Campus: Polo Méier
Curso: Desenvolvimento Full Stack
Disciplina: Introdução à Programação OO em Java
Número da Turma: EAD
Semestre Letivo: 2024.4
Nome do Integrante da Prática: Gabriel dos Reis Ferreira

Cadastro de Pessoas Físicas e Jurídicas com Persistência e Recuperação de Dados

Objetivo da Prática

Desenvolver um sistema de cadastro para pessoas físicas e jurídicas, que permita realizar operações de inclusão, alteração, exclusão, exibição e recuperação de dados, com persistência em arquivos. A prática também teve como objetivo reforçar conceitos de orientação a objetos, como herança, polimorfismo e encapsulamento, além do uso de classes repositório para organização do código e da manipulação de arquivos em Java.

Códigos Solicitados

Os códigos solicitados incluem a implementação de classes para representar pessoas físicas e jurídicas, repositórios para gerenciar dados, e funcionalidades de persistência e recuperação de dados. Abaixo estão os trechos de código implementados.

Código da 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);
}
}
```

Código da 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);
}
}
```

Código da Classe PessoaJuridica

```
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;
}
public String getCnpj() {
return cnpj;
}
public void setCnpj(String cnpj) {
this.cnpj = cnpj;
}
@Override
public void exibir() {
System.out.println("Pessoa Jurídica:");
super.exibir();
System.out.println("CNPJ: " + cnpj);
}
}
```

Código da Classe PessoaFisicaRepo

```
package model;
import java.io.*;
import java.util.ArrayList;
public class PessoaFisicaRepo {
private ArrayList<PessoaFisica> lista = new ArrayList<>();
// Método inserir
public void inserir(PessoaFisica pessoaFisica) {
lista.add(pessoaFisica);
```

```
}
// Método alterar
public void alterar(PessoaFisica pessoaFisica) {
for (int i = 0; i < lista.size(); i++) {
if (lista.get(i).getId() == pessoaFisica.getId()) {
lista.set(i, pessoaFisica);
return;
}
}
}
// Método excluir
public void excluir(int id) {
lista.removeIf(pessoa -> pessoa.getId() == id);
}
// Método obter
public PessoaFisica obter(int id) {
for (PessoaFisica pessoa : lista) {
if (pessoa.getId() == id) {
return pessoa;
}
}
return null;
}
// Método obterTodos
public ArrayList<PessoaFisica> obterTodos() {
return new ArrayList<>(lista);
}
```

```
// Método persistir (salvar no disco)

public void persistir(String nomeArquivo) throws IOException {

try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {

oos.writeObject(lista);

}

}

// Método recuperar (carregar do disco)

public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {

lista = (ArrayList<PessoaFisica>) ois.readObject();

}

}

}
```

Código da Classe PessoaJuridicaRepo

```
package model;

import java.io.*;

import java.util.ArrayList;

public class PessoaJuridicaRepo {

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

// Método inserir

public void inserir(PessoaJuridica pessoaJuridica) {

lista.add(pessoaJuridica);

}

// Método alterar

public void alterar(PessoaJuridica pessoaJuridica) {

for (int i = 0; i < lista.size(); i++) {

if (lista.get(i).getId() == pessoaJuridica.getId()) {
```

```
lista.set(i, pessoaJuridica);
return;
}
}
}
// Método excluir
public void excluir(int id) {
lista.removeIf(pessoa -> pessoa.getId() == id);
}
// Método obter
public PessoaJuridica obter(int id) {
for (PessoaJuridica pessoa : lista) {
if (pessoa.getId() == id) {
return pessoa;
}
}
return null;
}
// Método obterTodos
public ArrayList<PessoaJuridica> obterTodos() {
return new ArrayList<>(lista);
}
// Método persistir (salvar no disco)
public void persistir(String nomeArquivo) throws IOException {
try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
oos.writeObject(lista);
}
```

```
}

// Método recuperar (carregar do disco)
public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
lista = (ArrayList<PessoaJuridica>) ois.readObject();
}
}
}
```

