RELATÓRIO DE PRÁTICA

UNIVERSIDADE ESTÁCIO DE SÁ

RELATÓRIO DE PRÁTICA DE DESENVOLVIMENTO FULL STACK

Recife 2024/2

EROS SANTOS DE VASCONCELOS

RELATÓRIO DE PRÁTICA DE DESEVOLVIMENTO FULL STACK

Este trabalho é pré-requisito para aprovação na disciplina Iniciando o Caminho Pelo Java, Turma 9001

(modalidade EAD), da instituição de Ensino Sá,

POLO IPUTINGA - RECIFE - PE 2024

OBJETIVO DA PRÁTICA:

O objetivo principal deste artigo é desenvolver um sistema de registro de clientes em Java, utilizando conceitos orientados a objetos, como herança e polimorfismo, e armazenando dados de clientes como arquivos binários. O sistema também deverá funcionar em modo texto, permitindo a interação do usuário através do console. Outra coisa importante é utilizar um gerenciamento diferenciado para lidar com erros que ocorrem durante o processo de backup e recuperação de dados..

DESENVOLVIMENTO PASSO A PASSO

1. Criação do Projeto

Para começar, foi criado um projeto NetBeans chamado CadastroPOO. O projeto possui dois pacotes:

Exemplo: entidades e áreas criadas para o gerenciamento de clientes.

Principal: O método principal é responsável por interagir com o usuário.

2. Definição das entidades

Começamos por definir as entidades do sistema. A estrutura tradicional foi utilizada para criar as classes Pessoa, PessoaPhysica e PessoaJuridica.

Pessoa: Classe base com campos ID (inteiro) e nome (texto). Esta classe possui métodos construtor, getter e setter, bem como um método de exibição que imprime os dados do usuário no console.

PessoaFisica: Enviar Pessoa e adicionar campos cpf (texto) e número (inteiro). O método de exibição foi reescrito para incluir essas novas informações.

PessoaJuridica: Também herdado de Pessoa, mas com o campo CNPJ (texto) adicionado. Além disso, reescrevi o método de exibição para exibir CNPJ.

Essas entidades são implementadas com interfaces seriais e podem escrever e ler objetos de arquivos binários.

3. Criação dos Repositórios

Criação do repositório Dois repositórios foram criados para gerenciar os clientes

.

PessoaFisicaRepo: Responsável por gerenciar uma lista de pessoas naturais. Inclui métodos como add, modificar, excluir, obter naturais por id, obter todos os naturais, etc.

PessoaJuridicaRepo: Semelhante a PessoaFisicaRepo, mas gerencia uma lista de entidades da matriz.

Ambas as lojas armazenam dados em arquivos binários e fornecem uma maneira de recuperar esses dados de arquivos usando fluxos de entrada e saída Java. Exceções relacionadas à entrada e saída de dados são aumentadas e o sistema pode lidar com erros como erros de gravação ou leitura do arquivo.

4. Implementação do Modo Texto

No método principal, foi criada uma interface de modo de texto que apresenta ao usuário um menu e as seguintes opções.

inclui clientes (físicos ou legais).

Altere o cliente (pessoa natural ou legal).

Exclua o cliente por id.

Mostram clientes por id.

Mostrar todos os clientes.

Salva dados em arquivo.

Recupera dados do arquivo.

Sair do programa.

O programa interage com o usuário usando a classe Scanner para receber entrada do console. Dependendo da opção selecionada, o sistema realizará diversos gerenciamentos de armazenamento, incluindo adição de novos clientes ou cancelamento de clientes e restauração de dados.

5. Persistência dos Dados

As opções 6 e 7 do menu permitem que o usuário salve os dados dos clientes em arquivos binários e os recupere quando necessário. A lógica de persistência utiliza a serialização de objetos, onde os dados são gravados em dois arquivos distintos:

Um arquivo para pessoas físicas, com a extensão .fisica.bin.

Um arquivo para pessoas jurídicas, com a extensão .juridica.bin.

Essa separação dos arquivos facilita o gerenciamento de cada tipo de cliente. Exceções são tratadas nas operações de salvamento e recuperação, garantindo que o sistema continue funcionando mesmo diante de problemas como arquivos corrompidos ou indisponíveis.

