Estácio

POLO PRINC. IZABEL - CACHOEIRINHA - RS

DESENVOLVIMENTO FULL STACK

RPG0014 -Iniciando o caminho pelo Java

Turma 9001

Primeiro semestre de 2024

Integrantes: Fagner Luiz Gimenez Mendes.

2º Procedimento | Criação do Cadastro em Modo Texto

Objetivos 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.

Todos os códigos solicitados neste roteiro de aula;

```
package model;

import java.io.Serializable;


public class Pessoa implements Serializable{

   private static final long serialVersionUID = 1L;
   int id;
   String nome;

   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;

    }
}

package model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable{

    private static final long serialVersionUID = 1L;
    String cpf;
    int idade;

    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 +"\n");
        }

    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;
        }
}


package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable{

   private static final long serialVersionUID = 1L;
   String cnpj;

   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+"\n");
        }

   public String getcnpj() {
        return cnpj;
        }

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


package model;

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

public class PessoaFisicaRepo implements Serializable {

   private static final long serialVersionUID = 1L;
   private List<PessoaFisica> pessoasFisicas = new ArrayList<>();
```

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

        public void alterar(PessoaFisica pessoa) {
        int index = obterIndexPorId(pessoa.getId());
        if (index != -1) {
        pessoasFisicas.set(index, pessoa);
        }
        }

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

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

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

        public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
        outputStream.writeObject(this);
        }
        }

        public static PessoaFisicaRepo recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
        return (PessoaFisicaRepo) inputStream.readObject();
        }
        }

        private int obterIndexPorId(int id) {
```

```
        for (int i = 0; i < pessoasFisicas.size(); i++) {
        if (pessoasFisicas.get(i).getId() == id) {
        return i;
        }
        }
        return -1;
        }
}


package model;

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

public class PessoaJuridicaRepo implements Serializable {


   private static final long serialVersionUID = 1L;
   private List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

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

        public void alterar(PessoaJuridica pessoa) {
        int index = obterIndexPorId(pessoa.getId());
        if (index != -1) {
        pessoasJuridicas.set(index, pessoa);
        }
        }

        public void excluir(int id) {
        PessoaJuridica pessoa = obter(id);
        if (pessoa != null) {
        pessoasJuridicas.remove(pessoa);
        }
        }

