



Estácio

Universidade Estácio de Sá

3º período em Desenvolvimento Full stack

Matrícula: 202302891292

Aluno: Luiz Fabrício Mello Ferreira

Iniciando o caminho pelo Java

1º Procedimento | Criação das Entidades e Sistema de Persistência

Objetivo da Prática

Conforme o passo-a-passo instruído pelo curso, é mostrar a forma de criar um aplicativo de cadastro, um CRUD (Create, Read, Update, Delete) com o Java de uma forma simples e objetiva, fazendo com que os alunos coloquem em prática todo o assunto estudado no nível 1. Com essas instruções e seguindo os passos, ao final estaremos aptos a elaborar um sistema de cadastro que cria, ler, altera e deleta o que foi cadastrado.

Quais as vantagens e desvantagens do uso de herança?

Na utilização da herança no Java, uma das maiores vantagens seria no quesito de reutilização de código, pois a superclasse compartilha todo o seu comportamento, atributos para as outras classes, evitando a repetição de código, promovendo a reutilização e a modularidade do mesmo.

Com a herança, temos o polimorfismo, onde um objeto de uma subclasse pode ser tratado como um objeto da superclasse. Isso facilita a escrita de código genérico e flexível. Além do reaproveitamento de código, todas as classes herdadas, ficaram organizadas em uma hierarquia, representando relações de especialização e generalização entre objetos do mundo real. Isso facilita a compreensão e a manutenção do código.

Ao utilizar a herança, é criado um acoplamento forte entre a subclasse e superclasse, o que pode tornar o código mais difícil de entender e manter. Mudanças na superclasse podem afetar as subclasses e vice-versa.

Quando se trata de heranças múltiplas, o Java não tem suporte. Devido a hierarquias profundas ao ser utilizado a herança, o código pode se tornar complexo e difícil de gerenciar, pois pode levar a uma estrutura excessivamente complicada e difícil de entender.

A herança em Java é uma poderosa ferramenta de programação orientada a objetos que oferece vantagens significativas, como reutilização de código, extensibilidade e polimorfismo. No entanto, é importante usar a herança com cuidado e considerar suas potenciais desvantagens, como acoplamento forte e herança frágil, ao projetar sistemas de software.

Por que a interface `Serializable` é necessária ao efetuar persistência em arquivos binários?

A interface `Serializable` em Java indica que uma classe pode ser convertida em uma sequência de bytes para armazenamento ou transmissão. Essa conversão é essencial para persistir objetos em arquivos binários, permitindo que o Java os armazene e reconstrua quando necessário.

Como o paradigma funcional é utilizado pela API `stream` no Java?

A API `Stream` em Java, desde o Java 8, utiliza conceitos do paradigma funcional para operar em coleções de dados de maneira mais eficiente e concisa. Ela oferece operações de alto nível, como `map`, `filter` e `reduce`, que simplificam a manipulação dos elementos da coleção de forma declarativa, sem a necessidade de iteração manual. Além disso, as operações em `Stream` são geralmente imutáveis, o que significa que não modificam o estado original da coleção, tornando o código mais seguro e concorrente. A API `Stream` também suporta a composição de funções, permitindo encadear várias operações em uma única expressão para criar pipelines de processamento de dados eficientes e legíveis.

Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?

Para persistir dados em arquivos em Java, é comum seguir o padrão de desenvolvimento chamado `Object Serialization`. Este padrão envolve a serialização de objetos em bytes para armazenamento em arquivos e inclui os seguintes passos: Implementar a interface `Serializable` nas classes desejadas. Usar classes como `ObjectInputStream` e `ObjectOutputStream` para serializar e desserializar objetos para e a partir de arquivos. Lidar com exceções apropriadas ao trabalhar com operações de entrada e saída de dados. Garantir que todas as classes usadas na serialização tenham uma versão única para evitar problemas de compatibilidade. Esse padrão oferece uma maneira eficiente e flexível de persistir objetos em arquivos binários em Java.

