Implementação de um Sistema Cadastral em Java Utilizando Herança, Polimorfismo e Persistência em Arquivos Binários

2º Procedimento | Criação do Cadastro em Modo Texto

Aluno: Caio Viana Castelo Branco

Matrícula: 202307465925

Disciplina: RPG0014 - Iniciando o caminho pelo Java

Curso: Desenvolvimento Full Stack

Turma: 9001

Semestre Letivo: 2025.1

Campus: Parangaba

1. Objetivo da Prática

- 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.

2. Códigos utilizados na prática

Classe principal: CadastroPOO.java

```
package cadastropoo;

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

public class CadastroPOO {
  private static final PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
  private static final PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();
  private static final Scanner scanner = new Scanner(System.in);

  public static void main(String[] args) {
    int opcao;
    do {
      exibirMenu();
      opcao = lerInt("Escolha uma alternativa: ");
      processarOpcao(opcao);
    } while (opcao != 0);
    scanner.close();
  }

  private static void exibirMenu() {
    System.out.println("\n=========================\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\n");
        System.out.println("\n=========================\n");
    }

    private static void processarOpcao(int opcao) {
        switch (opcao) {
            case 1 -> incluirPessoa();
            case 2 -> alterarPessoa();
            case 3 -> excluirPessoa();
            case 4 -> buscarPorId();
            case 5 -> exibirTodos();
            case 6 -> persistirDados();
            case 7 -> recuperarDados();
            case 0 -> System.out.println("Programa finalizado.");
            default -> System.out.println("Alternativa inválida!");
        }
    }

    private static int lerInt(String mensagem) {
        System.out.print(mensagem);
        return Integer.parseInt(scanner.nextLine());
    }

    private static String lerString(String mensagem) {
        System.out.print(mensagem);
        return scanner.nextLine();
    }

    private static char lerTipoPessoa() {
        System.out.print("\nF - Pessoa Fisica\nJ - Pessoa Juridica\nDigite a alternativa
que deseja: ");
        return scanner.nextLine().toUpperCase().charAt(0);
    }

    private static void incluirPessoa() {
        char tipo = lerTipoPessoa();
        int id = lerInt("Digite o ID: ");
        String nome = lerString("Nome: ");

        if (tipo == 'F') {
            String cpf = lerString("CPF: ");
            int idade = lerInt("Idade: ");
            repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
            System.out.println("Pessoa física cadastrada com sucesso!");
        } else {
```

```
            String cnpj = lerString("CNPJ: ");
            repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
            System.out.println("Pessoa juridica cadastrada com sucesso!");
        }
    }

    private static void alterarPessoa() {
        char tipo = lerTipoPessoa();
        int id = lerInt("Digite o ID para alteração: ");

        if (tipo == 'F') {
            PessoaFisica pf = repoFisica.obter(id);
            if (pf != null) {
                System.out.println("Dados atuais:");
                pf.exibir();
                pf.setNome(lerString("Novo nome: "));
                pf.setCpf(lerString("Novo CPF: "));
                pf.setIdade(lerInt("Nova idade: "));
                repoFisica.alterar(pf);
                System.out.println("Pessoa física alterada com sucesso!");
            } else {
                System.out.println("Pessoa nao encontrada!");
            }
        } else {
            PessoaJuridica pj = repoJuridica.obter(id);
            if (pj != null) {
                System.out.println("Dados atuais:");
                pj.exibir();
                pj.setNome(lerString("Novo nome: "));
                pj.setCnpj(lerString("Novo CNPJ: "));
                repoJuridica.alterar(pj);
                System.out.println("Pessoa juridica alterada com sucesso!");
            } else {
                System.out.println("Pessoa nao encontrada!");
            }
        }
    }

    private static void excluirPessoa() {
        char tipo = lerTipoPessoa();
        int id = lerInt("Digite o ID para exclusao: ");

        if (tipo == 'F') {
            if (repoFisica.excluir(id)) {
                System.out.println("Pessoa fisica excluida com sucesso!");
            } else {
                System.out.println("Pessoa nao encontrada!");
```

```
        }
    } else {
        if (repoJuridica.excluir(id)) {
            System.out.println("Pessoa juridica excluida com sucesso!");
        } else {
            System.out.println("Pessoa nao encontrada!");
        }
    }
}

private static void buscarPorId() {
    char tipo = lerTipoPessoa();
    int id = lerInt("Digite o ID para busca: ");

    if (tipo == 'F') {
        PessoaFisica pf = repoFisica.obter(id);
        if (pf != null) {
            pf.exibir();
        } else {
            System.out.println("Pessoa nao encontrada!");
        }
    } else {
        PessoaJuridica pj = repoJuridica.obter(id);
        if (pj != null) {
            pj.exibir();
        } else {
            System.out.println("Pessoa nao encontrada!");
        }
    }
}

private static void exibirTodos() {
    char tipo = lerTipoPessoa();

    if (tipo == 'F') {
        System.out.println("\nLista de todas as pessoas fisicas:\n");
        for (PessoaFisica pf : repoFisica.obterTodos()) {
            pf.exibir();
            System.out.println("------------------");
        }
    } else {
        System.out.println("\nLista de todas as pessoas juridicas:\n");
        for (PessoaJuridica pj : repoJuridica.obterTodos()) {
            pj.exibir();
            System.out.println("------------------");
        }
    }
```

```
    }

    private static void persistirDados() {
        try {
            String prefixo = lerString("Digite o prefixo para os arquivos: ");
            repoFisica.persistir(prefixo + ".fisica.bin");
            repoJuridica.persistir(prefixo + ".juridica.bin");
            System.out.println("Dados salvos com sucesso!");
        } catch (IOException e) {
            System.err.println("Erro ao salvar: " + e.getMessage());
        }
    }

    private static void recuperarDados() {
        try {
            String prefixo = lerString("Digite o prefixo dos arquivos: ");
            repoFisica.recuperar(prefixo + ".fisica.bin");
            repoJuridica.recuperar(prefixo + ".juridica.bin");
            System.out.println("Dados recuperados com sucesso!");
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Erro ao carregar: " + e.getMessage());
        }
    }
}
```