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

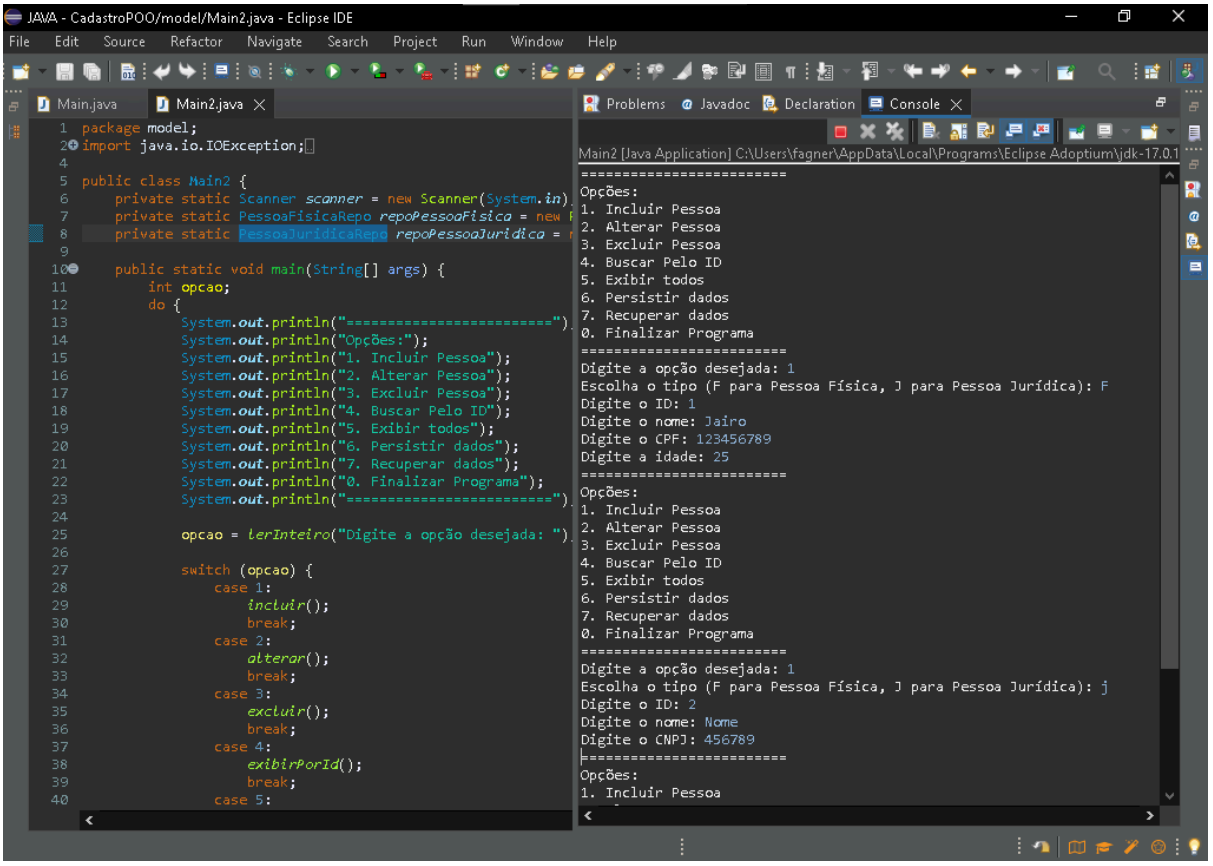
```
	public List<PessoaJuridica> obterTodos() {
	return new ArrayList<>(pessoasJuridicas);
	}

	public void persistir(String nomeArquivo) throws IOException {
	try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
	outputStream.writeObject(this);
	}
	}

	public static PessoaJuridicaRepo recuperar(String nomeArquivo) throws IOException,
ClassNotFoundException {
	try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
	return (PessoaJuridicaRepo) inputStream.readObject();
	}
	}

	private int obterIndexPorId(int id) {
	for (int i = 0; i < pessoasJuridicas.size(); i++) {
	if (pessoasJuridicas.get(i).getId() == id) {
	return i;
	}
	}
	return -1;
	}
}

package model;
import java.io.IOException;
import java.util.Scanner;

public class Main2 {
	private static Scanner scanner = new Scanner(System.in);
	private static PessoaFisicaRepo repoPessoaFisica = new PessoaFisicaRepo();
	private static PessoaJuridicaRepo repoPessoaJuridica = new PessoaJuridicaRepo();

	public static void main(String[] args) {
	int opcao;
	do {
		System.out.println("==========================");
	System.out.println("Opções:");
	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("========================");

opcao = lerInteiro("Digite a opção desejada: ");

switch (opcao) {
case 1:
        incluir();
        break;
case 2:
        alterar();
        break;
case 3:
        excluir();
        break;
case 4:
        exibirPorId();
        break;
case 5:
        exibirTodos();
        break;
case 6:
        salvarDados();
        break;
case 7:
        recuperarDados();
        break;
case 0:
        System.out.println("Finalizando o programa.");
        break;
default:
        System.out.println("Opção inválida. Tente novamente.");
}
} while (opcao != 0);
}

private static void incluir() {
String tipo = lerString("Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): ");
if (tipo.equalsIgnoreCase("F")) {
incluirPessoaFisica();
} else if (tipo.equalsIgnoreCase("J")) {
incluirPessoaJuridica();
} else {
System.out.println("Tipo inválido.");
}
```

```
}

private static void incluirPessoaFisica() {
int id = lerInteiro("Digite o ID: ");
String nome = lerString("Digite o nome: ");
String cpf = lerString("Digite o CPF: ");
int idade = lerInteiro("Digite a idade: ");
repoPessoaFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
}

private static void incluirPessoaJuridica() {
int id = lerInteiro("Digite o ID: ");
String nome = lerString("Digite o nome: ");
String cnpj = lerString("Digite o CNPJ: ");
repoPessoaJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
}

private static void alterar() {
 String tipo = lerString("Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): ");
int id = lerInteiro("Digite o ID: ");
if (tipo.equalsIgnoreCase("F")){
PessoaFisica pessoa = repoPessoaFisica.obter(id);
if (pessoa != null) {
System.out.println("Dados atuais da pessoa física:");
pessoa.exibir();
System.out.println("Digite os novos dados:");
incluirPessoaFisica();
} else {
System.out.println("Pessoa física não encontrada.");
}
} else if (tipo.equalsIgnoreCase("J")){
PessoaJuridica pessoa = repoPessoaJuridica.obter(id);
if (pessoa != null) {
System.out.println("Dados atuais da pessoa jurídica:");
pessoa.exibir();
System.out.println("Digite os novos dados:");
incluirPessoaJuridica();
} else {
System.out.println("Pessoa jurídica não encontrada.");
}
} else {