Segue abaixo os códigos utilizados para criar o sistema de cadastro:
Logo de início, criamos um pacote chamado “`model`”, para armazenar as

entidades e os gerenciadores. No pacote model, criamos as entidades Pessoa, PessoaFisica e PessoaJuridica como pode ver abaixo:



```
Pessoa.java X
CadastroPOO > model > Pessoa.java > Pessoa > getId()
1  package _CadastroPOO.model;
2
3  import java.io.Serializable;
4
5  public class Pessoa implements Serializable {
6      private int id;
7      private String nome;
8
9      //metodo exibir
10     public void exibir() {
11         System.out.println("ID: " + id + "\nNome: " + nome);
12     }
13
14     //construtor
15     public Pessoa(int id, String nome) {
16         this.id = id;
17         this.nome = nome;
18     }
19
20     //getters e setters
21     public int getId() {
22         return id;
23     }
24
25     public void setId(int id) {
26         this.id = id;
27     }
28
29     public String getNome() {
30         return nome;
31     }
32
33     public void setNome(String nome) {
34         this.nome = nome;
35     }
36 }
```

Imagem 1 - Classe Pessoa

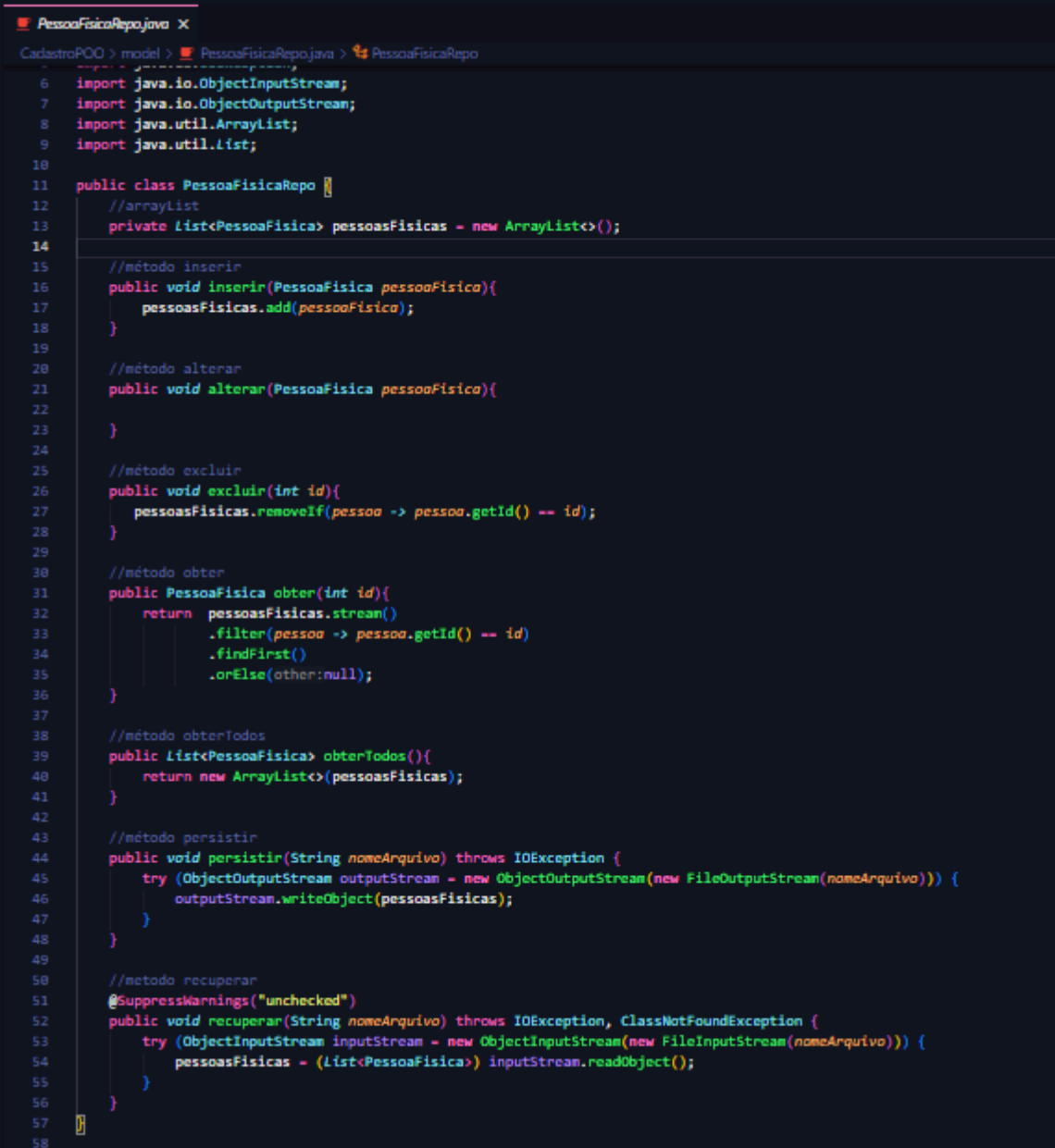


The image shows a code editor window titled "PessoaFisica.java". The code is in Java and defines a class "PessoaFisica" that extends "Pessoa". The code includes package declarations, class definition, constructor, an "exibir" method, and getters/setters for "cpf" and "idade".

```
1 package CadastroPOO.model;
2
3 public class PessoaFisica extends Pessoa{
4     private String cpf;
5     private int idade;
6
7     //construtores
8     public PessoaFisica(int idade, String cpf, String nome, int id) {
9         super(id, nome);
10        this.cpf = cpf;
11        this.idade = idade;
12    }
13
14    //metodo exibir
15    @Override
16    public void exibir() {
17        super.exibir();
18        System.out.println("CPF: " + cpf + "\nIdade: " + idade);
19    }
20
21    //getters e setters
22    public String getCpf() {
23        return cpf;
24    }
25
26    public void setCpf(String cpf) {
27        this.cpf = cpf;
28    }
29
30    public int getIdade() {
