Desenvolvimento de um Sistema de Cadastro de Pessoas Utilizando POO em Java

Fábio Agostinho da Silva Nascimento Filho - 202405004477

Campus Via Corpvs
Programação Back-end com Java – DGT2821 – 2025.4

Objetivo da Prática

Compreender e aplicar os conceitos básicos de Programação Orientada a Objetos em Java, por meio da criação de classes, métodos e organização em pacotes, além de executar e testar a aplicação utilizando o método main, garantindo o funcionamento correto do sistema desenvolvido.

1º Procedimento | Criação das Entidades e Sistema de Persistência

Códigos para o primeiro procedimento:

CadastroPOO.java:

```
package cadastropoo;


import model.*;


public class CadastroPOO {


    public static void main(String[] args) {


        try {
            // ============================
            // Pessoa Física
            // ============================
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
```

```
repo1.inserir(new PessoaFisica(1, "Ana", "11111111111", 25));
repo1.inserir(new PessoaFisica(2, "Carlos", "22222222222", 52));

repo1.persistir("pf.bin");
System.out.println("Dados de Pessoa Fisica Armazenados.");

PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
repo2.recuperar("pf.bin");
System.out.println("Dados de Pessoa Fisica Recuperados.");

for (PessoaFisica pf : repo2.obterTodos()) {
   pf.exibir();
}

// ========================
// Pessoa Jurídica
// ========================
PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
repo3.inserir(new PessoaJuridica(3, "XPTO Sales", "33333333333333"));
repo3.inserir(new PessoaJuridica(4, "XPTO Solutions", "44444444444444"));

repo3.persistir("pj.bin");
System.out.println("Dados de Pessoa Juridica Armazenados.");

PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
repo4.recuperar("pj.bin");
System.out.println("Dados de Pessoa Juridica Recuperados.");
```

```
            for (PessoaJuridica pj : repo4.obterTodos()) {
                pj.exibir();
            }

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

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; }
}
```

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; }
}
```

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; }
}
```

PessoaFisicaRepo.java:

```
package model;

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

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);
```

```
            return;
        }
    }
}

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

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

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

public void persistir(String arquivo) throws Exception {
    ObjectOutputStream out =
            new ObjectOutputStream(new FileOutputStream(arquivo));
    out.writeObject(lista);
    out.close();
}
```

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

PessoaJuridicaRepo.java:

```
package model;

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

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);
                return;
            }
```

```
    }
}

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

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

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

public void persistir(String arquivo) throws Exception {
    ObjectOutputStream out =
            new ObjectOutputStream(new FileOutputStream(arquivo));
    out.writeObject(lista);
    out.close();
}

public void recuperar(String arquivo) throws Exception {
    ObjectInputStream in =
```

```
            new ObjectInputStream(new FileInputStream(arquivo));
        lista = (ArrayList<PessoaJuridica>) in.readObject();
        in.close();
    }
}
```

a. Quais as vantagens e desvantagens do uso de herança?

- A herança permite reutilizar código e organizar melhor as classes, já que características comuns ficam na superclasse. No projeto, Pessoa concentrou atributos básicos, enquanto PessoaFisica e PessoaJuridica adicionaram seus próprios dados. Como desvantagem, a herança pode aumentar o acoplamento entre as classes, dificultando mudanças futuras se a hierarquia não for bem planejada.

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

- A interface Serializable é necessária para que os objetos possam ser convertidos em bytes e gravados em arquivos binários. Sem ela, o Java não consegue salvar nem recuperar os objetos usando ObjectOutputStream e ObjectInputStream. No sistema, ela permitiu armazenar e recuperar os dados das pessoas diretamente dos arquivos.

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

- O paradigma funcional aparece no uso de expressões lambda e operações sobre coleções. Um exemplo é o método removeIf, que usa uma função para verificar qual elemento deve ser removido da lista. Isso torna o código mais direto e legível, evitando laços tradicionais mais longos.

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

- Foi utilizado o padrão Repository, onde classes específicas (repositórios) ficam responsáveis por salvar, recuperar e manipular os dados. Assim, a lógica de persistência fica separada das entidades, deixando o código mais organizado e fácil de manter.

