

# Relatório de Acompanhamento - CadastroPOO

---

Luis Eduardo Wataro Nagata

---

## Missão 1 | Mundo 3

---

### Parte 1

### Objetivos:

---

- **Utilizar Herança e Polimorfismo:**
  - Aplicar conceitos de herança e polimorfismo para definir entidades no sistema.
- **Persistência em Arquivos Binários:**
  - Implementar a persistência de objetos em arquivos binários.
- **Interface Cadastral em Modo Texto:**
  - Criar uma interface de cadastro que funcione em modo de texto.
- **Controle de Exceções em Java:**
  - Utilizar o mecanismo de tratamento de exceções da plataforma Java.
- **Sistema Cadastral em Java:**
  - Ao final do projeto, o aluno terá desenvolvido um sistema cadastral em Java, aproveitando os recursos da programação orientada a objetos e a persistência em arquivos binários.

## Códigos Solicitados

---

A seguir, apresentamos os códigos desenvolvidos durante a prática:

- Class Pessoa

```
package model;  
  
public class Pessoa {  
  
    private int id;  
    private String nome;
```

```
// Construtor padrão
public Pessoa() {
}

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

// 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;
}

/**
 * Método para exibir os dados da Pessoa
 */
public void exibir() {
    System.out.println("ID: " + id);
    System.out.println("Nome: " + nome);
}
}
```

- Class Pessoa Física

```
package model;

import java.io.*;

public class PessoaFisica extends Pessoa implements Serializable {
    private int id;
    private String nome;
    private String cpf;
    private int idade;

    // Construtor padrão
    public PessoaFisica() {
    }
}
```

```
// Construtor completo
public PessoaFisica(int id, String nome, String cpf, int idade) {
    this.id = id;
    this.nome = nome;
    this.cpf = cpf;
    this.idade = idade;
}

// 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;
}

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;
}

// Método para exibir os dados da PessoaFisica
@Override
public void exibir() {
    super.exibir(); // Chama o método exibir da classe Pessoa
    System.out.println("CPF: " + cpf);
    System.out.println("Idade: " + idade);
}
}
```

- Class Pessoa Física Repo

```
package model;

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

/**
 * Representa um repositório para operações CRUD em pessoas físicas.
 */
public class PessoaFisicaRepo implements Serializable {

    /**
     * Para garantir que a codificação de caracteres UTF-8 seja usada ao ler e
     * escrever dados nos arquivos dessas classes.
     */
    private static final String CHARSET = "UTF-8";
    private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();

    /**
     * definir o serialVersionUID para o mesmo valor em todas as classes
     * serializáveis que podem ser gravadas em arquivos e recuperadas
     * posteriormente. Isso garante compatibilidade entre diferentes versões do
     * seu programa.
     */
    private static final long serialVersionUID = 123456789L;

    /**
     * Insere uma nova pessoa física no repositório.
     *
     * @param pessoaFisica A pessoa física a ser inserida.
     */
    public void inserir(PessoaFisica pessoaFisica) {
        pessoasFisicas.add(pessoaFisica);
    }

    /**
     * Altera uma pessoa física existente no repositório.
     *
     * @param pessoaFisica A pessoa física com as alterações a serem aplicadas.
     */
    public void alterar(PessoaFisica pessoaFisica) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(i).getId() == pessoaFisica.getId()) {
                pessoasFisicas.set(i, pessoaFisica);
                break;
            }
        }
    }

    /**
     * Exclui uma pessoa física do repositório pelo ID.
     */
}
```

```
*
* @param id O ID da pessoa física a ser excluída.
*/
public void excluir(int id) {
    for (int i = 0; i < pessoasFisicas.size(); i++) {
        if (pessoasFisicas.get(i).getId() == id) {
            pessoasFisicas.remove(i);
            break;
        }
    }
}

/**
 * Obtém uma pessoa física do repositório pelo ID.
 *
 * @param id O ID da pessoa física a ser obtida.
 * @return A pessoa física encontrada ou null se não encontrada.
 */
public PessoaFisica obter(int id) {
    for (PessoaFisica pessoa : pessoasFisicas) {
        if (pessoa.getId() == id) {
            return pessoa;
        }
    }
    return null;
}

/**
 * Obtém todas as pessoas físicas do repositório.
 *
 * @return Uma lista de todas as pessoas físicas no repositório.
 */
public ArrayList<PessoaFisica> obterTodos() {
    return pessoasFisicas;
}

/**
 * Persiste os dados das pessoas físicas em um arquivo.
 *
 * @param nomeArquivo O nome do arquivo onde os dados serão persistidos.
 * @throws IOException Se ocorrer um erro de I/O durante a persistência.
 */
public void persistir(String nomeArquivo) throws IOException {
    try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
        out.writeObject(pessoasFisicas);
    }
}

/**
 * Recupera os dados das pessoas físicas de um arquivo.
 *
 * @param nomeArquivo O nome do arquivo de onde os dados serão recuperados.
 */
}
```

```

    * @throws IOException Se ocorrer um erro de I/O durante a recuperação.
    * @throws ClassNotFoundException Se a classe das pessoas físicas não for
    * encontrada durante a recuperação.
    */
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            pessoasFisicas = (ArrayList<PessoaFisica>) in.readObject();
        }
    }
}

