



Endereço do Campus/Polo:

Rua Moraes E Silva 40 - Maracanã - Rio de Janeiro - RJ - CEP:
20.271-030

Curso: Desenvolvimento Full Stack

Disciplina: Iniciando o Caminho Pelo Java

Turma: 9001

Semestre Letivo: Terceiro

Nome: Matheus Zimmer Moreira Martins

Endereço do Repositório:

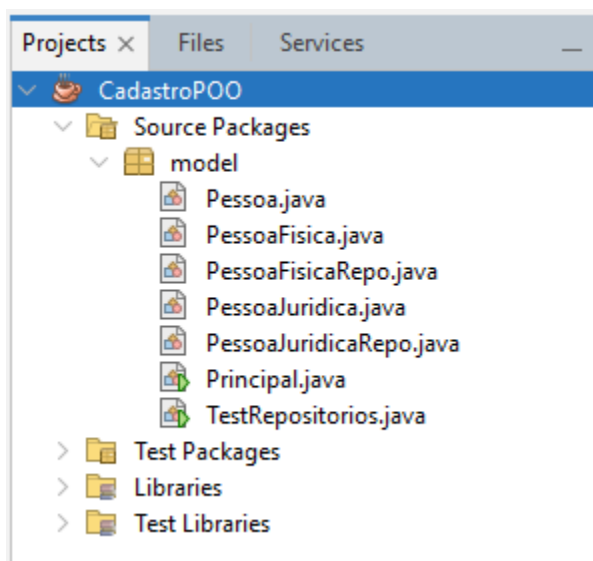
<https://github.com/Mazimo-Marts/Trabalho-Mundo3-Nvl1>

Criação das Entidades e Sistema de Persistência

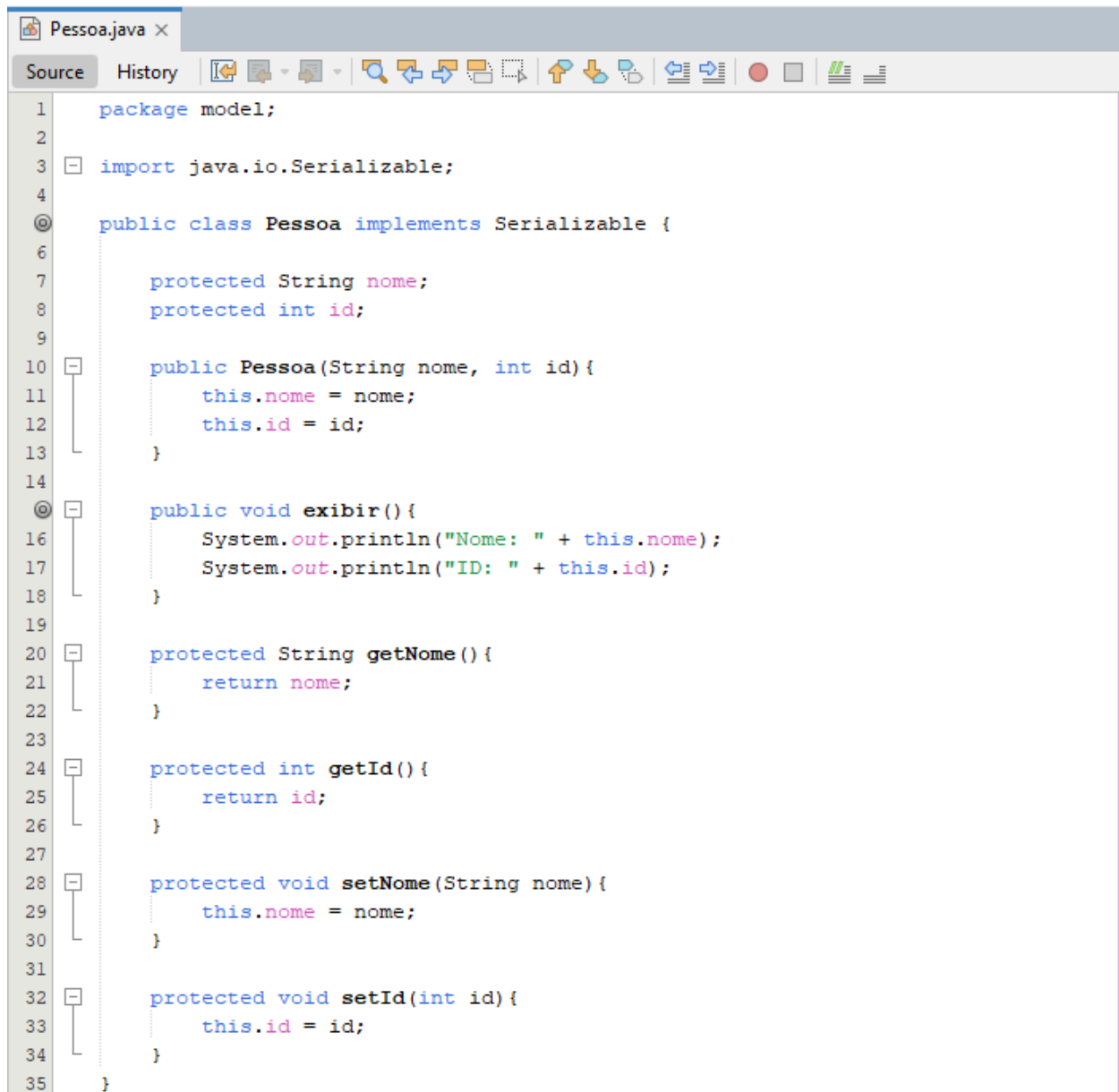
Objetivo:

