Nome do Campus: Polo Cruzeiro-Sp

Nome do Curso: Desenvolvimento FullStack

Nome da Disciplina: Iniciando o caminho pelo Java

Número da Turma: 2025.1

Semestre Letivo: Primeiro Semestre

Nome do Aluno: Moniza de Oliveira Silva Santos Pelegrini

Matrícula: 202401190829

Segundo Procedimento:

Criação do cadastro em modo texto

Título da Prática

Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.

Objetivo da Prática

Utilizar herança e polimorfismo na definição de entidades:

Criar uma hierarquia de classes (Pessoa, PessoaFisica, PessoaJuridica) para demonstrar herança e polimorfismo.

Utilizar persistência de objetos em arquivos binários:

Implementar métodos para salvar e recuperar dados em arquivos binários usando serialização.

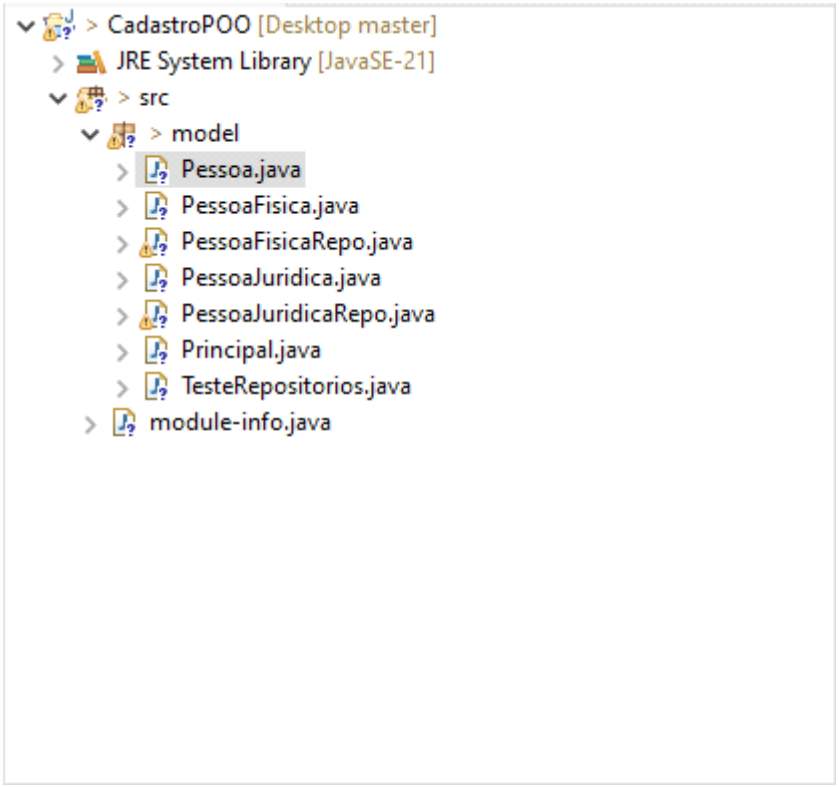
Implementar uma interface cadastral em modo texto:

Desenvolver um menu interativo para interação com o usuário via linha de comando.

Utilizar o controle de exceções da plataforma Java:

Tratar possíveis erros durante operações como persistência e recuperação de dados.

Códigos utlizados neste roteiro:



Classe Teste Principal:

TesteRepositorios.java

```java
package model;

import java.io.IOException;

public class TesteRepositorios {
    public static void main(String[] args) {
        // Teste do repositório de pessoas físicas
        try {
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            repo1.inserir(new PessoaFisica(1, "Mauricio Campos", "123.456.789-00", 46));
            repo1.inserir(new PessoaFisica(2, "Fatima Dantas", "111.222.333-50", 47));

            String arquivoPessoasFisicas = "pessoas_fisicas.dat";
            repo1.persistir(arquivoPessoasFisicas);

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar(arquivoPessoasFisicas);

            System.out.println("Pessoas Físicas Recuperadas:");
            for (PessoaFisica pessoa : repo2.obterTodos()) {
                pessoa.exibir();
                System.out.println("----------------------------");
            }
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao manipular o repositório de pessoas físicas: " + e.getMessage());
        }

        // Teste do repositório de pessoas jurídicas
        try {
            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            repo3.inserir(new PessoaJuridica(1, "Nome_empresa 1", "12.345.678/0001-00"));
            repo3.inserir(new PessoaJuridica(2, "Nome_empresa 2", "11.222.333/0001-00"));

            String arquivoPessoasJuridicas = "pessoas_juridicas.dat";
            repo3.persistir(arquivoPessoasJuridicas);

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
            repo4.recuperar(arquivoPessoasJuridicas);

            System.out.println("\nPessoas Jurídicas Recuperadas:");
            for (PessoaJuridica pessoa : repo4.obterTodos()) {
                pessoa.exibir();
                System.out.println("----------------------------");
            }
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Erro ao manipular o repositório de pessoas jurídicas: " + e.getMessage());
        }
    }
}
```