Análise

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

Elementos estáticos em Java pertencem à classe e não a instâncias individuais. O método main é estático porque é o ponto de entrada do programa e precisa ser acessado pela JVM sem a necessidade de criar uma instância da classe. Assim, podemos iniciar o programa diretamente com o método main.

2. **Para que serve a classe Scanner?**

A classe Scanner é usada para ler entradas de diferentes fontes, como o console ou arquivos. No projeto, foi utilizada para capturar as entradas do usuário via console, permitindo que ele fornecesse os dados necessários para o cadastro e manipulação dos clientes.

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

O uso de classes de repositório trouxe uma clara separação entre a lógica de manipulação de dados e a lógica de interação com o usuário. Isso torna o código mais modular, facilitando a manutenção e a escalabilidade. Cada repositório é responsável por gerenciar um tipo específico de cliente, o que evita a duplicação de código e torna o sistema mais coeso.

Conclusão

Esta prática proporcionou uma experiência prática com conceitos fundamentais de Programação Orientada a Objetos e persistência de dados em Java. O sistema de cadastro de clientes foi implementado com sucesso, utilizando herança, polimorfismo e o uso de repositórios para organizar os dados. O controle de exceções foi aplicado para garantir que o sistema pudesse lidar com erros na leitura e gravação de arquivos. No final, o sistema foi capaz de executar todas as funcionalidades propostas de maneira eficiente.

CÓDIGOS UTILIZADOS

MAIN:

```
package CadastroPOO;

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

import java.io.IOException;
import java.util.InputMismatchException;
import java.util.List;
import java.util.Scanner;

public class Main {
    private static PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
    private static PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();
    private static Scanner scanner = new Scanner(System.in);

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

```
do {
    mostrarMenu();
    opcao = lerOpcao();

    switch (opcao) {
        case 1:
            incluir();
            break;
        case 2:
            alterar();
            break;
        case 3:
            excluir();
            break;
        case 4:
            obterPorId();
            break;
        case 5:
            obterTodos();
            break;
        case 6:
            salvarDados();
            break;
```

```
            case 7:
                recuperarDados();
                break;
            case 0:
                System.out.println("Saindo...");
                break;
            default:
                System.out.println("Opção inválida. Tente novamente.");
        }
    } while (opcao != 0);

    scanner.close();
  }

  private static void mostrarMenu() {
    System.out.println("\nMenu:");
    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");
```

```
    }

    private static int lerOpcao() {
        int opcao = -1;
        while (opcao < 0 || opcao > 7) {
            System.out.print("Escolha uma opção: ");
            try {
                opcao = scanner.nextInt();
            } catch (InputMismatchException e) {
                System.out.println("Entrada inválida. Digite um
número.");
                scanner.next();
            }
        }
        return opcao;
    }

    private static void incluir() {
        System.out.println("Escolha o tipo: 1 - Física, 2 - Jurídica");
        int tipo = lerOpcao();
        scanner.nextLine();
        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();
            PessoaFisica pf = new PessoaFisica(id, nome, cpf, idade);
            pessoaFisicaRepo.inserir(pf);
            System.out.println("Pessoa Física incluída com sucesso.");
        } 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();
            PessoaJuridica pj = new PessoaJuridica(id, nome, cnpj);
            pessoaJuridicaRepo.inserir(pj);
            System.out.println("Pessoa Jurídica incluída com
sucesso.");
        } else {
            System.out.println("Tipo inválido.");
        }
    }
```

```
    private static void alterar() {
        System.out.println("Escolha o tipo: 1 - Física, 2 - Jurídica");
        int tipo = lerOpcao();
        scanner.nextLine();
        if (tipo == 1) {
            System.out.print("ID da Pessoa Física a ser alterada: ");
            int id = scanner.nextInt();
            scanner.nextLine();
            PessoaFisica pf = pessoaFisicaRepo.obter(id);
            if (pf != null) {
                System.out.println("Dados atuais: " + pf.getNome() + ", CPF: " + pf.getCpf() + ", Idade: " + pf.getIdade());
                System.out.print("Novo Nome: ");
                String nome = scanner.nextLine();
                System.out.print("Novo CPF: ");
                String cpf = scanner.nextLine();
                System.out.print("Nova Idade: ");
                int idade = scanner.nextInt();
                pf.setNome(nome);
                pf.setCpf(cpf);
                pf.setIdade(idade);
                pessoaFisicaRepo.alterar(pf);
                System.out.println("Pessoa Física alterada com sucesso.");
```