Utilizar herança e polimorfismo na definição de entidades, utilizar persistência de objetos em arquivos binários, implementar uma interface cadastral em modo texto e utilizar o controle de exceções da plataforma java.

Códigos da Prática:

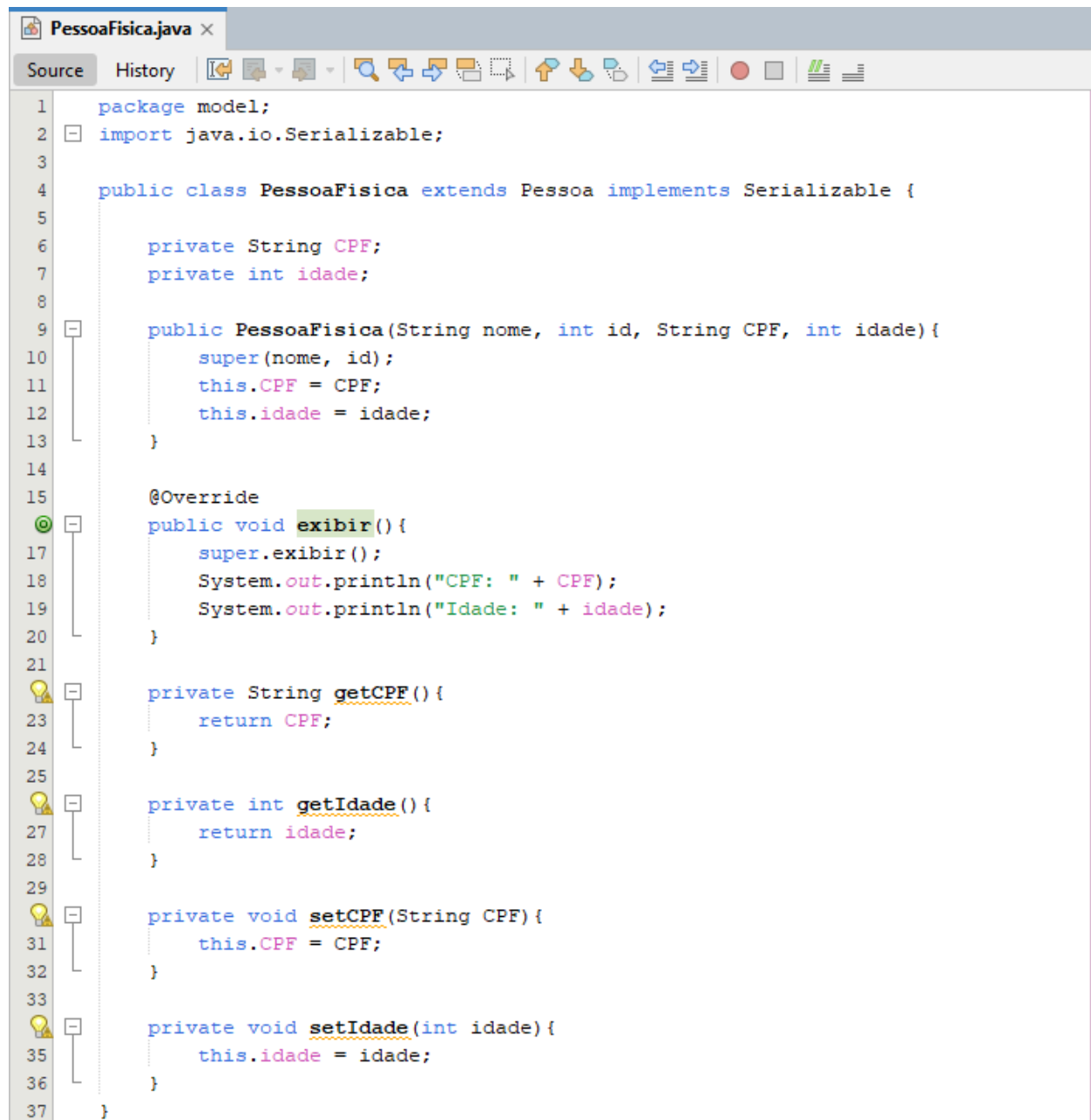


Classe Pessoa:



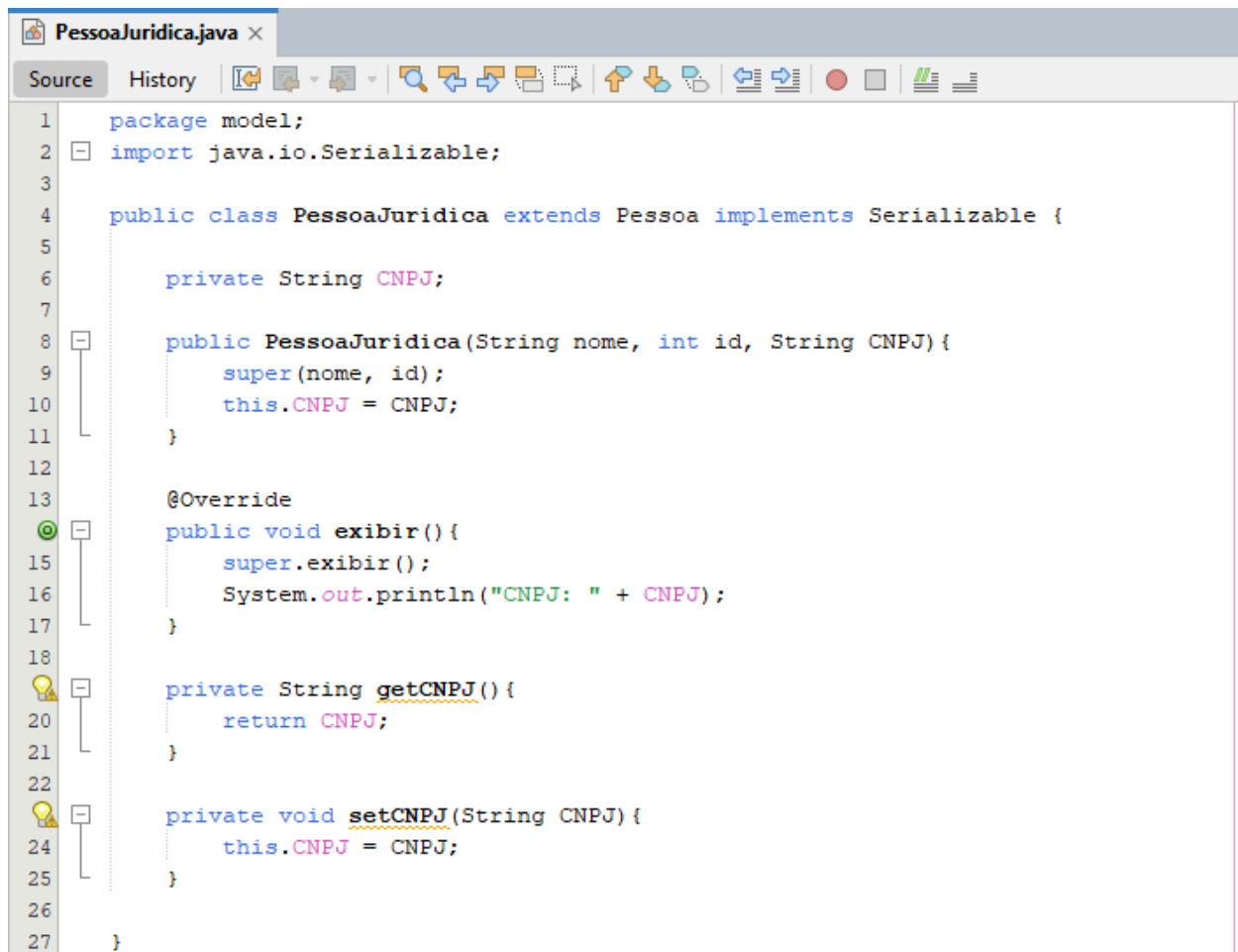
```
1 package model;
2
3 import java.io.Serializable;
4
5 public class Pessoa implements Serializable {
6
7     protected String nome;
8     protected int id;
9
10    public Pessoa(String nome, int id){
11        this.nome = nome;
12        this.id = id;
13    }
14
15    public void exibir(){
16    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

Classe Pessoa Física:



