: ");
            id = Integer.parseInt(sc.nextLine());
            if (tipo.equalsIgnoreCase("F")) {
                repoFisica.excluir(id);
            } else {
                repoJuridica.excluir(id);
            }
            break;
        case 4:
            System.out.print("Tipo (F ou J): ");
            tipo = sc.nextLine();
            System.out.print("ID: ");
            id = Integer.parseInt(sc.nextLine());
            if (tipo.equalsIgnoreCase("F")) {
                PessoaFisica pf = repoFisica.obter(id);
                if (pf != null) pf.exibir();
            } else {
                PessoaJuridica pj = repoJuridica.obter(id);
                if (pj != null) pj.exibir();
            }
            break;
        case 5:
            System.out.print("Tipo (F ou J): ");
```

```
            tipo = sc.nextLine();
            if (tipo.equalsIgnoreCase("F")) {
                for (PessoaFisica pf : repoFisica.obterTodos()) {
                    pf.exibir();
                    System.out.println("-----");
                }
            } else {
                for (PessoaJuridica pj : repoJuridica.obterTodos()) {
                    pj.exibir();
                    System.out.println("-----");
                }
            }
            break;
        case 6:
            System.out.print("Prefixo do arquivo: ");
            String prefixoSalvar = sc.nextLine();
            try {
                repoFisica.persistir(prefixoSalvar + ".fisica.bin");
                repoJuridica.persistir(prefixoSalvar + ".juridica.bin");
            } catch (Exception e) {
                System.out.println("Erro ao salvar: " + e.getMessage());
            }
            break;
        case 7:
            System.out.print("Prefixo do arquivo: ");
            String prefixoRecuperar = sc.nextLine();
            try {
```

```
                repoFisica.recuperar(prefixoRecuperar + ".fisica.bin");
                repoJuridica.recuperar(prefixoRecuperar + ".juridica.bin");
            } catch (Exception e) {
                System.out.println("Erro ao recuperar: " + e.getMessage());
            }
            break;
        case 0:
            System.out.println("Encerrando...");
            break;
        default:
            System.out.println("Opção inválida.");
      }
    } while (opcao != 0);
    sc.close();
  }
}
```

PessoaFisica.java

```
package model;

public class PessoaFisica extends Pessoa {
  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);
        System.out.println("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;
    }
}
```

Resultados da Execução

A aplicação foi executada com sucesso no terminal, apresentando todas as funcionalidades conforme solicitado: inserção, alteração, exclusão, exibição por ID, listagem completa, salvamento e recuperação dos dados de pessoas físicas e jurídicas. A interface em modo texto permitiu interação simples e eficaz com o usuário.

Análise e Conclusão

- Vantagens da herança: permite reaproveitamento de código, maior organização e manutenção facilitada.

- Desvantagens da herança: pode gerar forte acoplamento e dificuldade de manutenção em hierarquias muito profundas.

- Serializable é necessário para que os objetos possam ser transformados em uma sequência de bytes, permitindo salvá-los e recuperá-los em arquivos binários.

- O paradigma funcional com API Stream do Java permite operar coleções de forma mais expressiva e com menos código, embora não tenha sido foco desta prática.

- O padrão adotado na persistência foi a serialização de objetos diretamente para arquivos binários (.bin).

- O método main é estático pois é o ponto de entrada da aplicação, e precisa ser acessível sem que uma instância da classe seja criada.

- A classe Scanner foi utilizada para capturar dados digitados pelo usuário no terminal.

- O uso de classes de repositório ajudou na separação de responsabilidades, organizando melhor o código e facilitando a manutenção.

Repositório no GitHub

Link do repositório: https://github.com/MarioELY/CadastroClientesJava