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

Roberto de Freitas Marins - 202410102261

Polo West Shopping – Campo Grande - Rj

Programação Back-end com Java – 9001 – 2025.2

Repositório do Projeto (GitHub): https://github.com/robertofmarins/cadastro-poo-java

Objetivo da Prática

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.

1º Procedimento | Criação das Entidades e Sistema de Persistência

Classe Pessoa (`Pessoa.java`)

```java
package model;

import java.io.Serializable;

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);
       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; }
}
```

Classe PessoaFisica (`PessoaFisica.java`)

```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; }
}
```

Classe PessoaJuridica (`PessoaJuridica.java`)

```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; }
}
```

Repositório PessoaFisica (`PessoaFisicaRepo.java`)

```java
package model;

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

public class PessoaFisicaRepo implements Serializable {

   private ArrayList<PessoaFisica> pessoas = new ArrayList<>();

   public void inserir(PessoaFisica pessoa) {
       pessoas.add(pessoa);
   }

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

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

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

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

   public void persistir(String nomeArquivo) throws IOException {
       ObjectOutputStream oos =
               new ObjectOutputStream(new FileOutputStream(nomeArquivo));
       oos.writeObject(pessoas);
       oos.close();
   }

   @SuppressWarnings("unchecked")
   public void recuperar(String nomeArquivo)
           throws IOException, ClassNotFoundException {

       ObjectInputStream ois =
               new ObjectInputStream(new FileInputStream(nomeArquivo));
       pessoas = (ArrayList<PessoaFisica>) ois.readObject();
       ois.close();
   }
}
```

Repositório PessoaJuridica (`PessoaJuridicaRepo.java`)

```
package model;

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

public class PessoaJuridicaRepo implements Serializable {

   private ArrayList<PessoaJuridica> pessoas = new ArrayList<>();

   public void inserir(PessoaJuridica pessoa) {
       pessoas.add(pessoa);
   }

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

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

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

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

   public void persistir(String nomeArquivo) throws IOException {
       ObjectOutputStream oos =
               new ObjectOutputStream(new FileOutputStream(nomeArquivo));
       oos.writeObject(pessoas);
       oos.close();
   }

   @SuppressWarnings("unchecked")
   public void recuperar(String nomeArquivo)
           throws IOException, ClassNotFoundException {

       ObjectInputStream ois =
               new ObjectInputStream(new FileInputStream(nomeArquivo));
       pessoas = (ArrayList<PessoaJuridica>) ois.readObject();
       ois.close();
   }
}
```

Classe Principal (`Main.java`)

```
import model.*;

public class Main {

    public static void main(String[] args) {

        try {
            /* ================================
               TESTE PESSOA FÍSICA
               ================================ */

            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

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

            String arquivoPF = "pessoas_fisicas.dat";
            repo1.persistir(arquivoPF);

            System.out.println("Dados de Pessoa Fisica Armazenados.");

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar(arquivoPF);

            System.out.println("Dados de Pessoa Fisica Recuperados.");

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

            /* ================================
               TESTE PESSOA JURÍDICA
               ================================ */

            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

            repo3.inserir(new PessoaJuridica(3, "XPTO Sales",
"33333333333333"));
            repo3.inserir(new PessoaJuridica(4, "XPTO Solutions",
"44444444444444"));

            String arquivoPJ = "pessoas_juridicas.dat";
            repo3.persistir(arquivoPJ);

            System.out.println("Dados de Pessoa Juridica Armazenados.");

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
            repo4.recuperar(arquivoPJ);

            System.out.println("Dados de Pessoa Juridica Recuperados.");

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

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

Resultado da Execução

```
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados.
Id: 1
Nome: Ana
CPF: 11111111111
Idade: 25
Id: 2
Nome: Carlos
CPF: 22222222222
Idade: 52
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados.
Id: 3
Nome: XPTO Sales
CNPJ: 33333333333333
Id: 4
Nome: XPTO Solutions
CNPJ: 44444444444444

