



# Estácio

**FACULDADE ESTÁCIO**

-

**POLO ARARUAMA – RJ**

**DESENVOLVIMENTO FULL STACK**

**DISCIPLINA – INICIANDO O CAMINHO PELO JAVA**

-

**TURMA – 2023.2**

**SEMESTRE – 3**

-

-

-

-

-

-

ARARUAMA, JUNHO 2024.

DESENVOLVIMENTO FULL STACK

-  
DISCIPLINA – INICIANDO O CAMINHO PELO JAVA

TURMA – 2023.2

SEMESTRE – 3

-  
ALUNO – GUSTAVO IGOR DA SILVA

-  
TUTOR – MARIA MANSO

-  
GITHUB - <https://github.com/gustavoxokito/trabalhojava.git>

## **RESUMO**

O código apresentado implementa um sistema de cadastro de pessoas físicas e jurídicas utilizando a linguagem Java. O sistema permite incluir, alterar, excluir, exibir pelo ID, exibir todos, salvar e recuperar dados de pessoas físicas e jurídicas. Para isso, são utilizadas classes de repositório (PessoaFisicaRepo e PessoaJuridicaRepo) que armazenam os dados e oferecem métodos para manipulá-los. A classe CadastroPOO contém o método main, que é responsável por exibir o menu de opções e gerenciar a interação com o usuário através da classe Scanner.

Palavras-chave: Programação Orientada a Objetos (POO), Java Application, Modelagem de Dados, Scanner, Gerenciamento de Dados, Persistência de Dados, Teste de Funcionalidades, Armazenamento em Git, Eficiência e Organização.