31        return idade;
32    }
33
34    public void setIdade(int idade) {
35        this.idade = idade;
36    }
37
```

Imagem 2 - Classe Pessoa Física

Após a criação das duas classes Pessoa e PessoaFisica, foi criada a parte das classes gerenciadoras com os nomes PessoaFisicaRepo e PessoaJuridicaRepo:



```
1  PessoaFisicaRepo.java X
2  CadastroPOO > model > PessoaFisicaRepo > PessoaFisicaRepo
3
4  6  import java.io.ObjectInputStream;
5  7  import java.io.ObjectOutputStream;
6  8  import java.util.ArrayList;
7  9  import java.util.List;
8
9  10
10 11 public class PessoaFisicaRepo {
11 12     //arraylist
12 13     private List<PessoaFisica> pessoasFisicas = new ArrayList<>();
13
14 14
15 15     //método inserir
16 16     public void inserir(PessoaFisica pessoaFisica){
17 17         pessoasFisicas.add(pessoaFisica);
18 18     }
19
20 19
21 20     //método alterar
22 21     public void alterar(PessoaFisica pessoaFisica){
23 22     }
24
25 24
26 25     //método excluir
27 26     public void excluir(int id){
28 27         pessoasFisicas.removeIf(pessoa -> pessoa.getId() == id);
29 28     }
30
31 29
32 30     //método obter
33 31     public PessoaFisica obter(int id){
34 32         return pessoasFisicas.stream()
35 33             .filter(pessoa -> pessoa.getId() == id)
36 34             .findFirst()
37 35             .orElse(null);
38 36     }
39
40 37
41 38     //método obterTodos
42 39     public List<PessoaFisica> obterTodos(){
43 40         return new ArrayList<>(pessoasFisicas);
44 41     }
45
46 42
47 43     //método persistir
48 44     public void persistir(String nomeArquivo) throws IOException {
49 45         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
50 46             outputStream.writeObject(pessoasFisicas);
51 47         }
52 48     }
53
54 49
55 50     //método recuperar
56 51     @SuppressWarnings("unchecked")
57 52     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
58 53         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
59 54             pessoasFisicas = (List<PessoaFisica>) inputStream.readObject();
60 55         }
61 56     }
62 57
63 58 }
```