Resultado da execução:

```
<terminated> TesteRepositorios [Java Application] C:\Users\MAURICIO\Desktop\sts-4.27.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.5.v20241023-1957\jre\bin\javaw.exe  (11 de mar. de 2025 17:15:45 – 17:15:45 elaps
Pessoas Físicas Recuperadas:
ID: 1
Nome: Mauricio Campos
CPF: 123.456.789-00
Idade: 46
----------------------------
ID: 2
Nome: Fatima Dantas
CPF: 111.222.333-50
Idade: 47
----------------------------

Pessoas Jurídicas Recuperadas:
ID: 1
Nome: Nome_empresa 1
CNPJ: 12.345.678/0001-00
----------------------------
ID: 2
Nome: Nome_empresa 2
CNPJ: 11.222.333/0001-00
----------------------------
```

Análise e Conclusão

1 - O que são elementos estáticos e por que o método main adota esse modificador?

O que são elementos estáticos?

Elementos **estáticos** (`static`) pertencem à **classe**, em vez de pertencerem a uma instância específica dela. Isso significa que podem ser acessados diretamente pelo nome da classe, sem a necessidade de criar um objeto.

Exemplos de elementos estáticos:

- **Métodos estáticos:** Exemplo: `Math.sqrt()`.
- **Variáveis estáticas:** Exemplo: contadores globais.
- **Blocos estáticos:** Usados para inicialização de variáveis ou configuração antes da execução.

Por que o método main é estático?

O método `main` é o ponto de entrada de qualquer programa Java. Ele precisa ser **estático** porque a JVM (Java Virtual Machine) **não cria automaticamente uma instância da classe principal** ao iniciar o programa.

Se `main` não fosse estático, seria necessário criar um objeto da classe antes de executar o programa, o que tornaria o processo de inicialização mais complexo.
Com o modificador `static`, a JVM pode chamar `main` diretamente, sem a necessidade de instanciar a classe.

Relação com o projeto:

No projeto, o método `main` na classe `Principal` foi declarado como **static** para permitir que o sistema seja iniciado sem precisar criar uma instância da classe `Principal`.

---

2 - Para que serve a classe Scanner?

O que é a classe Scanner?

A classe `Scanner`, do pacote `java.util`, é utilizada para capturar **entradas do usuário** via console (linha de comando).
Ela fornece métodos convenientes para ler diferentes tipos de dados, como:

- **Texto (`String`)**: `nextLine()`.
- **Números inteiros (`int`)**: `nextInt()`.
- **Números de ponto flutuante (`double`)**: `nextDouble()`.

Como funciona?

- Um objeto `Scanner` é criado associado à entrada padrão (`System.in`), que representa o teclado.
- O programa pode então capturar a entrada do usuário usando métodos como `nextInt()`, `nextLine()`, `nextDouble()`, entre outros.

Importância no projeto:

No **menu interativo**, a classe `Scanner` foi fundamental para:

- Permitir que o usuário **escolhesse opções**.
- Inserir dados como **nome, CPF e CNPJ**.
- Interagir com o sistema de maneira intuitiva.

---

3 - Como o uso de classes de repositório impactou a organização do código?

O que são classes de repositório?

Classes de **repositório** (como `PessoaFisicaRepo` e `PessoaJuridicaRepo`) são responsáveis pela **persistência e recuperação de dados**.
Elas encapsulam a lógica de armazenamento e manipulação dos dados, separando-a da lógica de negócios e da interface com o usuário.

Impacto na organização do código:

✅ **Separação de Responsabilidades**

- As classes de repositório centralizam as operações de **CRUD** (*Create, Read, Update, Delete*) e persistência.
- A classe `Principal` foca exclusivamente na **interação com o usuário**, tornando o código mais modular e organizado.

✅ **Reutilização de Código**

- As classes de repositório podem ser reaproveitadas em **outros projetos** ou partes do sistema, sem necessidade de reescrever a lógica de persistência.

✅ **Facilidade de Testes**

- Como a persistência está isolada nas classes de repositório, torna-se **mais fácil testá-las** independentemente do restante do sistema.

✅ **Escalabilidade**

- Caso seja necessário alterar a forma de persistência (por exemplo, **migrar de arquivos binários para um banco de dados**), basta modificar as classes de repositório **sem impactar o restante do código**.

Exemplo no projeto:

A classe `PessoaFisicaRepo` gerencia todas as operações relacionadas a **pessoas físicas**, como:

- **Inserção, alteração, exclusão e persistência de dados**.

Isso permitiu que a classe `Principal` ficasse mais **limpa** e **focada no menu e na interação com o usuário**, tornando o código mais organizado e fácil de manter.