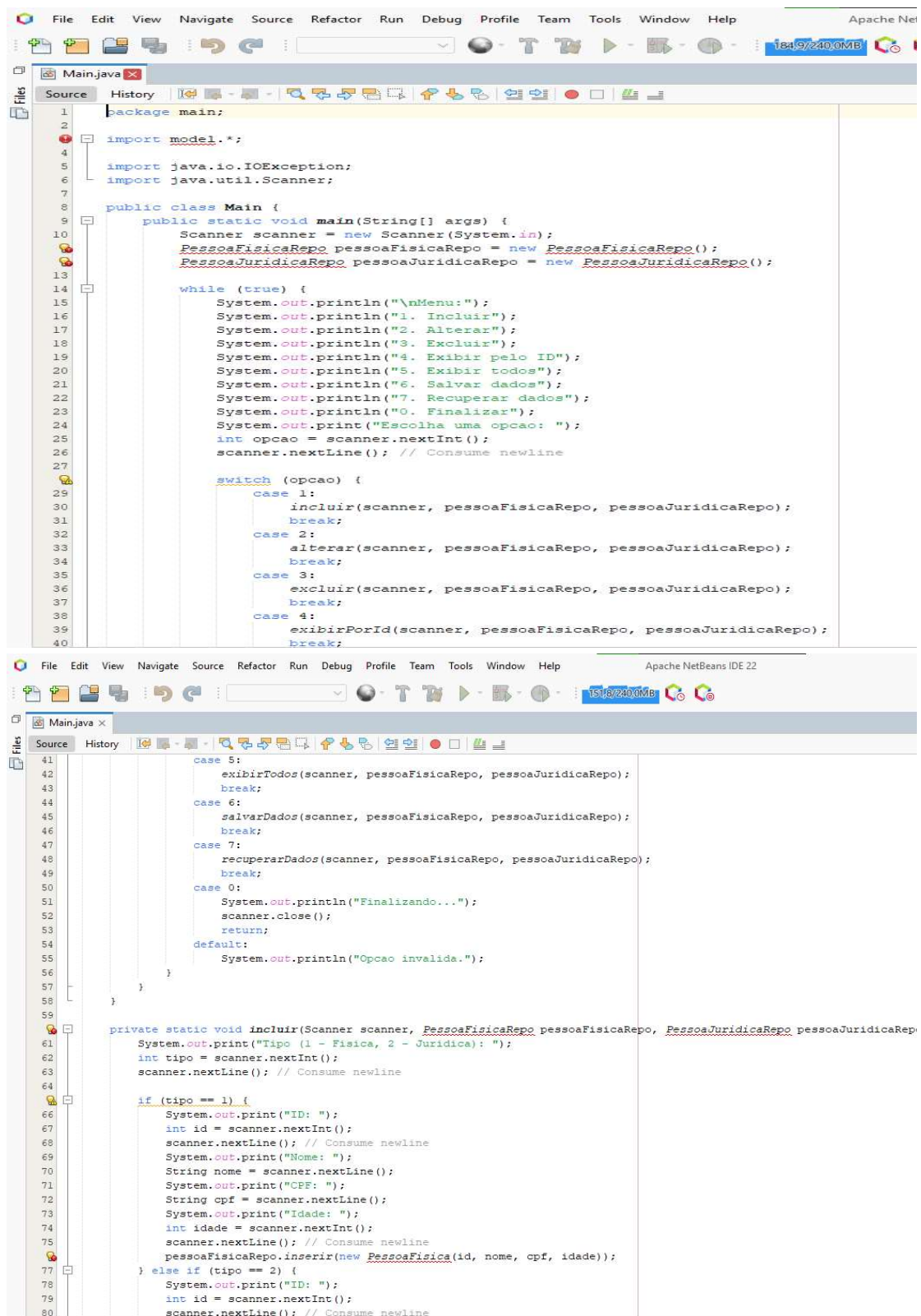
## **• INTRODUÇÃO**

Este trabalho apresenta o desenvolvimento de um sistema de cadastro de pessoas físicas e jurídicas utilizando a linguagem de programação Java. O sistema é projetado para operar com eficiência e clareza, permitindo a inclusão, alteração, exclusão, exibição e persistência de dados. A implementação faz uso de conceitos fundamentais de programação orientada a objetos (POO), como encapsulamento, modularização e reutilização de código, o que facilita a manutenção e expansão do sistema. O ponto de entrada da aplicação é o método main, marcado como static, permitindo que a JVM (Java Virtual Machine) o invoque diretamente sem a necessidade de instanciar a classe CadastroPOO. A interação com o usuário é gerida pela classe Scanner, que captura entradas do usuário e facilita a navegação pelas diversas opções do menu.

As operações de manipulação de dados são centralizadas em classes de repositório (PessoaFisicaRepo e PessoaJuridicaRepo), que encapsulam a lógica de acesso a dados e promovem uma separação clara entre a lógica de negócio e a lógica de interface com o usuário. Esta estrutura não só melhora a legibilidade do código, mas também segue o princípio da responsabilidade única, tornando cada classe responsável por uma única parte do sistema. Este documento também discutirá os elementos estáticos e seu papel no método main, a funcionalidade da classe Scanner, e

o impacto positivo do uso de classes de repositório na organização do código.