Imagem 3 - Classe PessoaFisicaRepo

```

PessoaJuridicaRepo.java 3 x
CadastroPOO > model > PessoaJuridicaRepo.java > ...
1  package CadastroPOO.model;
2
3  import CadastroPOO.model.PessoaJuridica;
4  import java.io.FileInputStream;
5  import java.io.FileOutputStream;
6  import java.io.IOException;
7  import java.io.ObjectInputStream;
8  import java.io.ObjectOutputStream;
9  import java.util.ArrayList;
10 import java.util.List;
11
12 public class PessoaJuridicaRepo {
13     public List<PessoaJuridica> pessoasJuridicas = new ArrayList();
14
15     //metodo inserir
16     public void inserir(PessoaJuridica pessoaJuridica){
17         pessoasJuridicas.add(pessoaJuridica);
18     }
19
20     //metodo alterar
21     public void alterar(PessoaJuridica pessoaJuridica){
22     }
23
24
25     //metodo excluir
26     public void excluir(int id){
27         pessoasJuridicas.removeIf(pessoa -> pessoa.getId() == id);
28     }
29
30     //metodo obter
31     public PessoaJuridica obter(int id){
32         return pessoasJuridicas.stream()
33             .filter(pessoa -> pessoa.getId() == id)
34             .findFirst()
35             .orElse(null);
36     }
37
38     //metodo obter todos
39     public List<PessoaJuridica> obterTodos(){
40         return new ArrayList<>(pessoasJuridicas);
41     }
42
43     //metodo persistir
44     public void persistir(String nomeArquivo) throws IOException {
45         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
46             outputStream.writeObject(pessoasJuridicas);
47         }
48     }
49
50     //metodo recuperar
51     @SuppressWarnings("unchecked")
52     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
53         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
54             pessoasJuridicas = (List<PessoaJuridica>) inputStream.readObject();
55         }
56     }
57 }

```

Imagem 4 - PessoaJuridicaRepo

Logo, foi criado a classe “MainCadastro”, com cadastros já definidos para termos uma ideia que o código esteja todo funcionando sem erros como mostra abaixo o código:

```

public class MainCadastro {
    Run | Debug
    public static void main(String[] args) {
        //Repositório (repo1)
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

        // adicionando pessoas
        PessoaFisica pessoaFisica1 = new PessoaFisica(idade:1, cpf:"Ana", nome:"11111111111", id:25);
        PessoaFisica pessoaFisica2 = new PessoaFisica(idade:2, cpf:"Carlos", nome:"22222222222", id:52);

        repo1.inserir(pessoaFisica1);
        repo1.inserir(pessoaFisica2);

        try {
            //método de persistência em repo1
            repo1.persistir(nomeArquivo:"pessoasFisicas.dat");

            //Repositório (repo2).
            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

            //método de recuperação em repo2
            repo2.recuperar(nomeArquivo:"pessoasFisicas.dat");

            // Exibir os dados de todas as pessoas físicas recuperadas.
            System.out.println(x:"Dados de Pessoa Fisica Recuperados:");
            repo2.obterTodos().forEach(PessoaFisica::exibir);

            //Repositório (repo3).
            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

            //Adicionando pessoas, utilizando o construtor completo.
            PessoaJuridica pessoaJuridica1 = new PessoaJuridica(id:3, cnnpj:"XPTO Sales", nome:"3333333333333333");
            PessoaJuridica pessoaJuridica2 = new PessoaJuridica(id:4, cnnpj:"XPTO Solutions", nome:"4444444444444444");

            repo3.inserir(pessoaJuridica1);
            repo3.inserir(pessoaJuridica2);

            //método de persistência em repo3
            repo3.persistir(nomeArquivo:"pessoasJuridicas.dat");

            //Repositório (repo4).
            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

            //método de recuperação repo4
            repo4.recuperar(nomeArquivo:"pessoasJuridicas.dat");

            // Exibir os dados de todas as pessoas jurídicas recuperadas.
            System.out.println(x:"\nDados de Pessoa Juridica Recuperados:");
            repo4.obterTodos().forEach(PessoaJuridica::exibir);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}