Process finished with exit code 0
```

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

A herança facilita o reaproveitamento de código, pois permite criar uma classe base com informações comuns e outras classes que herdam essas características. No trabalho, isso acontece com a classe `Pessoa`, que é usada pelas classes `PessoaFisica` e `PessoaJuridica`. Isso deixa o código mais organizado e evita repetição.

Como desvantagem, se a herança não for bem planejada, pode deixar o sistema difícil de modificar no futuro, pois mudanças na classe base podem afetar todas as classes que herdam dela.

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

A interface `Serializable` é necessária porque ela permite que os objetos sejam transformados em dados que podem ser gravados em arquivos binários. Sem essa interface, o Java não permite salvar os objetos em arquivo. No trabalho, ela foi usada para garantir que os dados das pessoas físicas e jurídicas pudessem ser salvos e recuperados corretamente do disco.

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

O paradigma funcional é utilizado no Java através de recursos como expressões lambda, que permitem escrever funções de forma mais simples. Um exemplo visto no trabalho é o uso do método `removeIf`, que utiliza uma função para remover elementos de uma lista com base em uma condição. Isso torna o código mais curto, fácil de entender e mais organizado.

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

Na persistência de dados em arquivos no Java, é comum utilizar o padrão Repository, que separa a parte responsável por guardar e recuperar os dados da lógica principal do sistema. No trabalho, esse padrão foi utilizado nas classes de repositório, que ficaram responsáveis por salvar, recuperar e gerenciar os dados das pessoas físicas e jurídicas, deixando o código mais organizado.

2º Procedimento | Criação do Cadastro em Modo Texto

Neste procedimento, foi implementada uma interface de cadastro em modo texto, permitindo a interação do usuário com o sistema através do teclado.

As classes de entidades e repositórios desenvolvidas no primeiro procedimento foram reaproveitadas, sendo necessária apenas a alteração da classe principal (`Main`) para incluir o menu de opções, leitura de dados com `Scanner` e tratamento de exceções.

Classe Principal (`Main.java`)

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

public class Main {

   public static void main(String[] args) {

       Scanner scanner = new Scanner(System.in);

       PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
       PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

       int opcao;

       do {
           System.out.println("\n================================");
           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: ");

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

           switch (opcao) {
               case 1:
                   System.out.print("Tipo (F - Física | J - Jurídica): ");
                   String tipo = scanner.nextLine();

                   if (tipo.equalsIgnoreCase("F")) {

                       System.out.print("ID: ");
                       int id = scanner.nextInt();
                       scanner.nextLine();
```

```
                System.out.print("Nome: ");
                String nome = scanner.nextLine();

                System.out.print("CPF: ");
                String cpf = scanner.nextLine();

                System.out.print("Idade: ");
                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 if (tipo.equalsIgnoreCase("J")) {

                System.out.print("ID: ");
                int id = scanner.nextInt();
                scanner.nextLine();

                System.out.print("Nome: ");
                String nome = scanner.nextLine();

                System.out.print("CNPJ: ");
                String cnpj = scanner.nextLine();

                PessoaJuridica pj = new PessoaJuridica(id, nome, cnpj);
                repoJuridica.inserir(pj);

                System.out.println("Pessoa Jurídica incluída com sucesso.");

            } else {
                System.out.println("Tipo inválido.");
            }
            break;
        case 2:
            System.out.print("Tipo (F - Física | J - Jurídica): ");
            String tipoAlt = scanner.nextLine();

            System.out.print("ID: ");
            int idAlt = scanner.nextInt();
            scanner.nextLine();

            if (tipoAlt.equalsIgnoreCase("F")) {
                PessoaFisica pf = repoFisica.obter(idAlt);
                if (pf != null) {
                    pf.exibir();

                    System.out.print("Novo nome: ");
                    String nome = scanner.nextLine();

                    System.out.print("Novo CPF: ");
                    String cpf = scanner.nextLine();

                    System.out.print("Nova idade: ");
                    int idade = scanner.nextInt();
                    scanner.nextLine();

                    PessoaFisica novaPf = new PessoaFisica(idAlt, nome, cpf, idade);
                    repoFisica.alterar(novaPf);
```

```
                    System.out.println("Pessoa Física alterada com sucesso.");
                } else {
                    System.out.println("Pessoa Física não encontrada.");
                }

            } else if (tipoAlt.equalsIgnoreCase("J")) {
                PessoaJuridica pj = repoJuridica.obter(idAlt);
                if (pj != null) {
                    pj.exibir();

                    System.out.print("Novo nome: ");
                    String nome = scanner.nextLine();

                    System.out.print("Novo CNPJ: ");
                    String cnpj = scanner.nextLine();

                    PessoaJuridica novaPj = new PessoaJuridica(idAlt, nome, cnpj);
                    repoJuridica.alterar(novaPj);

                    System.out.println("Pessoa Jurídica alterada com sucesso.");
                } else {
                    System.out.println("Pessoa Jurídica não encontrada.");
                }

            } else {
                System.out.println("Tipo inválido.");
            }
            break;

        case 3:
            System.out.print("Tipo (F - Física | J - Jurídica): ");
            String tipoExc = scanner.nextLine();

            System.out.print("ID: ");
            int idExc = scanner.nextInt();
            scanner.nextLine();

            if (tipoExc.equalsIgnoreCase("F")) {
                PessoaFisica pf = repoFisica.obter(idExc);
                if (pf != null) {
                    repoFisica.excluir(idExc);
                    System.out.println("Pessoa Física excluída com sucesso.");
                } else {
                    System.out.println("Pessoa Física não encontrada.");
                }

            } else if (tipoExc.equalsIgnoreCase("J")) {
                PessoaJuridica pj = repoJuridica.obter(idExc);
                if (pj != null) {
                    repoJuridica.excluir(idExc);
                    System.out.println("Pessoa Jurídica excluída com sucesso.");
                } else {
                    System.out.println("Pessoa Jurídica não encontrada.");
                }

            } else {
                System.out.println("Tipo inválido.");
            }
            break;

        case 4:
            System.out.print("Tipo (F - Física | J - Jurídica): ");
```

```
                String tipoId = scanner.nextLine();

                System.out.print("ID: ");
                int idBusca = scanner.nextInt();
                scanner.nextLine();

                if (tipoId.equalsIgnoreCase("F")) {
                    PessoaFisica pf = repoFisica.obter(idBusca);
                    if (pf != null) {
                        pf.exibir();
                    } else {
                        System.out.println("Pessoa Física não encontrada.");
                    }

                } else if (tipoId.equalsIgnoreCase("J")) {
                    PessoaJuridica pj = repoJuridica.obter(idBusca);
                    if (pj != null) {
                        pj.exibir();
                    } else {
                        System.out.println("Pessoa Jurídica não encontrada.");
                    }

                } else {
                    System.out.println("Tipo inválido.");
                }
                break;

            case 5:
                System.out.print("Tipo (F - Física | J - Jurídica): ");
                String tipoExibir = scanner.nextLine();

                if (tipoExibir.equalsIgnoreCase("F")) {
                    if (repoFisica.obterTodos().isEmpty()) {
                        System.out.println("Nenhuma Pessoa Física cadastrada.");
                    } else {
                        for (PessoaFisica pf : repoFisica.obterTodos()) {
                            pf.exibir();
                            System.out.println("--------------------");
                        }
                    }

                } else if (tipoExibir.equalsIgnoreCase("J")) {
                    if (repoJuridica.obterTodos().isEmpty()) {
                        System.out.println("Nenhuma Pessoa Jurídica cadastrada.");
                    } else {
                        for (PessoaJuridica pj : repoJuridica.obterTodos()) {
                            pj.exibir();
                            System.out.println("--------------------");
                        }
                    }

                } else {
                    System.out.println("Tipo inválido.");
                }
                break;

            case 6:
                try {
                    System.out.print("Prefixo do arquivo: ");
                    String prefixo = scanner.nextLine();

                    repoFisica.persistir(prefixo + ".fisica.bin");
```

```
                        repoJuridica.persistir(prefixo + ".juridica.bin");

                        System.out.println("Dados salvos com sucesso.");
                    } catch (Exception e) {
                        System.out.println("Erro ao salvar os dados.");
                    }
                    break;

                case 7:
                    try {
                        System.out.print("Prefixo do arquivo: ");
                        String prefixo = scanner.nextLine();

                        repoFisica.recuperar(prefixo + ".fisica.bin");
                        repoJuridica.recuperar(prefixo + ".juridica.bin");

                        System.out.println("Dados recuperados com sucesso.");
                    } catch (Exception e) {
                        System.out.println("Erro ao recuperar os dados.");
                    }
                    break;

                case 0:
                    System.out.println("Sistema finalizado.");
                    break;
                default:
                    System.out.println("Opção inválida.");
            }

        } while (opcao != 0);

        scanner.close();
    }
}
```