- **main** - Main.Java (atualizada)



```
1 package main;
2
3 import model.*;
4 import java.io.IOException;
5 import java.util.Scanner;
6
7 public class Main {
8     public static void main(String[] args) {
9         Scanner scanner = new Scanner(System.in);
10        PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
11        PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();
12
13        while (true) {
14            System.out.println("\nMenu:");
15            System.out.println("1. Incluir");
16            System.out.println("2. Alterar");
17            System.out.println("3. Excluir");
18            System.out.println("4. Exibir pelo ID");
19            System.out.println("5. Exibir todos");
20            System.out.println("6. Salvar dados");
21            System.out.println("7. Recuperar dados");
22            System.out.println("0. Finalizar");
23            System.out.print("Escolha uma opcao: ");
24            int opcao = scanner.nextInt();
25            scanner.nextLine(); // Consume newline
26
27            switch (opcao) {
28                case 1:
29                    incluir(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
30                    break;
31                case 2:
32                    alterar(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
33                    break;
34                case 3:
35                    excluir(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
36                    break;
37                case 4:
38                    exibirPorId(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
39                    break;
40                case 5:
41                    exibirTodos(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
42                    break;
43                case 6:
44                    salvarDados(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
45                    break;
46                case 7:
47                    recuperarDados(scanner, pessoaFisicaRepo, pessoaJuridicaRepo);
48                    break;
49                case 0:
50                    System.out.println("Finalizando...");
51                    scanner.close();
52                    return;
53                default:
54                    System.out.println("Opcao invalida.");
55            }
56        }
57    }
58
59    private static void incluir(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo) {
60        System.out.print("Tipo (1 - Fisica, 2 - Juridica): ");
61        int tipo = scanner.nextInt();
62        scanner.nextLine(); // Consume newline
63
64        if (tipo == 1) {
65            System.out.print("ID: ");
66            int id = scanner.nextInt();
67            scanner.nextLine(); // Consume newline
68            System.out.print("Nome: ");
69            String nome = scanner.nextLine();
70            System.out.print("CPF: ");
71            String cpf = scanner.nextLine();
72            System.out.print("Idade: ");
73            int idade = scanner.nextInt();
74            scanner.nextLine(); // Consume newline
75            pessoaFisicaRepo.inserir(new PessoaFisica(id, nome, cpf, idade));
76        } else if (tipo == 2) {
77            System.out.print("ID: ");
78            int id = scanner.nextInt();
79            scanner.nextLine(); // Consume newline
80            System.out.print("Nome: ");
81            String nome = scanner.nextLine();
82            System.out.print("CPF: ");
83            String cpf = scanner.nextLine();
84            System.out.print("Idade: ");
85            int idade = scanner.nextInt();
86            scanner.nextLine(); // Consume newline
87            pessoaJuridicaRepo.inserir(new PessoaJuridica(id, nome, cpf, idade));
88        }
89    }
90}
```

```
81      System.out.print("Nome: ");
82      String nome = scanner.nextLine();
83      System.out.print("CNPJ: ");
84      String cnpj = scanner.nextLine();
85      pessoaJuridicaRepo.inserir(new PessoaJuridica(id, nome, cnpj));
86  } else {
87      System.out.println("Tipo invalido.");
88  }
89  }
90
91  private static void alterar(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo)
92  {
93      System.out.print("Tipo (1 - Fisica, 2 - Juridica): ");
94      int tipo = scanner.nextInt();
95      scanner.nextLine(); // Consume newline
96
97      if (tipo == 1) {
98          System.out.print("ID: ");
99          int id = scanner.nextInt();
100         scanner.nextLine(); // Consume newline
101         PessoaFisica pessoaFisica = pessoaFisicaRepo.obter(id);
102         if (pessoaFisica != null) {
103             System.out.println("Dados atuais: ");
104             pessoaFisica.exibir();
105             System.out.print("Nome: ");
106             String nome = scanner.nextLine();
107             System.out.print("CPF: ");
108             String cpf = scanner.nextLine();
109             System.out.print("Idade: ");
110             int idade = scanner.nextInt();
111             scanner.nextLine(); // Consume newline
112             pessoaFisicaRepo.alterar(new PessoaFisica(id, nome, cpf, idade));
113         } else {
114             System.out.println("Pessoa Fisica nao encontrada.");
115         }
116     } else if (tipo == 2) {
117         System.out.print("ID: ");
118         int id = scanner.nextInt();
119         scanner.nextLine(); // Consume newline
120         PessoaJuridica pessoaJuridica = pessoaJuridicaRepo.obter(id);
121         if (pessoaJuridica != null) {
122             System.out.println("Dados atuais: ");
123             pessoaJuridica.exibir();
124             System.out.print("Nome: ");
125             String nome = scanner.nextLine();
126             System.out.print("CNPJ: ");
127             String cnpj = scanner.nextLine();
128             pessoaJuridicaRepo.alterar(new PessoaJuridica(id, nome, cnpj));
129         } else {
130             System.out.println("Pessoa Juridica nao encontrada.");
131         }
132     } else {
133         System.out.println("Tipo invalido.");
134     }
135  }
136
137  private static void excluir(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo)
138  {
139      System.out.print("Tipo (1 - Fisica, 2 - Juridica): ");
140      int tipo = scanner.nextInt();
141      scanner.nextLine(); // Consume newline
142
143      System.out.print("ID: ");
144      int id = scanner.nextInt();
145      scanner.nextLine(); // Consume newline
146
147      if (tipo == 1) {
148          pessoaFisicaRepo.excluir(id);
149      } else if (tipo == 2) {
150          pessoaJuridicaRepo.excluir(id);
151      } else {
152          System.out.println("Tipo invalido.");
153      }
154  }
155
156  private static void exibirPorId(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo)
157  {
158      System.out.print("Tipo (1 - Fisica, 2 - Juridica): ");
159      int tipo = scanner.nextInt();
160      scanner.nextLine(); // Consume newline
161
162      System.out.print("ID: ");
163      int id = scanner.nextInt();
```

```

161 scanner.nextLine(); // Consume newline
162
163 if (tipo == 1) {
164     PessoaFisica pessoaFisica = pessoaFisicaRepo.obter(id);
165     if (pessoaFisica != null) {
166         pessoaFisica.exibir();
167     } else {
168         System.out.println("Pessoa Fisica nao encontrada.");
169     }
170 } else if (tipo == 2) {
171     PessoaJuridica pessoaJuridica = pessoaJuridicaRepo.obter(id);
172     if (pessoaJuridica != null) {
173         pessoaJuridica.exibir();
174     } else {
175         System.out.println("Pessoa Juridica nao encontrada.");
176     }
177 } else {
178     System.out.println("Tipo invalido.");
179 }
180
181 private static void exibirTodos(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo) {
182     System.out.print("Tipo (1 - Fisica, 2 - Juridica): ");
183     int tipo = scanner.nextInt();
184     scanner.nextLine(); // Consume newline
185
186     if (tipo == 1) {
187         for (PessoaFisica pessoaFisica : pessoaFisicaRepo.obterTodos()) {
188             pessoaFisica.exibir();
189         }
190     } else if (tipo == 2) {
191         for (PessoaJuridica pessoaJuridica : pessoaJuridicaRepo.obterTodos()) {
192             pessoaJuridica.exibir();
193         }
194     } else {
195         System.out.println("Tipo invalido.");
196     }
197 }
198
199 private static void salvarDados(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo) {