```

- Class Pessoa Jurídica

```

package model;

import java.io.Serializable;

/**
 * Representa uma pessoa jurídica.
 */
public class PessoaJuridica extends Pessoa implements Serializable {

    /**
     * definir o serialVersionUID para o mesmo valor em todas as classes
     * serializáveis que podem ser gravadas em arquivos e recuperadas
     * posteriormente. Isso garante compatibilidade entre diferentes versões do
     * seu programa.
     */
    private static final long serialVersionUID = 123456789L;
    private String cnpj;

    // Construtor completo
    public PessoaJuridica(int id, String nome, String cnpj) {
        super(id, nome); // Chama o construtor da classe Pai (Pessoa)
        this.cnpj = cnpj;
    }

    // Getters e Setters para os atributos específicos de PessoaJuridica
    public String getCnpj() {
        return cnpj;
    }

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

    // Sobrescreve o método exibir para incluir os dados específicos de Pessoa
    @Override
    public void exibir() {
        super.exibir(); // Chama o método exibir da classe Pai (Pessoa)
        System.out.println("CNPJ: " + cnpj);
    }
}

```

```
}  
}
```

- Class Pessoa Jurídica Repo

```
package model;  
  
import java.io.*;  
import java.util.ArrayList;  
  
/**  
 * Representa um repositório para operações CRUD em pessoas jurídicas.  
 */  
public class PessoaJuridicaRepo implements Serializable {  
  
    /**  
     * Para garantir que a codificação de caracteres UTF-8 seja usada ao ler e  
     * escrever dados nos arquivos dessas classes.  
     */  
    private static final String CHARSET = "UTF-8";  
    private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();  
    /**  
     * definir o serialVersionUID para o mesmo valor em todas as classes  
     * serializáveis que podem ser gravadas em arquivos e recuperadas  
     * posteriormente. Isso garante compatibilidade entre diferentes versões do  
     * seu programa.  
     */  
    private static final long serialVersionUID = 123456789L;  
  
    /**  
     * Insere uma nova pessoa jurídica no repositório.  
     *  
     * @param pessoaJuridica A pessoa jurídica a ser inserida.  
     */  
    public void inserir(PessoaJuridica pessoaJuridica) {  
        pessoasJuridicas.add(pessoaJuridica);  
    }  
  
    /**  
     * Altera uma pessoa jurídica existente no repositório.  
     *  
     * @param pessoaJuridica A pessoa jurídica com as alterações a serem  
     * aplicadas.  
     */  
    public void alterar(PessoaJuridica pessoaJuridica) {  
        for (int i = 0; i < pessoasJuridicas.size(); i++) {  
            if (pessoasJuridicas.get(i).getId() == pessoaJuridica.getId()) {  
                pessoasJuridicas.set(i, pessoaJuridica);  
                break;  
            }  
        }  
    }  
}
```

```
    }
}

/**
 * Exclui uma pessoa jurídica do repositório pelo ID.
 *
 * @param id O ID da pessoa jurídica a ser excluída.
 */
public void excluir(int id) {
    for (int i = 0; i < pessoasJuridicas.size(); i++) {
        if (pessoasJuridicas.get(i).getId() == id) {
            pessoasJuridicas.remove(i);
            break;
        }
    }
}

/**
 * Obtém uma pessoa jurídica do repositório pelo ID.
 *
 * @param id O ID da pessoa jurídica a ser obtida.
 * @return A pessoa jurídica encontrada ou null se não encontrada.
 */
public PessoaJuridica obter(int id) {
    for (PessoaJuridica pessoa : pessoasJuridicas) {
        if (pessoa.getId() == id) {
            return pessoa;
        }
    }
    return null;