System.out.println("Tipo inválido.");
}
}

private static void excluir() {
 String tipo = lerString("Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): ");
int id = lerInteiro("Digite o ID: ");
```

```
if (tipo.equalsIgnoreCase("F")) {
repoPessoaFisica.excluir(id);
} else if (tipo.equalsIgnoreCase("J")){
repoPessoaJuridica.excluir(id);
} else {
System.out.println("Tipo inválido.");
}
}

private static void exibirPorId() {
 String tipo = lerString("Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): ");
int id = lerInteiro("Digite o ID: ");
if (tipo.equalsIgnoreCase("F")) {
PessoaFisica pessoa = repoPessoaFisica.obter(id);
if (pessoa != null) {
pessoa.exibir();
} else {
System.out.println("Pessoa física não encontrada.");
}
} else if (tipo.equalsIgnoreCase("J")){
PessoaJuridica pessoa = repoPessoaJuridica.obter(id);
if (pessoa != null) {
pessoa.exibir();
} else {
System.out.println("Pessoa jurídica não encontrada.");
}
} else {
System.out.println("Tipo inválido.");
}
}

private static void exibirTodos() {
 String tipo = lerString("Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): ");
if (tipo.equalsIgnoreCase("F")) {
for (PessoaFisica pessoa : repoPessoaFisica.obterTodos()) {
pessoa.exibir();
System.out.println();
}
} else if (tipo.equalsIgnoreCase("J")){
for (PessoaJuridica pessoa : repoPessoaJuridica.obterTodos()) {
pessoa.exibir();
System.out.println();
}
} else {
System.out.println("Tipo inválido.");
}
}
```

```
        private static void salvarDados() {
        String prefixo = lerString("Digite o prefixo dos arquivos: ");
        try {
        repoPessoaFisica.persistir(prefixo + ".fisica.bin");
        repoPessoaJuridica.persistir(prefixo + ".juridica.bin");
        System.out.println("Dados salvos com sucesso.");
        } catch (IOException e) {
        System.out.println("Erro ao salvar dados: " + e.getMessage());
        }
        }

        private static void recuperarDados() {
        String prefixo = lerString("Digite o prefixo dos arquivos: ");
        try {
        repoPessoaFisica = PessoaFisicaRepo.recuperar(prefixo + ".fisica.bin");
        repoPessoaJuridica = PessoaJuridicaRepo.recuperar(prefixo + ".juridica.bin");
        System.out.println("Dados recuperados com sucesso.");
        } catch (IOException | ClassNotFoundException e) {
        System.out.println("Erro ao recuperar dados: " + e.getMessage());
        }
        }

        private static int lerInteiro(String mensagem) {
        System.out.print(mensagem);
        return scanner.nextInt();
        }

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


JAVA - CadastroPOO/model/Main2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Main.java
Main2.java
package model;
import java.io.IOException;
public class Main2 {
private static Scanner scanner = new Scanner(System.in);
private static PessoaFisicaRepo repoPessoaFisica = new
private static PessoaJuridicaRepo repoPessoaJuridica =
public static void main(String[] args) {
int opcao;
do {
System.out.println("========================");
System.out.println("Opções:");
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("========================");
opcao = lerInteiro("Digite a opção desejada: ");
switch (opcao) {
case 1:
incluir();
break;
case 2:
alterar();
break;
case 3:
excluir();
break;
case 4:
exibirPorId();
break;
case 5:
Problems Javadoc Declaration Console
Main2 [Java Application] C:\Users\fagner\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.1