```
1 package model;
2 import java.io.Serializable;
3
4 public class PessoaFisica extends Pessoa implements Serializable {
5
6     private String CPF;
7     private int idade;
8
9     public PessoaFisica(String nome, int id, String CPF, int idade) {
10         super(nome, id);
11         this.CPF = CPF;
12         this.idade = idade;
13     }
14
15     @Override
16     public void exibir() {
17         super.exibir();
18         System.out.println("CPF: " + CPF);
19         System.out.println("Idade: " + idade);
20     }
21
22     private String getCPF() {
23         return CPF;
24     }
25
26     private int getIdade() {
27         return idade;
28     }
29
30     private void setCPF(String CPF) {
31         this.CPF = CPF;
32     }
33
34     private void setIdade(int idade) {
35         this.idade = idade;
36     }
37 }
```

Classe Pessoa Jurídica:



```
1 package model;
2 import java.io.Serializable;
3
4 public class PessoaJuridica extends Pessoa implements Serializable {
5
6     private String CNPJ;
7
8     public PessoaJuridica(String nome, int id, String CNPJ) {
9         super(nome, id);
10        this.CNPJ = CNPJ;
11    }
12
13    @Override
14    public void exhibir() {
15        super.exibir();
16        System.out.println("CNPJ: " + CNPJ);
17    }
18
19    private String getCNPJ() {
20        return CNPJ;
21    }
22
23    private void setCNPJ(String CNPJ) {
24        this.CNPJ = CNPJ;
25    }
26
27 }
```

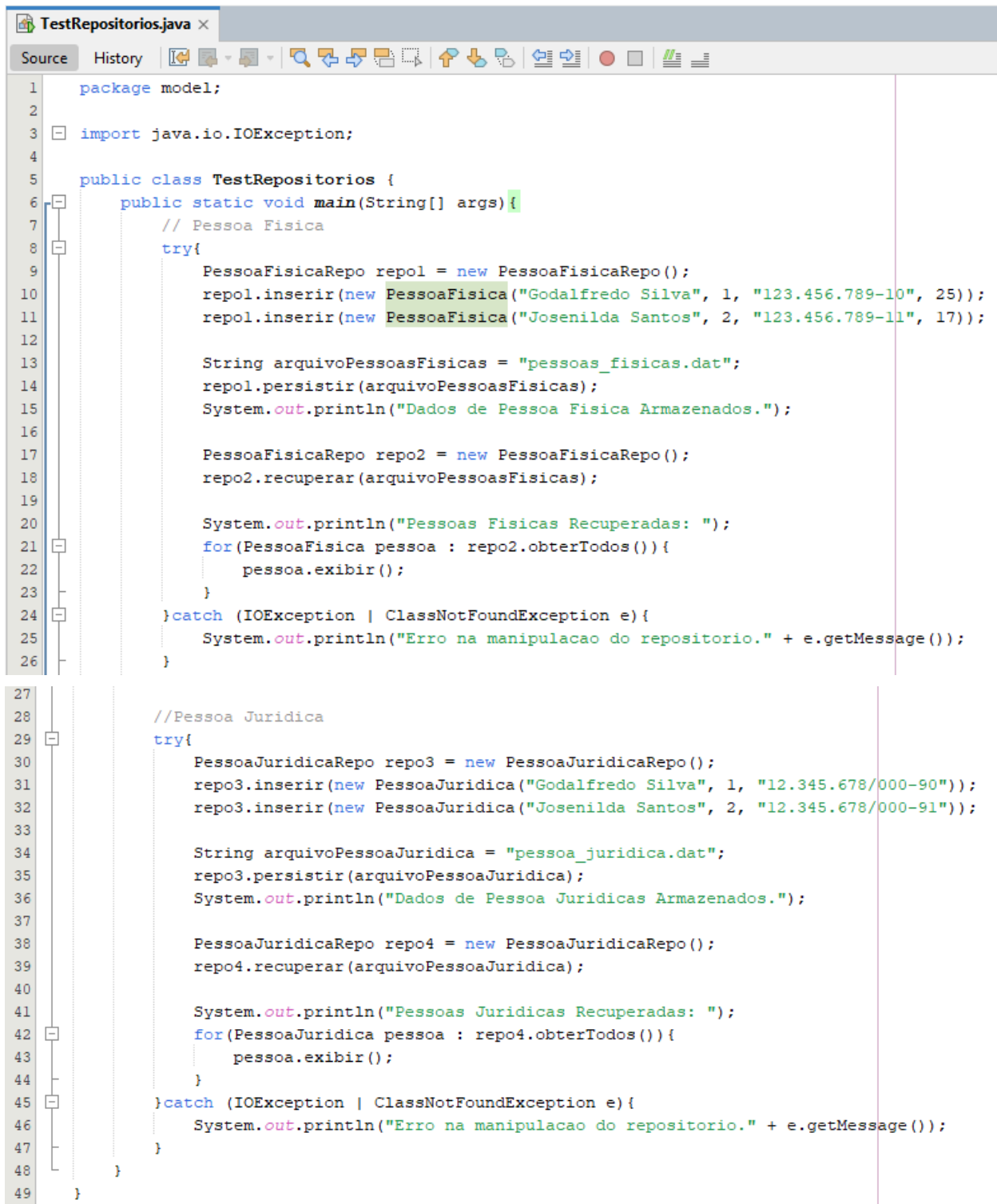
Classe Pessoa Física Repo:

```
PessoaFisicaRepo.java x
Source History
1 package model;
2
3 + import ...6 lines
9
10 public class PessoaFisicaRepo {
11
12     private ArrayList<PessoaFisica> List = new ArrayList<>();
13
14     public void inserir(PessoaFisica pessoaF) {
15         List.add(pessoaF);
16     }
17
18     public void alterar(PessoaFisica pessoaF) {
19         for(int i=0; i < List.size(); i++){
20             if(List.get(i).getId() == pessoaF.getId()){
21                 List.set(i, pessoaF);
22                 break;
23             }
24         }
25     }
26
27     public void excluir(int id) {
28         List.removeIf(pessoaF -> pessoaF.getId() == id);
29     }
30
31     public PessoaFisica obter(int id) {
32         for(PessoaFisica pessoaF : List) {
33             if(pessoaF.getId() == id) {
34                 return pessoaF;
35             }
36         }
37         return null;
38     }
39
40     public ArrayList<PessoaFisica> obterTodos() {
41         return List;
42     }
43
44     public void persistir(String nomeArquivo) throws IOException {
45         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
46             outputStream.writeObject(List);
47         }
48     }
49
50     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
51         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
52             List = (ArrayList<PessoaFisica>) inputStream.readObject();
53         }
54     }
55 }
```

Classe Pessoa Jurídica Repo:

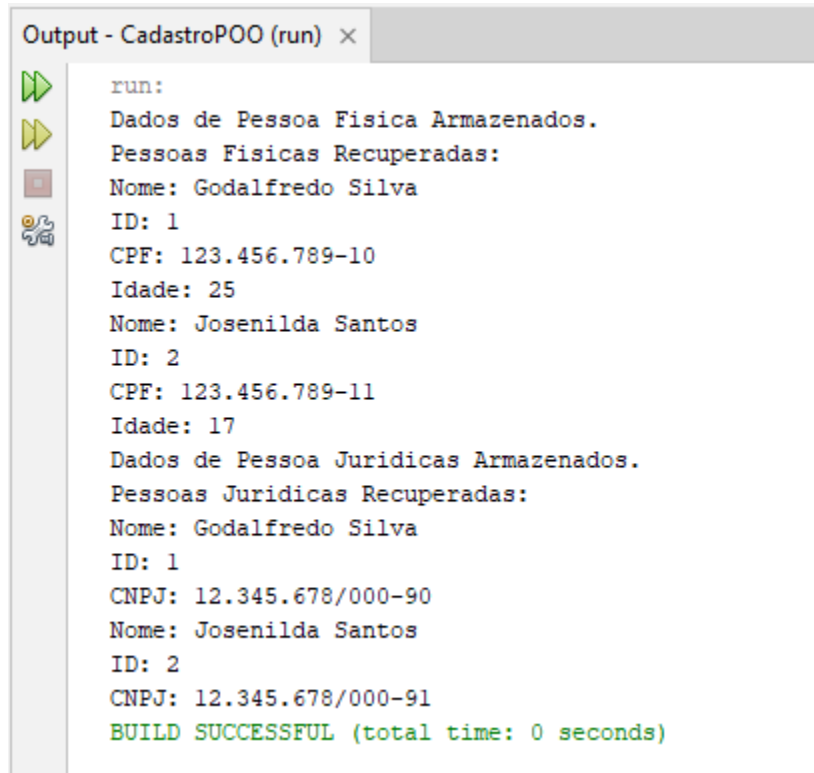
```
PessoaJuridicaRepo.java x
Source History
1 package model;
2
3 + import ...6 lines
9
10 public class PessoaJuridicaRepo {
11
12     private ArrayList<PessoaJuridica> List = new ArrayList<>();
13
14     public void inserir(PessoaJuridica pessoaJ) {
15         List.add(pessoaJ);
16     }
17
18     public void alterar(PessoaJuridica pessoaJ) {
19         for(int i=0; i < List.size(); i++){
20             if(List.get(i).getId() == pessoaJ.getId()){
21                 List.set(i, pessoaJ);
22                 break;
23             }
24         }
25     }
26
27     public void excluir(int id) {
28         List.removeIf(pessoaJ -> pessoaJ.getId() == id);
29     }
30
31     public PessoaJuridica obter(int id) {
32         for(PessoaJuridica pessoaJ : List) {
33             if(pessoaJ.getId() == id) {
34                 return pessoaJ;
35             }
36         }
37         return null;
38     }
39
40     public ArrayList<PessoaJuridica> obterTodos() {
41         return List;
42     }
43
44     public void persistir(String nomeArquivo) throws IOException {
45         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
46             outputStream.writeObject(List);
47         }
48     }
49
50     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
51         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
52             List = (ArrayList<PessoaJuridica>) inputStream.readObject();
53         }
54     }
55 }
56
```

Teste Repositórios:



```
1 package model;
2
3 import java.io.IOException;
4
5 public class TestRepositorios {
6     public static void main(String[] args){
7         // Pessoa Fisica
8         try{
9             PessoaFisicaRepo repol = new PessoaFisicaRepo();
10            repol.inserir(new PessoaFisica("Godalfredo Silva", 1, "123.456.789-10", 25));
11            repol.inserir(new PessoaFisica("Josenilda Santos", 2, "123.456.789-11", 17));
12
13            String arquivoPessoasFisicas = "pessoas_fisicas.dat";
14            repol.persistir(arquivoPessoasFisicas);
15            System.out.println("Dados de Pessoa Fisica Armazenados.");
16
17            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
18            repo2.recuperar(arquivoPessoasFisicas);
19
20            System.out.println("Pessoas Fisicas Recuperadas: ");
21            for(PessoaFisica pessoa : repo2.obterTodos()){
22                pessoa.exibir();
23            }
24        }catch (IOException | ClassNotFoundException e){
25            System.out.println("Erro na manipulacao do repositorio." + e.getMessage());
26        }
27
28        //Pessoa Juridica
29        try{
30            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
31            repo3.inserir(new PessoaJuridica("Godalfredo Silva", 1, "12.345.678/000-90"));
32            repo3.inserir(new PessoaJuridica("Josenilda Santos", 2, "12.345.678/000-91"));
33
34            String arquivoPessoaJuridica = "pessoa_juridica.dat";
35            repo3.persistir(arquivoPessoaJuridica);
36            System.out.println("Dados de Pessoa Juridicas Armazenados.");
37
38            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
39            repo4.recuperar(arquivoPessoaJuridica);
40
41            System.out.println("Pessoas Juridicas Recuperadas: ");
42            for(PessoaJuridica pessoa : repo4.obterTodos()){
43                pessoa.exibir();
44            }
45        }catch (IOException | ClassNotFoundException e){
46            System.out.println("Erro na manipulacao do repositorio." + e.getMessage());
47        }
48    }
49 }
```


Resultado da execução:



```
run:
Dados de Pessoa Fisica Armazenados.
Pessoas Fisicas Recuperadas:
Nome: Godalfredo Silva
ID: 1
CPF: 123.456.789-10
Idade: 25
Nome: Josenilda Santos
ID: 2
CPF: 123.456.789-11
Idade: 17
Dados de Pessoa Juridicas Armazenados.
Pessoas Juridicas Recuperadas:
Nome: Godalfredo Silva
ID: 1
CNPJ: 12.345.678/000-90
Nome: Josenilda Santos
ID: 2
CNPJ: 12.345.678/000-91
BUILD SUCCESSFUL (total time: 0 seconds)
```

Análise

a)Quais as vantagens e desvantagens do uso de herança?

Vantagens: Possibilita a reutilização de códigos, facilita a extensão de comportamentos existentes sem modificar o código original, cria uma estrutura hierárquica clara entre classes, permitindo operações polimórficas e referências a objetos por suas superclasses e facilita a organização do código.

Desvantagens: Cria uma dependência forte entre classes pai e filhas, mudanças na classe pai podem afetar todas as subclasses, o Java não suporta herança múltipla de classes, apenas de interfaces, limitando alguns designs, hierarquias profundas de herança podem tornar o código difícil de entender e manter.

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

A interface Serializable é um marcador que indica que uma classe pode ser convertida em bytes para ser armazenada ou transmitida. Sem ela, o Java não sabe como serializar os objetos, resultando em erros durante a persistência.

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

A API stream do Java utiliza vários princípios do paradigma funcional, representando uma abordagem híbrida que incorpora conceitos funcionais em uma linguagem orientada a objetos, utilizando conceitos como:

- Funções de Primeira Classe
- Imutabilidade
- Composição de Funções
- Expressões Lambda
- Paralelismo sem Estado

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

Existem vários padrões, nós utilizamos o padrão de “Serialização de objetos Java”, ele utiliza a interface `Serializable` e permite converter objetos em sequências de bytes e vice-versa que é útil para armazenar objetos completos em arquivos.