}

/**
 * Obtém todas as pessoas jurídicas do repositório.
 *
 * @return Uma lista de todas as pessoas jurídicas no repositório.
 */
public ArrayList<PessoaJuridica> obterTodos() {
    return pessoasJuridicas;
}

/**
 * Persiste os dados das pessoas jurídicas em um arquivo.
 *
 * @param nomeArquivo O nome do arquivo onde os dados serão persistidos.
 * @throws IOException Se ocorrer um erro de I/O durante a persistência.
 */
public void persistir(String nomeArquivo) throws IOException {
    try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
        out.writeObject(pessoasJuridicas);
    }
}
```



```

/**
 * Recupera os dados das pessoas jurídicas de um arquivo.
 *
 * @param nomeArquivo O nome do arquivo de onde os dados serão recuperados.
 * @throws IOException Se ocorrer um erro de I/O durante a recuperação.
 * @throws ClassNotFoundException Se a classe das pessoas jurídicas não for
 * encontrada durante a recuperação.
 */
public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
    try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
        pessoasJuridicas = (ArrayList<PessoaJuridica>) in.readObject();
    }
}
}

```

- Class CadastroPOO

```

package cadastrapoo;

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

/**
 * Classe de exemplo para demonstrar o uso de repositórios de pessoas físicas
 */
public class CadastroPOO {

    public static void main(String[] args) throws Exception {
        String arquivoPessoas = "pessoas.dat";

        try {
            // Repositório de pessoas físicas
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            PessoaFisica pessoa1 = new PessoaFisica(1, "Joao", "123.456.789-00");
            PessoaFisica pessoa2 = new PessoaFisica(2, "Maria", "987.654.321-00");

            repo1.inserir(pessoa1);
            repo1.inserir(pessoa2);

            repo1.persistir(arquivoPessoas);
            System.out.println("Dados de Pessoa Fisica armazenados.");

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

            for (PessoaFisica pessoa : repo2.obterTodos()) {
                System.out.println("Id: " + pessoa.getId());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```
        System.out.println("Nome: " + pessoa.getNome());
        System.out.println("CPF: " + pessoa.getCpf());
        System.out.println("Idade: " + pessoa.getIdade());
    }
} catch (Exception e) {
    System.out.println("Erro: " + e.getMessage());
}

String arquivoEmpresas = "empresas.dat";

// Repositório de pessoas jurídicas
PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

// Adicionando duas empresas
PessoaJuridica empresa1 = new PessoaJuridica(1, "XPTO Sales", "12345678901234567890");
PessoaJuridica empresa2 = new PessoaJuridica(2, "XPTO Solutions", "98765432109876543210");

repo3.inserir(empresa1);
repo3.inserir(empresa2);

try {
    // Persistindo os dados em disco
    repo3.persistir(arquivoEmpresas);
    System.out.println("\nDados de Pessoa Juridica armazenados.");

    // Recuperando os dados
    PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
    repo4.recuperar(arquivoEmpresas);
    System.out.println("Dados de Pessoa Juridica Recuperados:");

    // Exibindo os dados das empresas recuperadas
    for (PessoaJuridica empresa : repo4.obterTodos()) {
        System.out.println("Id: " + empresa.getId());
        System.out.println("Nome: " + empresa.getNome());
        System.out.println("CNPJ: " + empresa.getCnpj());
    }
} catch (Exception e) {
    System.out.println("Erro: " + e.getMessage());
}

}
```

