Campus: Asa Norte

Curso: Desenvolvimento Full Stack

Disciplina: Iniciando o caminho pelo Java

Matrícula: 2023.09.96862-2

Semestre Letivo: 3º Semestre

Integrantes: André Luis Soares de Oliveira

Sistema de Cadastro com Persistência de Dados em Java

Objetivo da Prática

Desenvolver um sistema de cadastro utilizando Java com persistência de dados em arquivos binários, aplicando conceitos de herança, interfaces e repositórios para armazenar e recuperar informações de pessoas físicas e jurídicas.

Códigos Solicitados

Classe Main (para cadastro em modo texto):

```
import java.util.Scanner;

import model.PessoaFisica;

import model.PessoaFisicaRepo;

import model.PessoaJuridica;

import model.PessoaJuridicaRepo;


public class CadastroPOO {

   public static void main(String[] args) {

      PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();

      PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

      Scanner scanner = new Scanner(System.in);

      int opcao;


      do {

         // Exibir menu formatado

         System.out.println("\n======= Sistema de Cadastro =======");

         System.out.println("1 - Incluir nova pessoa");

         System.out.println("2 - Alterar dados de pessoa");

         System.out.println("3 - Excluir pessoa");

         System.out.println("4 - Exibir pessoa por ID");
```

```
System.out.println("5 - Exibir todas as pessoas");
System.out.println("6 - Salvar dados");
System.out.println("7 - Recuperar dados");
System.out.println("0 - Sair");
System.out.println("==================================\n");
System.out.print("Escolha uma opção: ");
opcao = scanner.nextInt();
scanner.nextLine(); // Limpar o buffer

switch (opcao) {
   case 1: // Incluir
      System.out.println("\n*** Inclusão de nova pessoa ***");
      System.out.print("Escolha o tipo (1 - Física, 2 - Jurídica): ");
      int tipo = scanner.nextInt();
      scanner.nextLine(); // Limpar o buffer

      if (tipo == 1) { // Pessoa Física
         System.out.print("Digite o ID: ");
         int id = scanner.nextInt();
         scanner.nextLine();
         System.out.print("Digite o nome: ");
         String nome = scanner.nextLine();
         System.out.print("Digite o CPF: ");
         String cpf = scanner.nextLine();
         System.out.print("Digite a idade: ");
         int idade = scanner.nextInt();
```

```
            scanner.nextLine();

            PessoaFisica pessoaFisica = new PessoaFisica(id, nome, cpf, idade);
            repoFisica.inserir(pessoaFisica);
            System.out.println("Pessoa física adicionada com sucesso!");

        } else if (tipo == 2) { // Pessoa Jurídica
            System.out.print("Digite o ID: ");
            int id = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Digite o nome: ");
            String nome = scanner.nextLine();
            System.out.print("Digite o CNPJ: ");
            String cnpj = scanner.nextLine();

            PessoaJuridica pessoaJuridica = new PessoaJuridica(id, nome, cnpj);
            repoJuridica.inserir(pessoaJuridica);
            System.out.println("Pessoa jurídica adicionada com sucesso!");
        }
        break;

    case 2: // Alterar
        System.out.println("\n*** Alteração de dados de pessoa ***");
        System.out.print("Escolha o tipo (1 - Física, 2 - Jurídica): ");
        tipo = scanner.nextInt();
        scanner.nextLine(); // Limpar o buffer
```

```
if (tipo == 1) { // Pessoa Física
    System.out.print("Digite o ID: ");
    int id = scanner.nextInt();
    scanner.nextLine();
    PessoaFisica pessoaFisica = repoFisica.obter(id);
    if (pessoaFisica != null) {
        System.out.println("Dados atuais: " + pessoaFisica);
        System.out.print("Digite o novo nome: ");
        String nome = scanner.nextLine();
        System.out.print("Digite o novo CPF: ");
        String cpf = scanner.nextLine();
        System.out.print("Digite a nova idade: ");
        int idade = scanner.nextInt();
        scanner.nextLine();

        pessoaFisica.setNome(nome);
        pessoaFisica.setCpf(cpf);
        pessoaFisica.setIdade(idade);
        repoFisica.alterar(pessoaFisica);
        System.out.println("Pessoa física alterada com sucesso!");
    } else {
        System.out.println("Pessoa física não encontrada.");
    }
} else if (tipo == 2) { // Pessoa Jurídica
    System.out.print("Digite o ID: ");
```

```
            int id = scanner.nextInt();
            scanner.nextLine();
            PessoaJuridica pessoaJuridica = repoJuridica.obter(id);
            if (pessoaJuridica != null) {
                System.out.println("Dados atuais: " + pessoaJuridica);
                System.out.print("Digite o novo nome: ");
                String nome = scanner.nextLine();
                System.out.print("Digite o novo CNPJ: ");
                String cnpj = scanner.nextLine();

                pessoaJuridica.setNome(nome);
                pessoaJuridica.setCnpj(cnpj);
                repoJuridica.alterar(pessoaJuridica);
                System.out.println("Pessoa jurídica alterada com sucesso!");
            } else {
                System.out.println("Pessoa jurídica não encontrada.");
            }
        }
        break;

    case 3: // Excluir
        System.out.println("\n*** Exclusão de pessoa ***");
        System.out.print("Escolha o tipo (1 - Física, 2 - Jurídica): ");
        tipo = scanner.nextInt();
        scanner.nextLine(); // Limpar o buffer
```