Resultado da Execução

```
==================================
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:
```

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

Elementos estáticos são aqueles que pertencem à classe e não a uma instância específica. Eles podem ser acessados diretamente pela classe, sem a necessidade de criar um objeto.

O método `main` é declarado como estático porque ele é o ponto de entrada da aplicação Java e precisa ser executado pela JVM sem que exista previamente um objeto da classe principal. Dessa forma, o programa pode ser iniciado corretamente.

B. Para que serve a classe Scanner?

A classe `Scanner` é utilizada para realizar a leitura de dados digitados pelo usuário através do teclado. No trabalho, ela foi usada para capturar informações como opções do menu, nomes, CPF, CNPJ, idade e prefixos de arquivos, permitindo a interação do usuário com o sistema em modo texto.

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

O uso das classes de repositório ajudou a separar as responsabilidades do sistema. Enquanto a classe `Main` ficou responsável apenas pela interação com o usuário, os repositórios ficaram responsáveis por inserir, alterar, excluir, salvar e recuperar os dados. Isso deixou o código mais organizado, mais fácil de entender e de manter.

Conclusão

A missão prática permitiu aplicar conceitos importantes da programação orientada a objetos, como herança, polimorfismo e separação de responsabilidades. A utilização de arquivos binários possibilitou compreender como funciona a persistência de dados no Java, enquanto a criação do cadastro em modo texto ajudou a entender a interação do usuário com o sistema. O uso de repositórios deixou o código mais organizado e facilitou a manutenção, contribuindo para um melhor aprendizado dos conceitos estudados na disciplina.