## Resultados da Execução dos Códigos

Os resultados obtidos ao executar os códigos foram os seguintes:

- Resultado

```
run:
Dados de Pessoa Fisica armazenados.
Dados de Pessoa Fisica Recuperados:
Id: 1
Nome: Joao
CPF: 123.456.789-01
Idade: 30
Id: 2
Nome: Maria
CPF: 987.654.321-09
Idade: 25

Dados de Pessoa Juridica armazenados.
Dados de Pessoa Juridica Recuperados:
Id: 1
Nome: XPT0 Sales
CNPJ: 12345678901234
Id: 2
Nome: XPT0 Solutions
CNPJ: 98765432109876
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Título da Prática

---

Missão Prática | Nível 1 | Mundo 3

- Parte 2

Alterações como solicitado no trabalho da parte 2.

- Class CadastroPOO

```
package cadastropoo;

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

/**
 * Classe de exemplo para demonstrar o uso de repositórios de pessoas físicas e
 * jurídicas.
 */
public class CadastroPOO {
```

```
public static void main(String[] args) {

    String arquivoPessoas = "pessoas.dat";
    String arquivoEmpresas = "empresas.dat";
    try (Scanner scanner = new Scanner(System.in)) {
        PessoaFisicaRepo fisicaRepo = new PessoaFisicaRepo();
        PessoaJuridicaRepo juridicaRepo = new PessoaJuridicaRepo();
        try {
            // Recuperando dados previamente armazenados
            fisicaRepo.recuperar(arquivoPessoas);
            juridicaRepo.recuperar(arquivoEmpresas);
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao recuperar os dados: " + e.getMessage());
        }
        int opcao;
        do {
            System.out.println("=====");
            System.out.println("Menu de Opcoes:");
            System.out.println("=====");
            System.out.println("1 - Incluir Pessoa/Empresa");
            System.out.println("2 - Alterar Pessoa/Empresa");
            System.out.println("3 - Excluir Pessoa/Empresa");
            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 opcao: ");
            opcao = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha pendente

            switch (opcao) {
                case 1 -> {
                    System.out.println("=====");
                    System.out.println("Escolha uma opcao para cadastrar:");
                    System.out.println("1 - Pessoa Fisica");
                    System.out.println("2 - Empresa");
                    int tipoPessoa = scanner.nextInt();
                    scanner.nextLine(); // Consumir a nova linha pendente
                    switch (tipoPessoa) {
                        case 1 -> {
                            // Cadastro de Pessoa Física
                            do {
                                System.out.println("=====");
                                System.out.println("Cadastro de Pessoa Fisica");
                                System.out.print("Informe o nome: ");
                                String nome = scanner.nextLine();
                                System.out.print("Informe o CPF: ");
                                String cpf = scanner.nextLine();
                                System.out.print("Informe a idade: ");
                                int idade = scanner.nextInt();
```

```
        fisicaRepo.inserir(new PessoaFisica(fisicaRepo.c
        System.out.println("Dados incluidos com sucesso.

    /**
     * Nesta implementação, após o cadastro de
     * uma pessoa física ou jurídica,
     * perguntamos ao usuário se ele deseja
     * cadastrar outra pessoa. Se a resposta for
     * "S" (sim), o loop continuará e o usuário
     * poderá cadastrar outra pessoa. Se a
     * resposta for qualquer outra coisa que não
     * "S", o loop será interrompido e o
     * programa retornará ao menu principal.
     * Isso oferece uma experiência de usuário
     * mais fluída e intuitiva.
     */
    // Consumir a nova linha pendente
    scanner.nextLine();