2º Procedimento | Criação do Cadastro em Modo Texto

Código atualizado da main para o Segundo procedimento:

CadastroPOO.java:

```
package cadastropoo;

import java.util.Scanner;
import model.*;

public class CadastroPOO {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        PessoaFisicaRepo repoPF = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoPJ = new PessoaJuridicaRepo();

        int opcao = -1;

        while (opcao != 0) {

            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("==================================");

opcao = sc.nextInt();
sc.nextLine();

if (opcao >= 1 && opcao <= 5) {
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    char tipo = sc.nextLine().toUpperCase().charAt(0);

    switch (opcao) {

        case 1: // Incluir
            System.out.println("Digite o id da pessoa:");
            int id = sc.nextInt(); sc.nextLine();

            System.out.println("Insira os dados...");
            System.out.print("Nome: ");
            String nome = sc.nextLine();

            if (tipo == 'F') {
                System.out.print("CPF: ");
                String cpf = sc.nextLine();
                System.out.print("Idade: ");
                int idade = sc.nextInt();
```

```
            repoPF.inserir(new PessoaFisica(id, nome, cpf, idade));
        } else {
            System.out.print("CNPJ: ");
            String cnpj = sc.nextLine();
            repoPJ.inserir(new PessoaJuridica(id, nome, cnpj));
        }
        break;

    case 2: // Alterar
        System.out.print("Digite o id da pessoa: ");
        int idAlt = sc.nextInt(); sc.nextLine();

        if (tipo == 'F') {
            PessoaFisica pf = repoPF.obter(idAlt);
            if (pf != null) {
                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(sc.nextInt());
                repoPF.alterar(pf);
            }
        } else {
            PessoaJuridica pj = repoPJ.obter(idAlt);
            if (pj != null) {
```

```
                pj.exibir();
                System.out.print("Novo nome: ");
                pj.setNome(sc.nextLine());
                System.out.print("Novo CNPJ: ");
                pj.setCnpj(sc.nextLine());
                repoPJ.alterar(pj);
            }
        }
        break;

    case 3: // Excluir
        System.out.print("Digite o id da pessoa: ");
        int idExc = sc.nextInt();

        if (tipo == 'F') repoPF.excluir(idExc);
        else repoPJ.excluir(idExc);
        break;

    case 4: // Buscar por ID
        System.out.print("Digite o id da pessoa: ");
        int idBusca = sc.nextInt();

        if (tipo == 'F') {
            PessoaFisica pf = repoPF.obter(idBusca);
            if (pf != null) pf.exibir();
        } else {
            PessoaJuridica pj = repoPJ.obter(idBusca);
```

```
                if (pj != null) pj.exibir();
            }
            break;

        case 5: // Exibir todos
            if (tipo == 'F') {
                for (PessoaFisica pf : repoPF.obterTodos()) pf.exibir();
            } else {
                for (PessoaJuridica pj : repoPJ.obterTodos()) pj.exibir();
            }
            break;
    }
}

else if (opcao == 6) {
    try {
        System.out.print("Prefixo do arquivo: ");
        String prefixo = sc.nextLine();
        repoPF.persistir(prefixo + ".fisica.bin");
        repoPJ.persistir(prefixo + ".juridica.bin");
        System.out.println("Dados persistidos com sucesso!");
    } catch (Exception e) {
        System.out.println("Erro ao persistir: " + e.getMessage());
    }
}

else if (opcao == 7) {
```

```
            try {
                System.out.print("Prefixo do arquivo: ");
                String prefixo = sc.nextLine();
                repoPF.recuperar(prefixo + ".fisica.bin");
                repoPJ.recuperar(prefixo + ".juridica.bin");
                System.out.println("Dados recuperados com sucesso!");
            } catch (Exception e) {
                System.out.println("Erro ao recuperar: " + e.getMessage());
            }
        }

        else if (opcao == 0) {
            System.out.println("Programa finalizado.");
        }
    }

    sc.close();
  }
}
```

a. que são elementos estáticos e qual o motivo para o método main adotar esse modificador?

- Elementos estáticos pertencem à classe e não dependem de objetos para serem utilizados. O método main é estático porque ele é o ponto de entrada do

programa, então o Java precisa executá-lo sem criar uma instância da classe principal.

b. Para que serve a classe Scanner?

- A classe Scanner serve para capturar dados digitados pelo usuário no teclado. No sistema desenvolvido, ela foi utilizada para implementar o menu em modo texto, permitindo incluir, alterar, excluir e consultar pessoas de forma interativa.

c. Como o uso de classes de repositório impactou na organização do código?

- As classes de repositório ajudaram a separar a lógica de manipulação dos dados da lógica principal do programa. Com isso, as entidades ficaram responsáveis apenas pelos atributos, enquanto os repositórios cuidam das operações de inserção, alteração, exclusão e persistência, deixando o código mais organizado e fácil de manter.

Conclusão

A prática permitiu compreender melhor os conceitos de Programação Orientada a Objetos, especialmente a criação de classes, métodos e a organização do código em pacotes. Foi possível testar a execução da aplicação, validar o funcionamento da main e identificar possíveis ajustes necessários durante o desenvolvimento. Com isso, consolidamos o uso de POO em Java na prática, entendendo melhor como estruturar projetos e executar corretamente a aplicação.