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

        if (tipo == 1) { // Pessoa Física
            repoFisica.excluir(id);
            System.out.println("Pessoa física excluída com sucesso!");
        } else if (tipo == 2) { // Pessoa Jurídica
            repoJuridica.excluir(id);
            System.out.println("Pessoa jurídica excluída com sucesso!");
        }
        break;

    case 4: // Exibir por ID
        System.out.println("\n*** Exibir pessoa por ID ***");
        System.out.print("Escolha o tipo (1 - Física, 2 - Jurídica): ");
        tipo = scanner.nextInt();
        scanner.nextLine(); // Limpar o buffer

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

        if (tipo == 1) { // Pessoa Física
            PessoaFisica pessoaFisica = repoFisica.obter(id);
            if (pessoaFisica != null) {
```

```
            System.out.println(pessoaFisica);
        } else {
            System.out.println("Pessoa física não encontrada.");
        }
    } else if (tipo == 2) { // Pessoa Jurídica
        PessoaJuridica pessoaJuridica = repoJuridica.obter(id);
        if (pessoaJuridica != null) {
            System.out.println(pessoaJuridica);
        } else {
            System.out.println("Pessoa jurídica não encontrada.");
        }
    }
    break;

case 5: // Exibir todos
    System.out.println("\n*** Exibir todas as pessoas ***");
    System.out.print("Escolha o tipo (1 - Física, 2 - Jurídica): ");
    tipo = scanner.nextInt();
    scanner.nextLine(); // Limpar o buffer

    if (tipo == 1) { // Pessoa Física
        for (PessoaFisica p : repoFisica.obterTodos()) {
            System.out.println(p);
        }
    } else if (tipo == 2) { // Pessoa Jurídica
        for (PessoaJuridica p : repoJuridica.obterTodos()) {
```

```
            System.out.println(p);
        }
    }
    break;

case 6: // Salvar dados
    System.out.println("\n*** Salvando dados ***");
    System.out.print("Digite o prefixo do arquivo: ");
    String prefixo = scanner.nextLine();

    try {
        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 dados: " + e.getMessage());
    }
    break;

case 7: // Recuperar dados
    System.out.println("\n*** Recuperando dados ***");
    System.out.print("Digite o prefixo do arquivo: ");
    prefixo = scanner.nextLine();

    try {
        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 dados: " + e.getMessage());
            }
            break;

        case 0: // Sair
            System.out.println("\nFinalizando o programa. Até logo!");
            break;

        default:
            System.out.println("Opção inválida. Tente novamente.");
            break;
      }

    } while (opcao != 0);

    scanner.close();
  }
}
```

Classe Main:

```
package cadastropoo;

import model.PessoaFisica;
```

```
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

/**
 * Classe principal para testar o sistema de persistência de pessoas físicas e jurídicas.
 */
public class CadastroPOO {

    /**
     * Método principal que executa os testes de persistência e recuperação de dados.
     *
     * @param args argumentos da linha de comando (não utilizados)
     */
    public static void main(String[] args) {
        try {
            // Repositório de pessoas físicas
            System.out.println("Criando repositorio de pessoas fisicas (repo1) e adicionando pessoas...");
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            repo1.inserir(new PessoaFisica(1, "Joao", "123.456.789-00", 30));
            repo1.inserir(new PessoaFisica(2, "Maria", "987.654.321-00", 25));
            repo1.persistir("pessoasFisicas.dat");
            System.out.println("Dados de pessoas fisicas foram armazenados com sucesso em 'pessoasFisicas.dat'.");

            System.out.println("\nRecuperando dados de pessoas fisicas (repo2)...");
```

```
        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        repo2.recuperar("pessoasFisicas.dat");
        System.out.println("Pessoas fisicas recuperadas:");
        for (PessoaFisica pf : repo2.obterTodos()) {
            pf.exibir();
        }
        System.out.println("Dados de pessoas fisicas recuperados com sucesso.");