    // Perguntar ao usuário se deseja cadastrar outr
    System.out.println("=====
    System.out.println("Deseja cadastrar outra pessc
    String resposta = scanner.nextLine();
    if (!(resposta.equalsIgnoreCase("S") || resposta
        break; // Sair do loop e continuar o fluxo c
    }
    } while (true); // Loop continuará até que o usuáric
}
case 2 -> {
    // Cadastro de Pessoa Jurídica
    do {
        System.out.println("=====
        System.out.println("Cadastro de Empresa");
        System.out.print("Informe o nome da empresa: ");
        String nomeEmpresa = scanner.nextLine();
        System.out.print("Informe o CNPJ: ");
        String cnpj = scanner.nextLine();
        juridicaRepo.inserir(new PessoaJuridica(juridica
        System.out.println("Dados incluidos com sucesso.

    /**
     * Nesta implementação, após o cadastro de
     * uma pessoa física ou jurídica,
     * perguntamos ao usuário se ele deseja
     * cadastrar outra pessoa. Se a resposta for
     * "S" (sim), o loop continuará e o usuário
     * poderá cadastrar outra pessoa. Se a
     * resposta for qualquer outra coisa que não
     * "S", o loop será interrompido e o
     * programa retornará ao menu principal.
     * Isso oferece uma experiência de usuário
     * mais fluída e intuitiva.
```

```

        */
        // Consumir a nova linha pendente
        //scanner.nextLine();

        // Perguntar ao usuário se deseja cadastrar outra
        System.out.println("=====");
        System.out.println("Deseja cadastrar outra empresa?");
        String resposta = scanner.nextLine();
        if (!(resposta.equalsIgnoreCase("S") || resposta.equalsIgnoreCase("N"))) {
            break; // Sair do loop e continuar o fluxo de execução
        }
    } while (true); // Loop continuará até que o usuário digite 'S' ou 'N'

}
default -> {
    System.out.println("Opcao invalida.");
    break;
}
}
}
case 2 -> {
    // Implementar alteração de pessoa
    System.out.println("=====");
    System.out.println("Escolha uma opcao a ser alterada:");
    System.out.println("1 - Pessoa Fisica");
    System.out.println("2 - Empresa");
    int tipoPessoa = scanner.nextInt();
    scanner.nextLine(); // Consumir a nova linha pendente
    switch (tipoPessoa) {
        case 1 -> {
            // Alterar Pessoa Física
            System.out.println("=====");
            System.out.print("Informe o ID da pessoa fisica a ser alterada: ");
            int idPessoa = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha pendente
            PessoaFisica pessoaFisica = fisicaRepo.obter(idPessoa);
            if (pessoaFisica != null) {
                System.out.println("Informe os novos dados:");
                System.out.print("Nome: ");
                String novoNome = scanner.nextLine();
                System.out.print("CPF: ");
                String novoCpf = scanner.nextLine();
                System.out.print("Idade: ");
                int novaIdade = scanner.nextInt();
                scanner.nextLine(); // Consumir a nova linha pendente

                pessoaFisica.setNome(novoNome);
                pessoaFisica.setCpf(novoCpf);
                pessoaFisica.setIdade(novaIdade);
                fisicaRepo.alterar(pessoaFisica);
                System.out.println("Pessoa fisica alterada com sucesso.");
            } else {

```

```

        System.out.println("Pessoa fisica com o ID forne
        break;
    }
}
case 2 -> {
    // Alterar Pessoa Jurídica
    System.out.println("=====
    System.out.print("Informe o ID da Empresa a ser alte
    int idPessoa = scanner.nextInt();
    scanner.nextLine(); // Consumir a nova linha pendent
    PessoaJuridica pessoaJuridica = juridicaRepo.obter(i
    if (pessoaJuridica != null) {
        System.out.println("Informe os novos dados:");
        System.out.print("Nome da empresa: ");
        String novoNome = scanner.nextLine();
        System.out.print("CNPJ: ");
        String novoCnpj = scanner.nextLine();