```

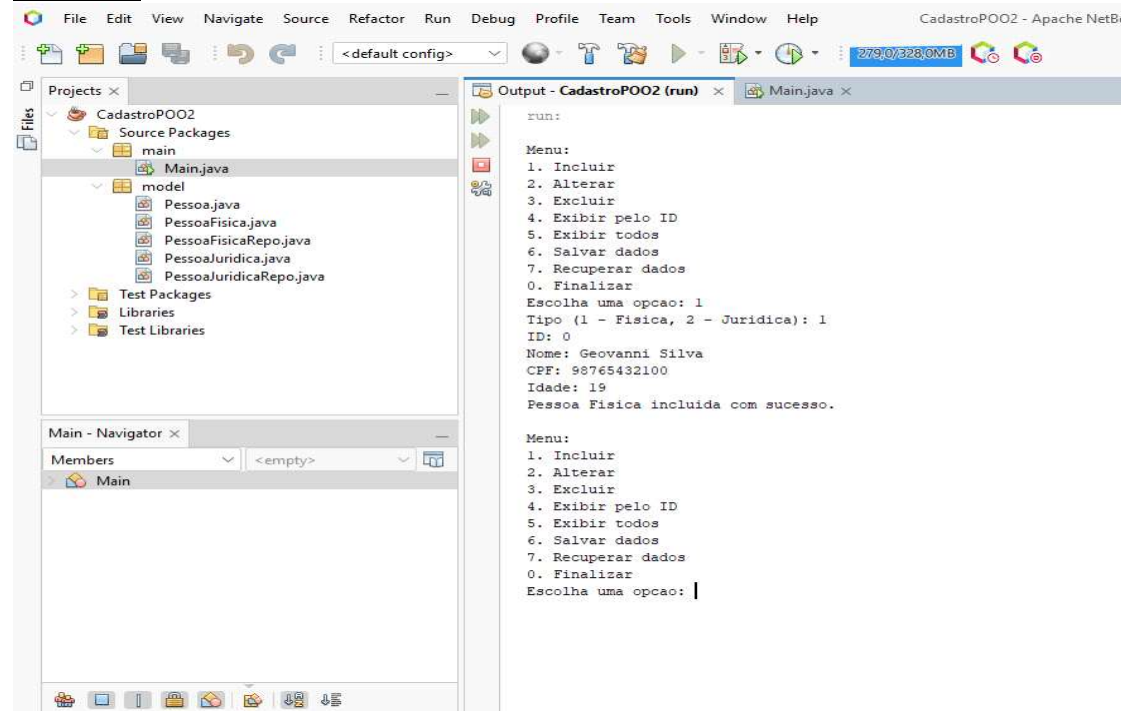
```

201 System.out.print("Prefixo dos arquivos: ");
202 String prefixo = scanner.nextLine();
203
204 try {
205     pessoaFisicaRepo.persistir(prefixo + ".fisica.bin");
206     pessoaJuridicaRepo.persistir(prefixo + ".juridica.bin");
207     System.out.println("Dados salvos com sucesso.");
208 } catch (IOException e) {
209     System.out.println("Erro ao salvar dados: " + e.getMessage());
210 }
211
212 private static void recuperarDados(Scanner scanner, PessoaFisicaRepo pessoaFisicaRepo, PessoaJuridicaRepo pessoaJuridicaRepo) {
213     System.out.print("Prefixo dos arquivos: ");
214     String prefixo = scanner.nextLine();
215
216     try {
217         pessoaFisicaRepo.recuperar(prefixo + ".fisica.bin");
218         pessoaJuridicaRepo.recuperar(prefixo + ".juridica.bin");
219         System.out.println("Dados recuperados com sucesso.");
220     } catch (IOException | ClassNotFoundException e) {
221         System.out.println("Erro ao recuperar dados: " + e.getMessage());
222     }
223 }
224
225 }
226
227

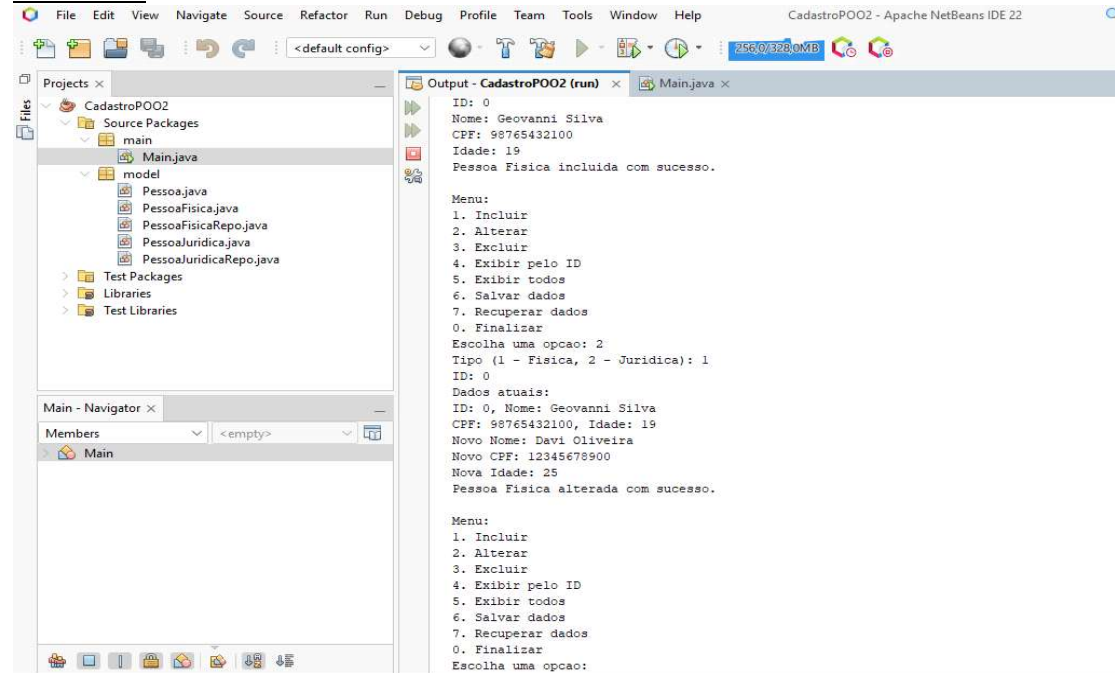
```