```
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipo == 2) {
            System.out.print("ID da Pessoa Jurídica a ser alterada: ");
            int id = scanner.nextInt();
            scanner.nextLine();
            PessoaJuridica pj = pessoaJuridicaRepo.obter(id);
            if (pj != null) {
                System.out.println("Dados atuais: " + pj.getNome() + ", CNPJ: " + pj.getCnpj());
                System.out.print("Novo Nome: ");
                String nome = scanner.nextLine();
                System.out.print("Novo CNPJ: ");
                String cnpj = scanner.nextLine();
                pj.setNome(nome);
                pj.setCnpj(cnpj);
                pessoaJuridicaRepo.alterar(pj);
                System.out.println("Pessoa Jurídica alterada com sucesso.");
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Tipo inválido.");
```

```
        }
    }

    private static void excluir() {
        System.out.println("Escolha o tipo: 1 - Física, 2 - Jurídica");
        int tipo = lerOpcao();
        if (tipo == 1) {
            System.out.print("ID da Pessoa Física a ser excluída: ");
            int id = scanner.nextInt();
            pessoaFisicaRepo.excluir(id);
            System.out.println("Pessoa Física excluída com
sucesso.");
        } else if (tipo == 2) {
            System.out.print("ID da Pessoa Jurídica a ser excluída: ");
            int id = scanner.nextInt();
            pessoaJuridicaRepo.excluir(id);
            System.out.println("Pessoa Jurídica excluída com
sucesso.");
        } else {
            System.out.println("Tipo inválido.");
        }
    }

    private static void obterPorId() {
        System.out.println("Escolha o tipo: 1 - Física, 2 - Jurídica");
```

```
        int tipo = lerOpcao();
        if (tipo == 1) {
            System.out.print("ID da Pessoa Física a ser exibida: ");
            int id = scanner.nextInt();
            PessoaFisica pf = pessoaFisicaRepo.obter(id);
            if (pf != null) {
                pf.exibir();
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipo == 2) {
            System.out.print("ID da Pessoa Jurídica a ser exibida: ");
            int id = scanner.nextInt();
            PessoaJuridica pj = pessoaJuridicaRepo.obter(id);
            if (pj != null) {
                pj.exibir();
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Tipo inválido.");
        }
    }
```

```
    private static void obterTodos() {
        System.out.println("Escolha o tipo: 1 - Física, 2 - Jurídica");
        int tipo = lerOpcao();
        if (tipo == 1) {
            List<PessoaFisica> pessoasFisicas = pessoaFisicaRepo.obterTodos();
            if (!pessoasFisicas.isEmpty()) {
                for (PessoaFisica pf : pessoasFisicas) {
                    pf.exibir();
                }
            } else {
                System.out.println("Nenhuma Pessoa Física encontrada.");
            }
        } else if (tipo == 2) {
            List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaRepo.obterTodos();
            if (!pessoasJuridicas.isEmpty()) {
                for (PessoaJuridica pj : pessoasJuridicas) {
                    pj.exibir();
                }
            } else {
                System.out.println("Nenhuma Pessoa Jurídica encontrada.");
            }
        } else {
```

```
            System.out.println("Tipo inválido.");
        }
    }

    private static void salvarDados() {
        System.out.print("Digite o prefixo para os arquivos: ");
        String prefixo = scanner.next();
        try {
            pessoaFisicaRepo.persistir(prefixo + ".fisica.bin");
            pessoaJuridicaRepo.persistir(prefixo + ".juridica.bin");
            System.out.println("Dados salvos com sucesso.");
        } catch (IOException e) {
            System.out.println("Erro ao salvar dados: " +
e.getMessage());
            e.printStackTrace();
        }
    }

    private static void recuperarDados() {
        System.out.print("Digite o prefixo para os arquivos: ");
        String prefixo = scanner.next();
        try {
            pessoaFisicaRepo.recuperar(prefixo + ".fisica.bin");
            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());
            e.printStackTrace();
        }
    }
}
```

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);
    }
}
```