        pessoaJuridica.setNome(novoNome);
        pessoaJuridica.setCnpj(novoCnpj);
        juridicaRepo.alterar(pessoaJuridica);
        System.out.println("Empresa alterada com sucessc
    } else {
        System.out.println("Empresa com o ID fornecido r
    }
}
default -> {
    System.out.println("Opção invalida.");
    break;
}
}
case 3 -> {
    // Implementar exclusão de pessoa
    System.out.println("=====");
    System.out.println("Escolha uma opcao a ser excluida:");
    System.out.println("1 - Pessoa Fisica");
    System.out.println("2 - Empresa");
    int tipoPessoa = scanner.nextInt();
    scanner.nextLine(); // Consumir a nova linha pendente

    switch (tipoPessoa) {
        case 1 -> {
            // Excluir Pessoa Física
            System.out.println("=====
            System.out.print("Informe o ID da pessoa fisica a se
            int idPessoa = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha pendent
            PessoaFisica pessoaFisica = fisicaRepo.obter(idPesso
            if (pessoaFisica != null) {
                fisicaRepo.excluir(idPessoa);
                System.out.println("Pessoa fisica excluída com s

```

```

        } else {
            System.out.println("Pessoa fisica com o ID forne
        }
    }
    case 2 -> {
        // Excluir Pessoa Jurídica
        System.out.println("=====
        System.out.print("Informe o ID da Empresa a ser excl
        int idPessoa = scanner.nextInt();
        scanner.nextLine(); // Consumir a nova linha pendent
        PessoaJuridica pessoaJuridica = juridicaRepo.obter(i
        if (pessoaJuridica != null) {
            juridicaRepo.excluir(idPessoa);
            System.out.println("Empresa excluída com sucessc
        } else {
            System.out.println("Empresa com o ID fornecido r
        }
    }
    default -> {
        System.out.println("Opcao invalida.");
        break;
    }
}
break;
}
case 4 -> {
    // Implementar busca por ID
    // Busca de pessoa pelo ID
    System.out.println("=====");
    System.out.println("Escolha uma opcao a ser buscada:");
    System.out.println("1 - Pessoa Fisica");
    System.out.println("2 - Empresa");
    int tipoPessoa = scanner.nextInt();
    scanner.nextLine(); // Consumir a nova linha pendente

    switch (tipoPessoa) {
        case 1 -> {
            // Buscar Pessoa Física
            System.out.println("=====
            System.out.print("Informe o ID da pessoa fisica a se
            int idPessoa = scanner.nextInt();
            scanner.nextLine(); // Consumir a nova linha pendent
            PessoaFisica pessoaFisica = fisicaRepo.obter(idPesso
            if (pessoaFisica != null) {
                System.out.println("Pessoa Física encontrada:");
                pessoaFisica.exibir();
            } else {
                System.out.println("Pessoa física com o ID forne
            }
        }
        case 2 -> {
            // Buscar Pessoa Jurídica

```



```

        System.out.println("=====");
        System.out.print("Informe o ID da Empresa a ser busc
        int idPessoa = scanner.nextInt();
        scanner.nextLine(); // Consumir a nova linha pendent
        PessoaJuridica pessoaJuridica = juridicaRepo.obter(j
        if (pessoaJuridica != null) {
            System.out.println("Empresa encontrada:");
            System.out.println("=====");
            pessoaJuridica.exibir();
        } else {
            System.out.println("Empresa com o ID fornecido r
        }
    }
    default ->
        System.out.println("Opção invalida.");
}
break;
}
case 5 -> {
    // Implementar exibição de todas as pessoas / Poder escolher
    // Exibir registros de pessoas físicas ou jurídicas
    System.out.println("=====");
    System.out.println("Escolha a opcao de exibicao:");
    System.out.println("1 - Pessoa Fisica");
    System.out.println("2 - Empresa");
    System.out.println("3 - Exibir todas pessoas Fisicas e Empre
    int tipoPessoa = scanner.nextInt();
    scanner.nextLine(); // Consumir a nova linha pendente
    switch (tipoPessoa) {
        case 1 -> {
            // Exibir pessoas físicas
            System.out.println("===== Pessoas Fisicas =====");
            if (fisicaRepo.obterTodos().isEmpty()) {
                System.out.println("Nenhuma pessoa fisica cadastrada");
            } else {
                for (PessoaFisica pessoa : fisicaRepo.obterTodos()) {
                    pessoa.exibir();
                    System.out.println("-----");
                    System.out.println(); // Adiciona uma linha
                }
            }
        }
        case 2 -> {
            // Exibir pessoas jurídicas
            System.out.println("===== Empresas =====");
            if (juridicaRepo.obterTodos().isEmpty()) {
                System.out.println("Nenhuma pessoa juridica cadastrada");
            } else {
                for (PessoaJuridica empresa : juridicaRepo.obterTodos()) {
                    empresa.exibir();
                    System.out.println("-----");
                    System.out.println(); // Adiciona uma linha
                }
            }
        }
    }
}

```