## Resultados Da Execução dos Códigos

### 1 - Incluir

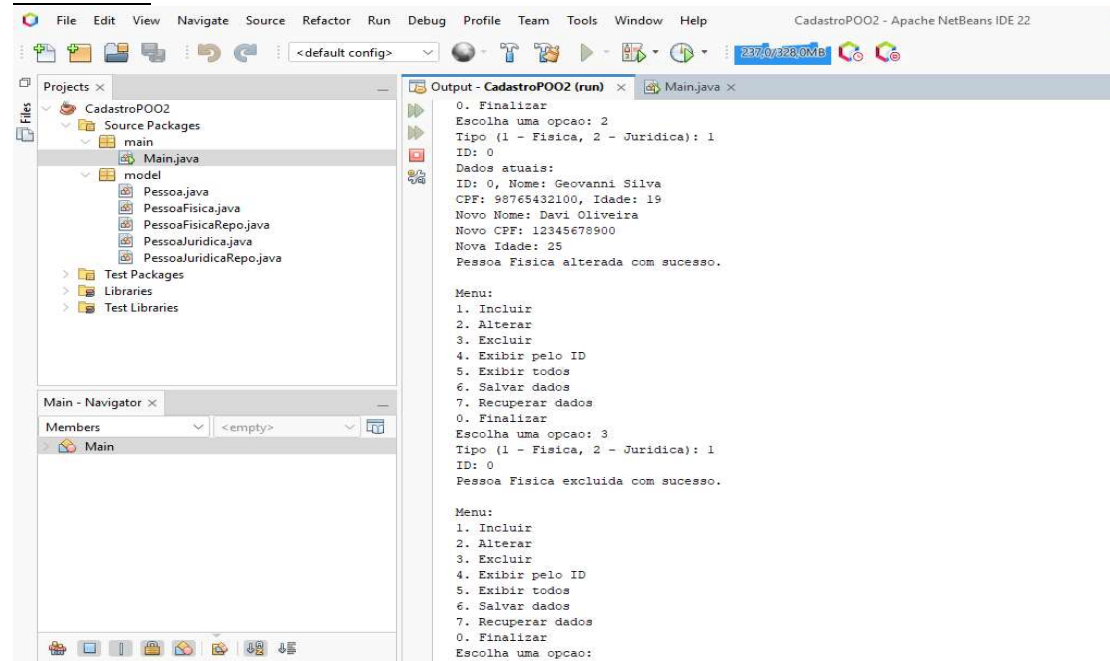


### 2 - Alterar

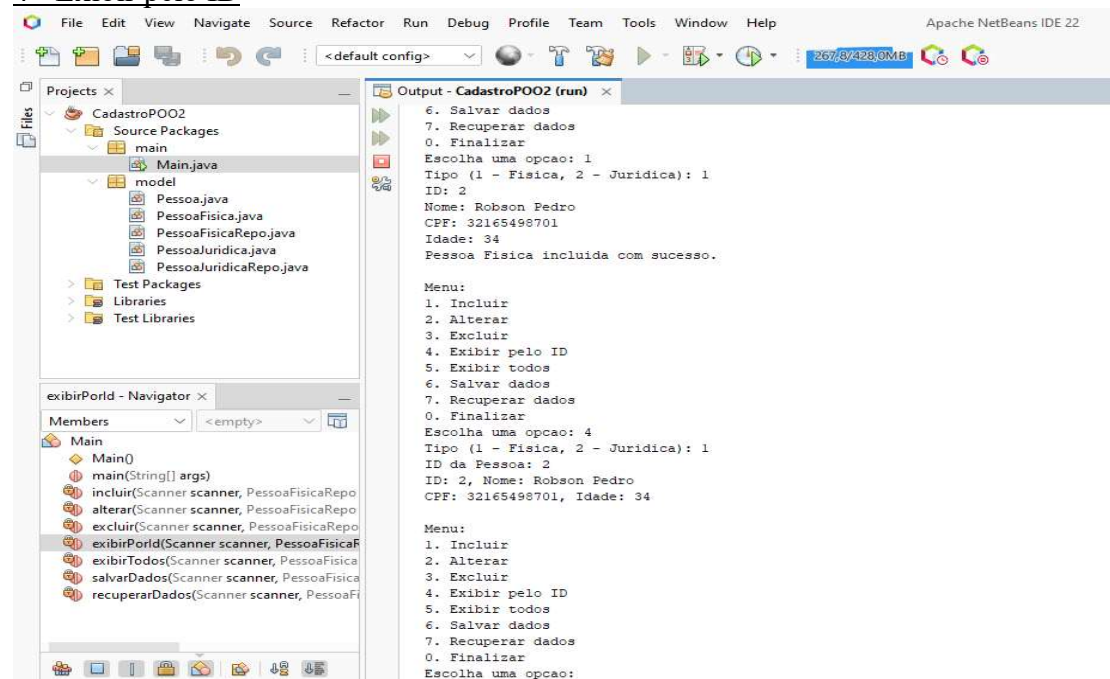




### 3 - Excluir

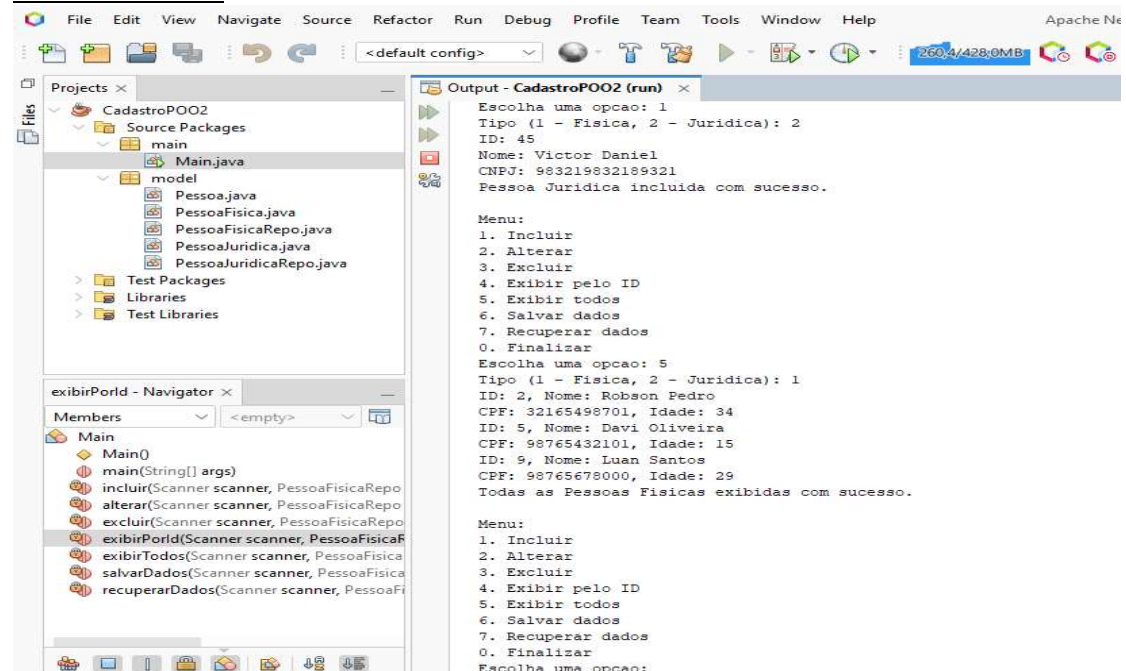


### 4 - Exibir pelo ID

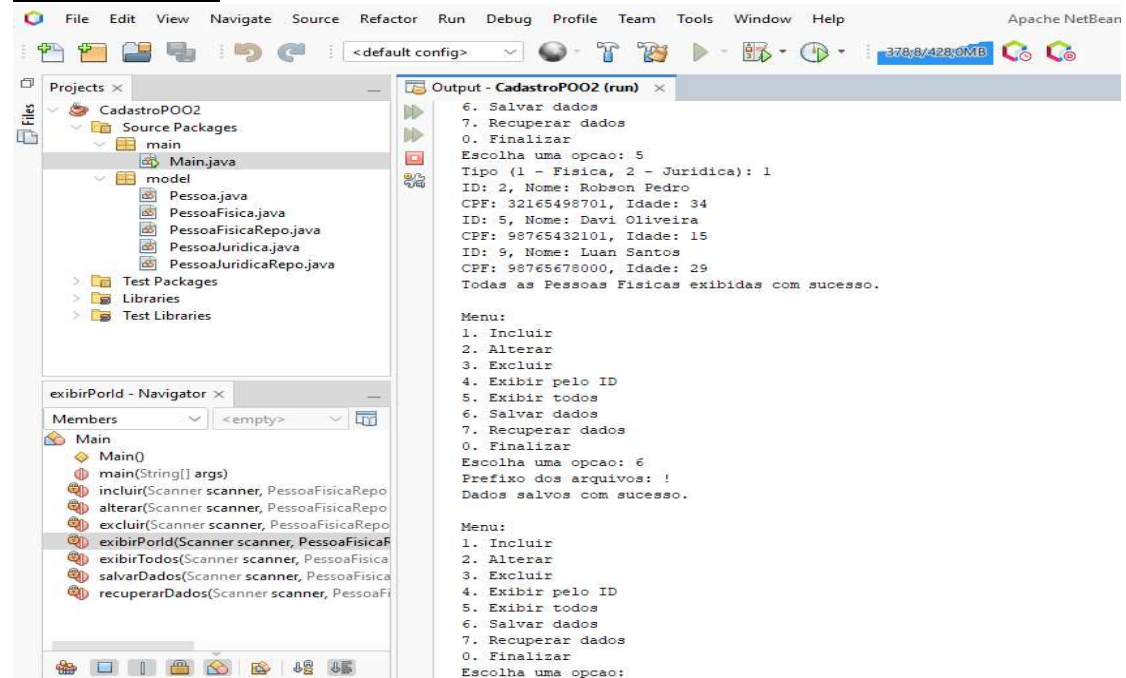




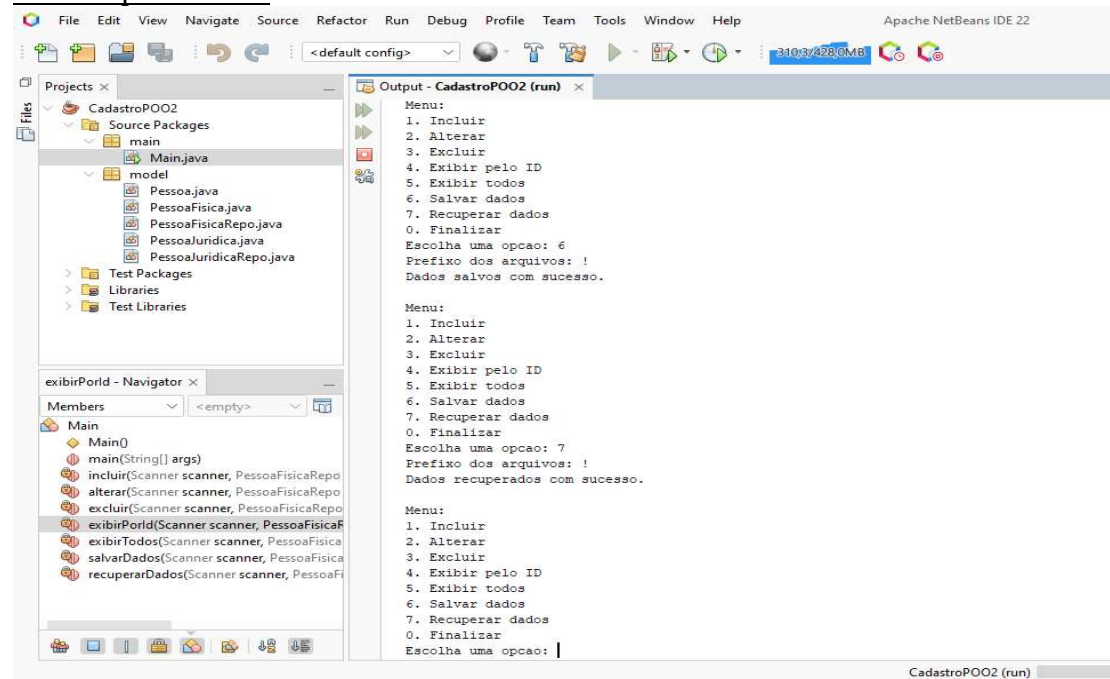
## 5 - Exibir Todos



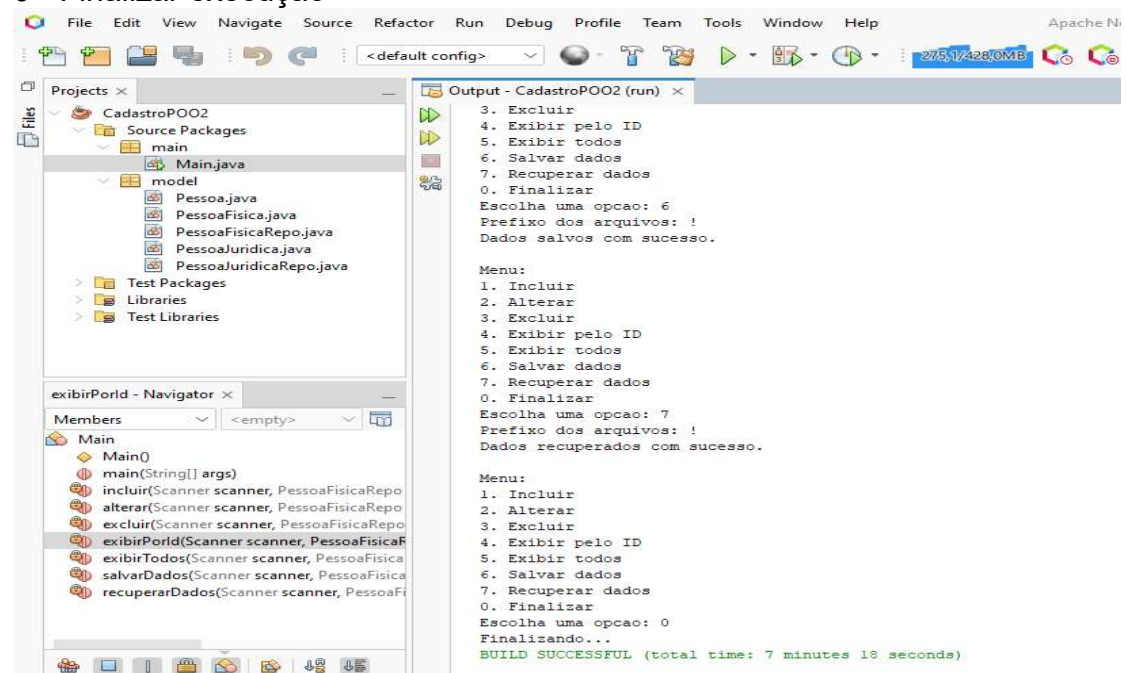
## 6 - Salvar dados



## 7 - Recuperar dados



## 0 - Finalizar execução



- **ANALISE**

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

Elementos estáticos, como o método main, são associados à classe e não às suas instâncias, permitindo que a JVM (Java Virtual Machine) invoque main diretamente para iniciar a execução do programa. A utilização do modificador static é essencial para que o método main possa ser executado sem a necessidade de instanciar a classe CadastroPOO. Isso é crucial, pois o método main serve como ponto de entrada da aplicação, e a JVM precisa ser capaz de chamá-lo diretamente sem criar um objeto da classe.

-- Para que serve a classe Scanner?