```

Imagem 5 - MainCadastro

Temos como resultado esperado:

```

Dados de Pessoa Fisica Recuperados:
ID: 25
Nome: 11111111111
CPF: Ana
Idade: 1
ID: 52
Nome: 22222222222
CPF: Carlos
Idade: 2

Dados de Pessoa Juridica Recuperados:
ID: 3
Nome: 3333333333333333
CNPJ: XPTO Sales
ID: 4
Nome: 4444444444444444
CNPJ: XPTO Solutions
PS C:\General_projects\UNESA\terceiroPeriodo\pratica-1>

```

Imagem 6 - Resultado

2º Procedimento | Criação do Cadastro em Modo Texto

Após a conclusão do primeiro procedimento, foi criada uma nova classe que solicita ao usuário o preenchimento para obtenção das informações do cadastro, tendo como opções a criação, alteração, leitura e exclusão de cada item cadastrado. Abaixo, segue uma explicação de cada parte do processo:

Elementos estáticos

Elementos estáticos no Java são aqueles que “existem” sem a necessidade de serem instanciados. Eles estão disponíveis para uso diretamente, sem a criação de objetos. Exemplos de elementos estáticos incluem métodos estáticos e variáveis estáticas.

Por que o método main é estático?

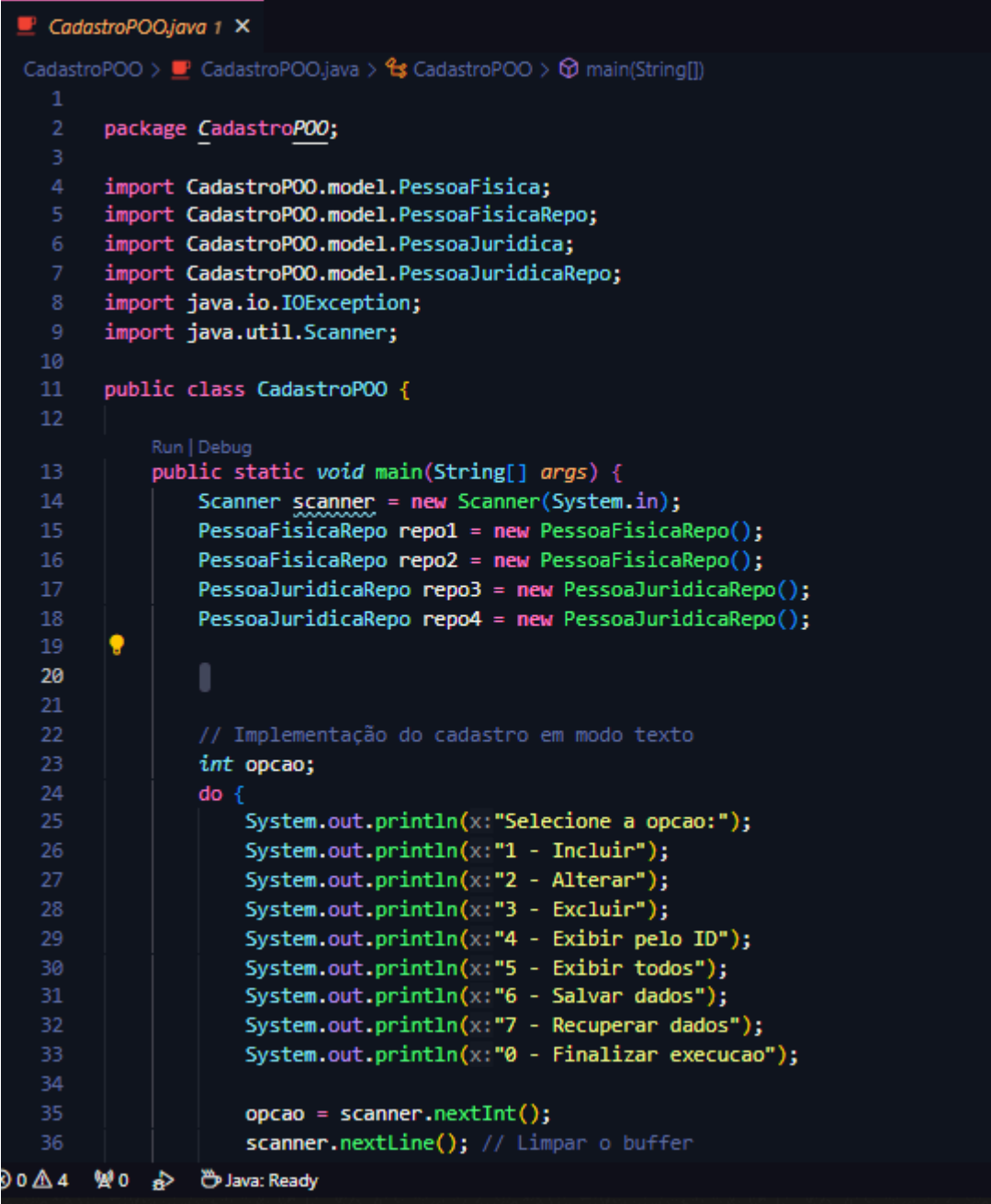
O método “main” é declarado como estático porque é o ponto de entrada de um programa Java e precisa ser invocado pelo sistema Java antes da criação de qualquer instância da classe que o contém.