```

        }
    }
}
case 3 -> {
    System.out.println("===== Pessoas Fisicas =====");
    if (fisicaRepo.obterTodos().isEmpty()) {
        System.out.println("Nenhuma pessoa fisica cadastrada");
    } else {
        for (PessoaFisica pessoa : fisicaRepo.obterTodos()) {
            pessoa.exibir();
            System.out.println("-----");
        }
    }

    System.out.println("===== Empresas =====");
    if (juridicaRepo.obterTodos().isEmpty()) {
        System.out.println("Nenhuma pessoa juridica cadastrada");
    } else {
        for (PessoaJuridica empresa : juridicaRepo.obterTodos()) {
            empresa.exibir();
            System.out.println("-----");
        }
    }
}
default ->
    System.out.println("Opção invalida.");
}
break;
}
case 6 -> {
    // Persistindo os dados
    File filePessoas = new File(arquivoPessoas);
    File fileEmpresas = new File(arquivoEmpresas);

    if (filePessoas.exists()) {
        filePessoas.delete();
    }

    if (fileEmpresas.exists()) {
        fileEmpresas.delete();
    }
    try {
        fisicaRepo.persistir(arquivoPessoas);
        juridicaRepo.persistir(arquivoEmpresas);
        System.out.println("=====");
        System.out.println("Dados persistidos com sucesso.");
    } catch (IOException e) {
        System.out.println("Erro ao persistir os dados: " + e.getMessage());
    }
}
case 7 -> {
    // Recuperando os dados

```

```

// Verificar se os arquivos existem antes de tentar recuperar
File filePessoas = new File(arquivoPessoas);
File fileEmpresas = new File(arquivoEmpresas);

/**
 * if (filePessoas.exists()) { filePessoas.delete(); }
 *
 * if (fileEmpresas.exists()) { fileEmpresas.delete(); }
 *
 */
if (!filePessoas.exists() || !fileEmpresas.exists()) {
    System.out.println("Arquivos de dados nao encontrados. C
    break;
}