        // Repositório de pessoas jurídicas
        System.out.println("\nCriando repositorio de pessoas juridicas (repo3) e adicionando pessoas...");
        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        repo3.inserir(new PessoaJuridica(1, "Empresa X", "12.345.678/0001-00"));
        repo3.inserir(new PessoaJuridica(2, "Empresa Y", "98.765.432/0001-00"));
        repo3.persistir("pessoasJuridicas.dat");
        System.out.println("Dados de pessoas juridicas foram armazenados com sucesso em 'pessoasJuridicas.dat'.");

        System.out.println("\nRecuperando dados de pessoas juridicas (repo4)...");
        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        repo4.recuperar("pessoasJuridicas.dat");
        System.out.println("Pessoas juridicas recuperadas:");
        for (PessoaJuridica pj : repo4.obterTodos()) {
            pj.exibir();
        }
        System.out.println("Dados de pessoas juridicas recuperados com sucesso.");
```

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

Classe Pessoa:

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

Classe PessoaFisica:

```
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;
    }

    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;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf + ", Idade: " + idade);
    }
}
```

Classe PessoaFisicaRepo:

```
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(int id, PessoaFisica novaPessoa) {
      PessoaFisica pessoa = obter(id);
      if (pessoa != null) {
         pessoa.setNome(novaPessoa.getNome());
         pessoa.setCpf(novaPessoa.getCpf());
         pessoa.setIdade(novaPessoa.getIdade());
      }
   }


   public void excluir(int id) {
      PessoaFisica pessoa = obter(id);
      if (pessoa != null) {
         pessoasFisicas.remove(pessoa);
```

```
    }
  }

  public PessoaFisica obter(int id) {
    return pessoasFisicas.stream()
        .filter(p -> p.getId() == id)
        .findFirst()
        .orElse(null);
  }

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

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

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

```
}
```

Classe PessoaJuridica:

```
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;
   }


   public String getCnpj() {
      return cnpj;
   }


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


   @Override
   public void exibir() {
      super.exibir();
      System.out.println("CNPJ: " + cnpj);
```

```
    }
}
```

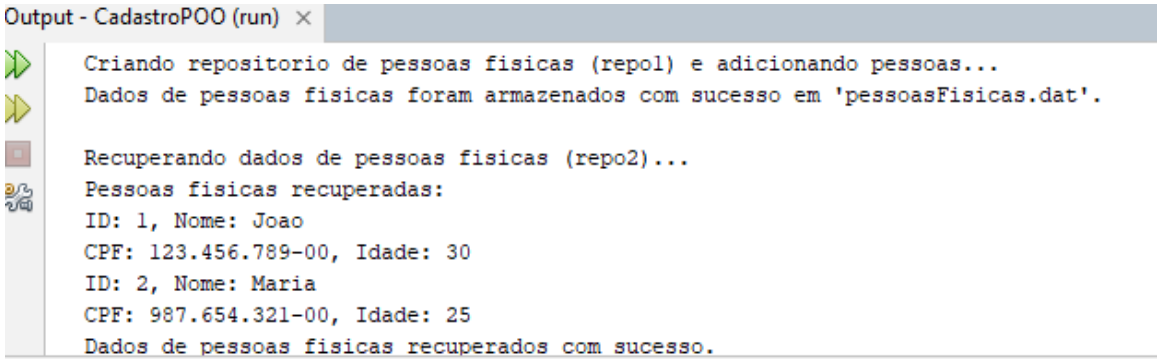
Classe PessoaJuridicaRepo:

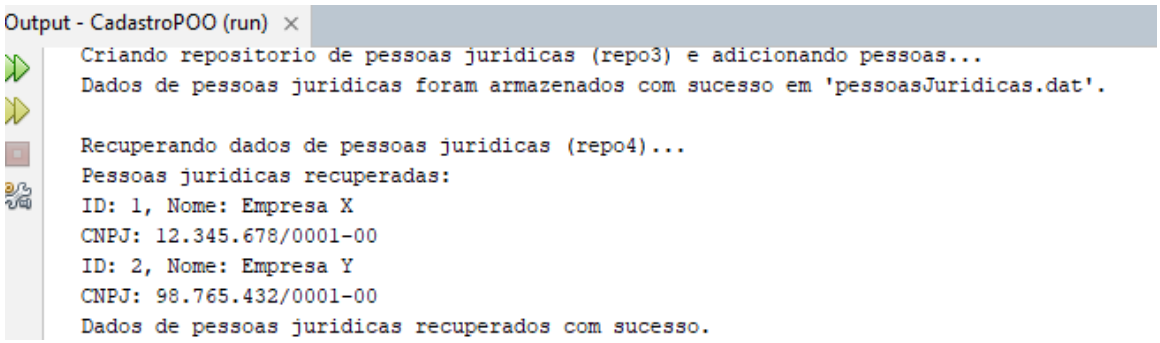
```
package model;

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(int id, PessoaJuridica novaPessoa) {
        PessoaJuridica pessoa = obter(id);
        if (pessoa != null) {
            pessoa.setNome(novaPessoa.getNome());
            pessoa.setCnpj(novaPessoa.getCnpj());
        }
    }

    public void excluir(int id) {
        PessoaJuridica pessoa = obter(id);
        if (pessoa != null) {
```

```
            pessoasJuridicas.remove(pessoa);
        }
    }

    public PessoaJuridica obter(int id) {
        return pessoasJuridicas.stream()
                .filter(p -> p.getId() == id)
                .findFirst()
                .orElse(null);
    }

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

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

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

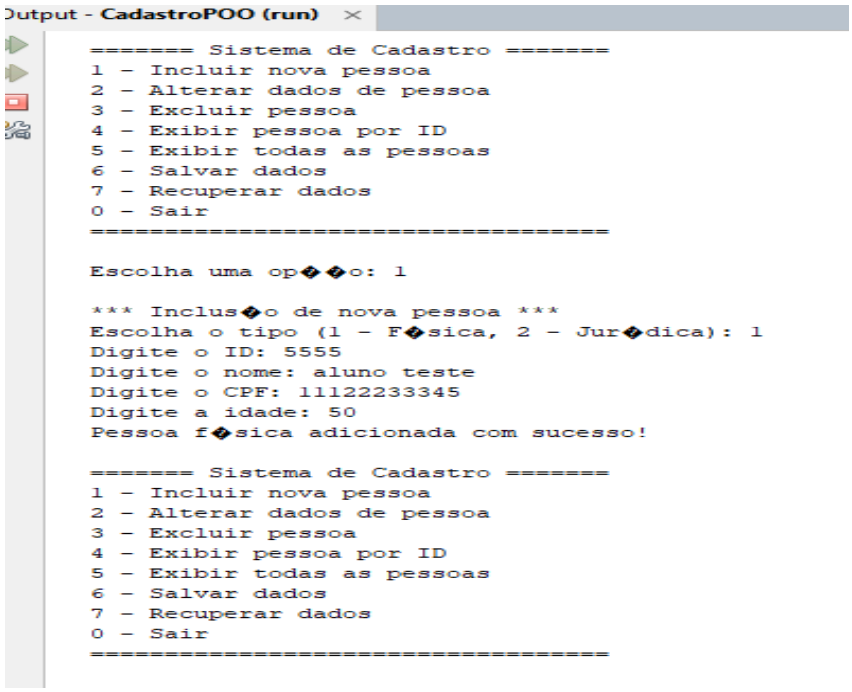
```
    }
}
```

Resultados da Execução





Cadastro em modo texto:



Análise e Conclusão

Vantagens e Desvantagens do Uso de Herança

Vantagens:

- - Reutilização de código: A herança permite a reutilização de funcionalidades comuns entre classes relacionadas.
- - Simplificação: Facilita a criação de classes mais especializadas sem duplicação de código.

Desvantagens:

- - Acoplamento: Pode criar um acoplamento mais forte entre classes, dificultando mudanças futuras.
- - Complexidade: A árvore de herança pode tornar o código mais difícil de manter e entender.

Por que a Interface Serializable é necessária para Persistência em Arquivos Binários?

A interface Serializable é essencial para que os objetos possam ser convertidos em um fluxo de bytes e assim armazenados em arquivos binários. Sem ela, o Java não sabe como serializar os objetos adequadamente, e a tentativa de persistência em arquivos falharia.

Como o Paradigma Funcional é Utilizado pela API Stream no Java?

A API Stream no Java segue o paradigma funcional ao permitir a aplicação de operações como mapeamento, filtragem e redução de dados em uma sequência de elementos, de maneira declarativa. Ela permite que você processe coleções com funções como map(), filter() e reduce(), promovendo um estilo de programação mais limpo e conciso.

Padrão de Desenvolvimento Adotado na Persistência de Dados em Arquivos no Java

O padrão mais utilizado para persistência de dados em arquivos no Java é o Data Access Object (DAO). Ele encapsula o acesso aos dados, fornecendo uma interface simples para persistir e recuperar informações, independentemente dos detalhes de armazenamento, como banco de dados ou arquivos binários.

Endereço do Repositório GIT

https://github.com/andreluissdo/Miss-o-Pr-tica---Iniciando-o-caminho-pelo-Java.git