Classe Scanner

A classe Scanner no Java é utilizada para facilitar a entrada de dados a partir do console. Ela permite ler valores de diferentes tipos primitivos (inteiros, números de ponto flutuante, strings etc.) a partir do teclado.

Impacto das Classes de Repositório na Organização do Código

O uso de classes de repositório, como `PessoaFisicaRepo` e `PessoaJuridicaRepo`, proporcionou uma organização mais modular e clara do código. Essas classes encapsulam a lógica de gerenciamento de dados relacionados a entidades específicas, separando a lógica de negócios da lógica de persistência. Isso facilita a manutenção do código, facilita adicionar novas funcionalidades e promove uma organização mais eficiente dos dados.



```
1
2 package CadastroPOO;
3
4 import CadastroPOO.model.PessoaFisica;
5 import CadastroPOO.model.PessoaFisicaRepo;
6 import CadastroPOO.model.PessoaJuridica;
7 import CadastroPOO.model.PessoaJuridicaRepo;
8 import java.io.IOException;
9 import java.util.Scanner;
10
11 public class CadastroPOO {
12
13     public static void main(String[] args) {
14         Scanner scanner = new Scanner(System.in);
15         PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
16         PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
17         PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
18         PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
19
20         // Implementação do cadastro em modo texto
21         int opcao;
22         do {
23             System.out.println("Selecione a opcao:");
24             System.out.println("1 - Incluir");
25             System.out.println("2 - Alterar");
26             System.out.println("3 - Excluir");
27             System.out.println("4 - Exibir pelo ID");
28             System.out.println("5 - Exibir todos");
29             System.out.println("6 - Salvar dados");
30             System.out.println("7 - Recuperar dados");
31             System.out.println("0 - Finalizar execucao");
32
33             opcao = scanner.nextInt();
34             scanner.nextLine(); // Limpar o buffer
35         } while (opcao != 0);
36     }
37 }
```