try {
    fisicaRepo.recuperar(arquivoPessoas);
    juridicaRepo.recuperar(arquivoEmpresas);
    System.out.println("=====
    System.out.println("Dados recuperados com sucesso.");
} catch (IOException | ClassNotFoundException e) {
    System.out.println("Erro ao recuperar os dados: " + e.ge
}
}
case 0 -> {
    System.out.println("Finalizando o programa. Ate mais!");
    break;
}
default -> {
    System.out.println("Opcao invalida. Tente novamente.");
    break;
}
}
} while (opcao != 0);
}
}
}

```

## Análise

### • Vantagens e Desvantagens do Uso de Herança

#### ◦ Vantagens:

- Reutilização de Código: A herança permite que uma classe herde atributos e métodos de outra classe, evitando duplicação de implementações semelhantes.
- Polimorfismo: Classes derivadas podem ser tratadas como objetos da classe base, facilitando a criação de código flexível e genérico.
- Organização Hierárquica: A herança ajuda a organizar classes em uma hierarquia,

refletindo relações do mundo real.

- Desvantagens:

- Acoplamento Forte: Mudanças na classe base afetam todas as classes derivadas, criando um acoplamento forte.
- Herança Múltipla Complexa: Em linguagens que suportam herança múltipla, conflitos podem surgir quando uma classe herda de várias classes.
- Quebra de Encapsulamento: Herdar membros privados pode expor detalhes internos da classe base.

- **Necessidade da Interface Serializable em Persistência em Arquivos Binários**

- A interface Serializable é essencial para permitir que objetos sejam convertidos em bytes e salvos em arquivos binários. Isso é fundamental para a persistência de dados.

- **Uso do Paradigma Funcional pela API Stream no Java**

- A API Stream utiliza conceitos do paradigma funcional, como operações de mapeamento, filtragem e redução. Isso permite escrever código mais conciso e expressivo, melhorando a legibilidade e a manutenção.

- **Padrão de Desenvolvimento na Persistência de Dados em Arquivos no Java**

- No contexto do desenvolvimento Java, o padrão comumente adotado para persistência de dados em arquivos é usar a serialização (como a interface Serializable) ou formatos como JSON ou XML. O projeto desenvolvido atendeu aos objetivos propostos, aplicando os conceitos estudados e demonstrando a capacidade de criar um sistema cadastral em Java. O uso de herança, polimorfismo e persistência em arquivos binários contribuiu para a solução eficiente e organizada. O controle de exceções garantiu a robustez do sistema.

## Relatório Final: Sistema de Cadastro de Pessoas

---

- **Objetivo do Projeto:** O objetivo deste projeto é desenvolver um sistema de cadastro de pessoas que possa armazenar informações tanto de pessoas físicas quanto jurídicas. O sistema oferece funcionalidades para inclusão, alteração, exclusão, busca e exibição de todas as pessoas cadastradas, além de permitir a persistência dos dados em arquivos para futuras consultas.

## Resumo das Opções do Sistema:

---

- **Incluir Pessoa:**

- Permite ao usuário cadastrar uma nova pessoa, podendo ser física ou jurídica.
- Solicita os dados necessários para o cadastro, como nome, CPF (para pessoas físicas) ou CNPJ (para pessoas jurídicas) e idade (para pessoas físicas).
- Após o cadastro, oferece a opção de cadastrar outra pessoa ou retornar ao menu

principal.

- **Alterar Pessoa:**

- Permite ao usuário alterar os dados de uma pessoa já cadastrada.
- Solicita o ID da pessoa que se deseja alterar e, em seguida, permite a atualização dos dados conforme necessário.

- **Excluir Pessoa:**

- Permite ao usuário excluir uma pessoa cadastrada com base no seu ID.
- Solicita o ID da pessoa a ser excluída e realiza a exclusão do registro correspondente.

- **Buscar Pelo ID:**

- Permite ao usuário buscar e exibir os dados de uma pessoa específica com base no seu ID.
- Solicita o ID da pessoa desejada e exibe todas as informações associadas a ela.

- **Exibir Todos:**

- Oferece três opções:
  - Exibir todas as pessoas físicas cadastradas.
  - Exibir todas as pessoas jurídicas cadastradas.
  - Exibir todas as pessoas, tanto físicas quanto jurídicas, de forma separada para melhor visualização.
- Exibe os dados de todas as pessoas cadastradas de acordo com a opção escolhida.

- **Persistir Dados:**

- Permite ao usuário salvar os dados cadastrados em um arquivo para futuras consultas.
- Os dados são armazenados de forma que possam ser recuperados posteriormente mesmo após o encerramento do programa.

- **Recuperar Dados:**

- Permite ao usuário recuperar os dados previamente salvos em um arquivo.
- Ao iniciar o programa, verifica a existência dos arquivos de dados e, se encontrados, os recupera para uso no sistema.

- **Finalizar Programa:**

- Encerra a execução do programa. Se precisar de mais informações ou tiver outras dúvidas, estou à disposição! 😊