A classe Scanner desempenha um papel crucial na interação com o usuário, permitindo a leitura de diversos tipos de dados (como int, double, String) a partir da entrada padrão, geralmente o teclado. No contexto do sistema, Scanner é utilizado para capturar as escolhas do usuário no menu e para coletar os dados necessários nas operações de inclusão, alteração e exclusão de cadastros. Isso facilita a entrada e manipulação de dados, tornando a interface do usuário mais amigável e eficiente.

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

O uso de classes de repositório (PessoaFisicaRepo e PessoaJuridicaRepo) é uma abordagem significativa que impacta positivamente a organização do código. Essas classes são responsáveis pela gestão dos dados, oferecendo métodos para inserir, alterar, excluir e recuperar registros. Ao encapsular a lógica de manipulação de dados, os repositórios promovem o princípio da responsabilidade única, isolando a lógica de persistência da lógica de interface com o usuário. Isso resulta em um código mais modular, legível e de fácil manutenção.

Além disso, a separação das operações de cadastro em métodos específicos na classe CadastroPOO (como incluir, alterar, excluir, exibirPorId, exibirTodos, salvarDados, recuperarDados) contribui para uma melhor organização e clareza do código. Cada método tem uma responsabilidade bem definida, tornando o fluxo do programa mais intuitivo e facilitando futuras modificações ou expansões do sistema.

A modularização do código, juntamente com a utilização de repositórios, também facilita a implementação de funcionalidades adicionais, como validação de dados e tratamento de exceções. Por exemplo, ao persistir dados em arquivos binários, os métodos persistir e recuperar nas classes de repositório garantem a integridade dos dados armazenados, permitindo a recuperação e utilização dos dados em execuções subsequentes do programa.

## • CONCLUSÃO

O sistema de cadastro de pessoas físicas e jurídicas desenvolvido em Java demonstra a eficácia da aplicação de conceitos de programação orientada a objetos, como modularização e encapsulamento. A utilização de elementos estáticos, como o método main, facilita a inicialização do programa, enquanto a classe Scanner simplifica a captura de entradas do usuário, tornando a interação mais intuitiva. As classes de repositório, responsáveis pela gestão dos dados, promovem uma separação clara entre a lógica de negócio e a interface com o usuário, resultando em um código mais organizado e de fácil manutenção.

A modularização do código não só melhora a legibilidade, mas também facilita futuras expansões e adaptações do sistema. A abordagem adotada permite uma manutenção eficiente e a implementação de novas funcionalidades de maneira estruturada. Em suma, a estrutura do sistema é robusta, escalável e mantém uma clara separação de responsabilidades, garantindo a integridade e a eficiência da aplicação.

## • REFERÊNCIAS

TutorialsPoint, Java Tutorial. Acessado em 2024. <https://www.tutorialspoint.com/java/index.htm>.

W3Schools, Java Tutorial. Acessado em 2024. <https://www.w3schools.com/java/>.

ORACLE, Java Downloads Acessado em 2024, [www.oracle.com/java/technologies/downloads/](http://www.oracle.com/java/technologies/downloads/).

GeeksforGeeks, Java Programming Language, Acessado em 2024. [www.geeksforgeeks.org/java](http://www.geeksforgeeks.org/java).