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);
    }
}
```

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();
```

```
        }
    }
}
```

PESSOAJURIDICAREPO:

```
package model;

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

public class PessoaJuridicaRepo {
    private List<PessoaJuridica> pessoasJuridicas;

    public PessoaJuridicaRepo() {
        pessoasJuridicas = new ArrayList<>();
    }

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

    public void alterar(PessoaJuridica pessoa) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            if (pessoasJuridicas.get(i).getId() == pessoa.getId()) {
```

```
            pessoasJuridicas.set(i, pessoa);
            return;
        }
    }
}

public void excluir(int id) {
    pessoasJuridicas.removeIf(pessoa -> pessoa.getId() == id);
}

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 oos = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            oos.writeObject(pessoasJuridicas);
        }
    }

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

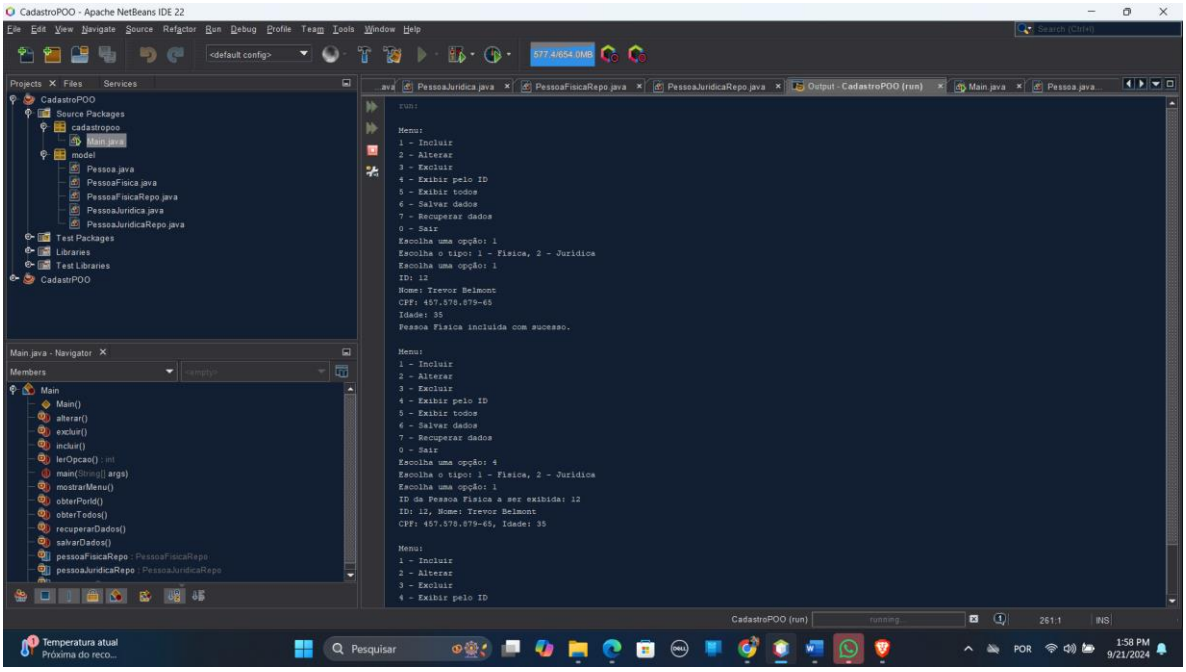
PESSOAFISICA:

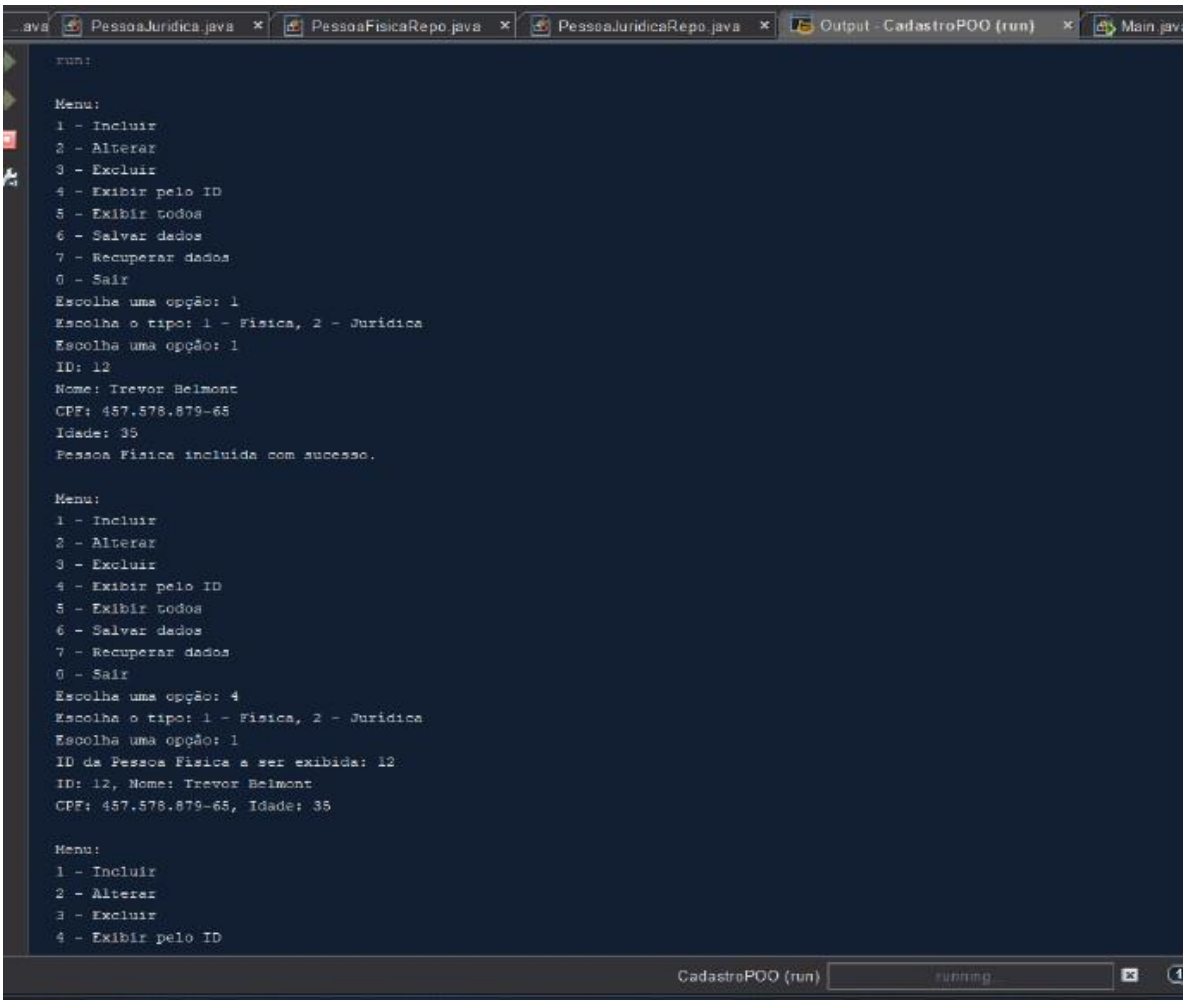
```
package model;

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);
    }
}
```

RESULTADO DA EXECUÇÃO DO CÓDIGO:




PessoaJuridica.java
PessoaFisicaRepo.java
PessoaJuridicaRepo.java
Output - CadastroPOO (run)
Main.java
run:
Menu:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair
Escolha uma opção: 1
Escolha o tipo: 1 - Fisica, 2 - Juridica
Escolha uma opção: 1
ID: 12
Nome: Trevor Belmont
CPF: 457.578.879-65
Idade: 35
Pessoa Fisica incluida com sucesso.
Menu:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair
Escolha uma opção: 4
Escolha o tipo: 1 - Fisica, 2 - Juridica
Escolha uma opção: 1
ID da Pessoa Fisica a ser exibida: 12
ID: 12, Nome: Trevor Belmont
CPF: 457.578.879-65, Idade: 35
Menu:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
CadastroPOO (run)
running...