Imagem 7 CadastroPOO (Parte 1)

```

opcao = scanner.nextInt();
scanner.nextLine(); // Limpar o buffer

switch (opcao) {
    case 1:
        System.out.println(x:"Escolha o tipo (F -> Fisica ou J -> Juridica):");
        String tipo = scanner.nextLine();
        if (tipo.equalsIgnoreCase(anotherString:"F")) {
            System.out.println(x:"Digite o ID:");
            int id = scanner.nextInt();
            scanner.nextLine(); // Limpar o buffer
            System.out.println(x:"Digite o nome:");
            String nome = scanner.nextLine();
            System.out.println(x:"Digite o CPF:");
            String cpf = scanner.nextLine();
            System.out.println(x:"Digite a idade:");
            int idade = scanner.nextInt();
            scanner.nextLine(); // Limpar o buffer
            repo1.inserir(new PessoaFisica(id, nome, cpf, idade));
        } else if (tipo.equalsIgnoreCase(anotherString:"J")) {
            System.out.println(x:"Digite o ID:");
            int id = scanner.nextInt();
            scanner.nextLine(); // Limpar o buffer
            System.out.println(x:"Digite o nome:");
            String nome = scanner.nextLine();
            System.out.println(x:"Digite o CNPJ:");
            String cnpj = scanner.nextLine();
            repo3.inserir(new PessoaJuridica(id, nome, cnpj));
        } else {
            System.out.println(x:"Tipo inválido.");
        }
    }
}

```

Imagem 8 - CadastroPOO (Parte 2)

```

CadastroPOO > CadastroPOO.java > CadastroPOO > main(String[])
11 public class CadastroPOO {
13     public static void main(String[] args) {
62         repos.inserir(new PessoaJuridica(id, nome, cnpj));
63     } else {
64         System.out.println(x:"Tipo inválido.");
65     }
66     break;
67     case 2:
68         System.out.println(x:"Escolha o tipo (F -> Fisica ou J -> Juridica:");
69         tipo = scanner.nextLine();
70         System.out.println(x:"Digite o ID:");
71         int id = scanner.nextInt();
72         scanner.nextLine(); // Limpar o buffer
73         if (tipo.equalsIgnoreCase(anotherString:"F")) {
74             PessoaFisica pessoaFisica = repo1.obter(id);
75             if (pessoaFisica != null) {
76                 System.out.println(x:"Dados atuais:");
77                 pessoaFisica.exibir();
78                 System.out.println(x:"Digite o novo nome:");
79                 String novoNome = scanner.nextLine();
80                 pessoaFisica.setNome(novoNome);
81                 System.out.println(x:"Digite o novo CPF:");
82                 String novoCpf = scanner.nextLine();
83                 pessoaFisica.setCpf(novoCpf);
84                 System.out.println(x:"Digite a nova idade:");
85                 int novaIdade = scanner.nextInt();
86                 scanner.nextLine(); // Limpar o buffer
87                 pessoaFisica.setIdade(novaIdade);
88                 repo1.alterar(pessoaFisica);
89                 System.out.println(x:"Pessoa fisica alterada com sucesso.");
90             } else {
91                 System.out.println(x:"Pessoa fisica não encontrada.");
92             }
93         } else if (tipo.equalsIgnoreCase(anotherString:"J")) {
94             PessoaJuridica pessoaJuridica = repo3.obter(id);
95             if (pessoaJuridica != null) {
96                 System.out.println(x:"Dados atuais:");