Código da Classe Main

```
package model;
import java.io.IOException;
import java.util.Scanner;
public class Main {
public static void main(String[] args) {
try {
// Configuração para corrigir problemas de codificação no console
System.setOut(new java.io.PrintStream(System.out, true, "UTF-8"));
PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();
Scanner scanner = new Scanner(System.in);
while (true) {
System.out.println("\n=== Cadastro de Pessoas ===");
System.out.println("1. Incluir");
System.out.println("2. Alterar");
System.out.println("3. Excluir");
System.out.println("4. Exibir pelo ID");
System.out.println("5. Exibir todos");
```

```
System.out.println("6. Salvar dados");
System.out.println("7. Recuperar dados");
System.out.println("0. Sair");
System.out.print("Selecione uma opção: ");
int opcao = scanner.nextInt();
scanner.nextLine(); // Limpa o buffer
switch (opcao) {
case 1 -> {
System.out.println("1. Pessoa Física");
System.out.println("2. Pessoa Jurídica");
System.out.print("Escolha o tipo: ");
int tipo = scanner.nextInt();
scanner.nextLine(); // Limpa o buffer
if (tipo == 1) {
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();
pessoaFisicaRepo.inserir(new PessoaFisica(id, nome, cpf, idade));
} else if (tipo == 2) {
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();
pessoaJuridicaRepo.inserir(new PessoaJuridica(id, nome, cnpj));
} else {
System.out.println("Opção inválida!");
}
}
case 2 -> {
// Similar à inclusão, mas altera o objeto no repositório.
System.out.println("Funcionalidade de alteração ainda não implementada.");
}
case 3 -> {
System.out.print("ID da pessoa a excluir: ");
int id = scanner.nextInt();
scanner.nextLine();
pessoaFisicaRepo.excluir(id);
pessoaJuridicaRepo.excluir(id);
}
case 4 -> {
System.out.print("ID da pessoa a exibir: ");
int id = scanner.nextInt();
scanner.nextLine();
PessoaFisica pf = pessoaFisicaRepo.obter(id);
```

```
if (pf != null) {
pf.exibir();
} else {
PessoaJuridica pj = pessoaJuridicaRepo.obter(id);
if (pj != null) {
pj.exibir();
} else {
System.out.println("Pessoa não encontrada.");
}
}
}
case 5 -> {
System.out.println("\nPessoas Físicas:");
for (PessoaFisica pf : pessoaFisicaRepo.obterTodos()) {
pf.exibir();
}
System.out.println("\nPessoas Jurídicas:");
for (PessoaJuridica pj : pessoaJuridicaRepo.obterTodos()) {
pj.exibir();
}
}
case 6 -> {
String arquivoPF = "pessoas_fisicas.dat";
String arquivoPJ = "pessoas_juridicas.dat";
pessoaFisicaRepo.persistir(arquivoPF);
pessoaJuridicaRepo.persistir(arquivoPJ);
System.out.println("Dados salvos com sucesso.");
```

```
}
case 7 -> {
String arquivoPF = "pessoas_fisicas.dat";
String arquivoPJ = "pessoas_juridicas.dat";
pessoaFisicaRepo.recuperar(arquivoPF);
pessoaJuridicaRepo.recuperar(arquivoPJ);
System.out.println("Dados recuperados com sucesso.");
}
case 0 -> {
System.out.println("Encerrando o programa...");
return;
}
default -> System.out.println("Opção inválida!");
}
}
} catch (IOException | ClassNotFoundException e) {
System.err.println("Erro: " + e.getMessage());
}
}
}
```

Resultados da Execução

Após a execução dos códigos, o sistema apresenta o seguinte resultado no console:

=== Cadastro de Pessoas ===

Pessoas Físicas recuperadas:

ID: 1, Nome: João Silva

CPF: 123.456.789-00, Idade: 30

ID: 2, Nome: Maria Oliveira

CPF: 987.654.321-00, Idade: 25

Pessoas Jurídicas recuperadas:

ID: 1, Nome: Minalba

CNPJ: 12.345.678/0001-90

ID: 2, Nome: Natura

CNPJ: 98.765.432/0001-11

Análise e Conclusão

• **O que são elementos estáticos e qual é a razão para o método main usar esse modificador?** Elementos estáticos são aqueles que são de nível de classe e não de nível de instância. O método main é estático porque precisa ser chamado sem criar um objeto da classe. O método main é o ponto de entrada do programa e, como é estático, a JVM pode chamá-lo de forma direta para iniciar o programa.

• **Para que serve a classe Scanner?** A classe Scanner em Java é usada para ler dados de uma variedade de fontes de entrada, como teclado, arquivos e strings. É amplamente utilizada para capturar dados de entrada do usuário, especialmente em casos onde o sistema é interativo.

• **Qual é a influência do uso de classes de repositório na estrutura geral do código?** O uso de classes de repositório possibilitou estruturar o código de forma melhor, permitindo isolar as operações de armazenamento e recuperação de dados em um único local. Isso aprimora a manutenção do programa e melhora a arquitetura do programa, uma vez que torna possível ter preocupações separadas de manipulação de dados e a lógica de codificação real do programa.

O código-fonte do projeto está disponível no repositório Git:
https://github.com/GabrielR3isS/GabrielR3isS-Miss-o-Pr-tica-N-vel-1-Mundo-3.git