Classes de Modelo:

Pessoa.java

```
package model;
import java.io.Serializable;

public abstract 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 abstract void exibir();

    // Getters e Setters
    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;
import java.io.Serializable;

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() {
        System.out.println("Id: " + super.getId());
        System.out.println("Nome: " + super.getNome());
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
    }

    // Getters e Setters
    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;
import java.io.Serializable;

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() {
        System.out.println("Id: " + super.getId());
        System.out.println("Nome: " + super.getNome());
        System.out.println("CNPJ: " + cnpj);
    }

    // Getters e Setters
    public String getCnpj() { return cnpj; }
    public void setCnpj(String cnpj) { this.cnpj = cnpj; }
}
```

Repositórios:

PessoaFisicaRepo.java

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

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

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

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

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

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

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

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

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

PessoaJuridicaRepo.java

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

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

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

  public void alterar(PessoaJuridica pessoa) {
    for (int i = 0; i < pessoasJuridicas.size(); i++) {
```

```
        if (pessoasJuridicas.get(i).getId() == pessoa.getId()) {
          pessoasJuridicas.set(i, pessoa);
          break;
        }
      }
    }

    public boolean excluir(int id) {
      return pessoasJuridicas.removeIf(p -> p.getId() == id);
    }

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

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

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

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

3. Resultados da execução dos códigos

```
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


==========================

Escolha uma alternativa: 1

F - Pessoa Fisica
J - Pessoa Juridica
Digite a alternativa que deseja: F
Digite o ID: 1001
Nome: Caio
CPF: 1234567890
Idade: 29
Pessoa f◆sica cadastrada com sucesso!

==========================

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


==========================

Escolha uma alternativa: 5

F - Pessoa Fisica
J - Pessoa Juridica
Digite a alternativa que deseja: F

Lista de todas as pessoas fisicas:

Id: 1001
Nome: Caio
CPF: 1234567890
Idade: 29
------------------
```

Arquivos gerados:
pessoas_fisicas.dat
pessoas_juridicas.dat

4. Análise e Conclusão

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

  Elementos estáticos (static) pertencem à classe em vez de instâncias individuais. São carregados quando a classe é inicializada.

  Main é estático porque é o ponto de entrada do programa, precisando ser acessível sem instanciar classe alguma. Caso main ele não fosse static seria necessário criar um objeto para poder executar o programa, resultando em falha.

- Para que serve a classe Scanner?
  O Scanner é como um "tradutor" que usei para ler entradas do usuário através do teclado e converter dados de texto para outros tipos.

- Como o uso de classes de repositório impactou na organização do código?
  O paradigma funcional apareceu no uso da Stream API, como no método removeIf(), que simplificou a exclusão de itens com uma expressão lambda, tornando o código mais limpo.