```

```
CadastroPOO.java 1 X
CadastroPOO > CadastroPOO.java > CadastroPOO > main(String[])
11 public class CadastroPOO {
13     public static void main(String[] args) {
14
15         PessoaJuridica pessoaJuridica = repo3.obter(id);
16         if (pessoaJuridica != null) {
17             System.out.println(x:"Dados atuais:");
18             pessoaJuridica.exibir();
19             System.out.println(x:"Digite o novo nome:");
20             String novoNome = scanner.nextLine();
21             pessoaJuridica.setNome(novoNome);
22             System.out.println(x:"Digite o novo CNPJ:");
23             String novoCnpj = scanner.nextLine();
24             pessoaJuridica.setCnpj(novoCnpj);
25             repo3.alterar(pessoaJuridica);
26             System.out.println(x:"Pessoa juridica alterada com sucesso.");
27         } else {
28             System.out.println(x:"Pessoa juridica não encontrada.");
29         }
30     } else {
31         System.out.println(x:"Tipo invalido.");
32     }
33     break;
34 case 3:
35     System.out.println(x:"Escolha o tipo (F -> Fisica ou J -> Juridica):");
36     tipo = scanner.nextLine();
37     System.out.println(x:"Digite o ID:");
38     id = scanner.nextInt();
39     scanner.nextLine(); // Limpar o buffer
40     if (tipo.equalsIgnoreCase(anotherString:"F")) {
41         repo1.excluir(id);
42         System.out.println(x:"Pessoa fisica excluida com sucesso.");
43     } else if (tipo.equalsIgnoreCase(anotherString:"J")) {
44         repo3.excluir(id);
45         System.out.println(x:"Pessoa juridica excluida com sucesso.");
46     } else {
47         System.out.println(x:"Tipo invalido.");
48     }
49     break;
50 }
```

```
CadastroPOO.java X
CadastroPOO > CadastroPOO.java > CadastroPOO > main(String[])
11 public class CadastroPOO {
13     public static void main(String[] args) {
14         break;
129     case 4:
130         System.out.println(x:"Escolha o tipo (F -> Fisica ou J -> Juridica):");
131         tipo = scanner.nextLine();
132         System.out.println(x:"Digite o ID:");
133         id = scanner.nextInt();
134         scanner.nextLine(); // Limpar o buffer
135         if (tipo.equalsIgnoreCase(anotherString:"F")) {
136             PessoaFisica pessoaFisica = repo1.obter(id);
137             if (pessoaFisica != null) {
138                 System.out.println(x:"Dados da pessoa fisica:");
139                 pessoaFisica.exibir();
140             } else {
141                 System.out.println(x:"Pessoa fisica nao encontrada.");
142             }
143         } else if (tipo.equalsIgnoreCase(anotherString:"J")) {
144             PessoaJuridica pessoaJuridica = repo3.obter(id);
145             if (pessoaJuridica != null) {
146                 System.out.println(x:"Dados da pessoa juridica:");
147                 pessoaJuridica.exibir();
148             } else {
149                 System.out.println(x:"Pessoa juridica não encontrada.");
150             }
151         } else {
152             System.out.println(x:"Tipo invalido.");
153         }
154         break;
155     case 5:
156         System.out.println(x:"Escolha o tipo (F -> Fisica ou J -> Juridica):");
157         tipo = scanner.nextLine();
158         if (tipo.equalsIgnoreCase(anotherString:"F")) {
159             System.out.println(x:"Pessoas fisicas cadastradas:");
160             for (PessoaFisica pessoa : repo1.obterTodos()) {
161                 pessoa.exibir();
162                 System.out.println();
163             }
164         } else if (tipo.equalsIgnoreCase(anotherString:"J")) {
165             System.out.println(x:"Pessoas juridicas cadastradas:");
166             for (PessoaJuridica pessoa : repo3.obterTodos()) {
167                 pessoa.exibir();
168                 System.out.println();
169             }
170         } else {
171             System.out.println(x:"Tipo invalido.");
172         }
173     }
```

Após a execução do código temos o seguinte resultado:

```
Selecione a opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Finalizar execucao
1
Escolha o tipo (F -> Fisica ou J -> Juridica):
f
Digite o ID:
1290
Digite o nome:
Luiz Fabrício
Digite o CPF:
3125479301
Digite a idade:
19
```

```
Digite a idade:
19
Selecione a opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Finalizar execucao
6
Dados salvos com sucesso.
Selecione a opcao:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Finalizar execucao
5
Escolha o tipo (F -> Fisica ou J -> Juridica):
F
Pessoas fisicas cadastradas:
ID: 19
Nome: 3125479301
CPF: Luiz Fabr?cio
Idade: 1290
```

Com isso, finalizamos todos os procedimentos na criação de cadastro.

Para realização desta atividade, foi utilizada a IDE Visual Studio Code, com a extensão “Extension Pack for Java” e o Java JDK Zulu 21.32.17