========================
Opções:
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
========================
Digite a opção desejada: 1
Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): F
Digite o ID: 1
Digite o nome: Jairo
Digite o CPF: 123456789
Digite a idade: 25
========================
Opções:
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
========================
Digite a opção desejada: 1
Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): j
Digite o ID: 2
Digite o nome: Nome
Digite o CNPJ: 456789
========================
Opções:
1. Incluir Pessoa

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

Elementos Estáticos:

Elementos estáticos em Java são membros de uma classe que pertencem à classe em si, em vez de pertencerem a instâncias individuais da classe. Isso significa que eles são compartilhados por todas as instâncias da classe. Elementos estáticos incluem variáveis estáticas (ou campos estáticos) e métodos estáticos.

- **Variáveis Estáticas:** São variáveis que pertencem à classe, em vez de pertencerem a qualquer instância específica dessa classe. Todas as instâncias da classe compartilham o mesmo valor de uma variável estática.
- **Métodos Estáticos:** São métodos que pertencem à classe em vez de pertencerem a qualquer instância específica da classe. Eles podem ser chamados sem criar uma instância da classe.

Método main e o Modificador Estático:

O método main é o ponto de entrada para qualquer aplicação Java. Ele é definido como estático porque quando a JVM (Java Virtual Machine) inicia a execução do programa, ela não cria uma instância da classe, mas sim chama o método main diretamente.

Se o método main não fosse estático, a JVM teria que criar uma instância da classe para chamar o método main, o que não faria sentido, pois o programa ainda não foi iniciado. Portanto, o método main deve ser estático para que a JVM possa chamá-lo sem criar uma instância da classe.

2. Para que serve a classe Scanner?

A classe Scanner em Java é utilizada para obter a entrada do usuário a partir do console ou de um arquivo. Ela fornece métodos para analisar tipos primitivos e strings. Aqui estão alguns usos comuns da classe Scanner:

- **Entrada do Usuário:** A classe Scanner é frequentemente usada para ler a entrada do usuário a partir do console.
- **Leitura de Arquivos:** Também pode ser utilizada para ler informações de um arquivo.
- **Análise de Tipos Primitivos:** Pode analisar tipos primitivos como inteiros, números decimais, booleanos, etc.

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

As classes de repositório são uma parte comum da arquitetura de software em Java, normalmente usadas para interagir com a camada de persistência (como um banco de dados ou, no seu caso, um arquivo). Aqui está como o uso de classes de repositório impactou na organização do código:

- **Separação de Responsabilidades:** O uso de classes de repositório ajuda a separar a lógica de acesso aos dados (persistência) do restante da aplicação. Isso promove um código mais organizado e coeso.

- **Reutilização de Código:** Uma vez que a lógica de acesso aos dados está encapsulada nas classes de repositório, ela pode ser reutilizada em toda a aplicação. Isso evita a duplicação de código e promove a manutenção mais fácil.
- **Manutenção Simples:** Com as classes de repositório, é mais fácil manter e atualizar o código, pois as mudanças na lógica de acesso aos dados são isoladas das outras partes do sistema.