 Estácio	Universidade Estácio Campus Caieiras Curso de Desenvolvimento Full Stack Relatório da Missão Prática 1 - Mundo 3
Disciplina:	RPG0014 - Iniciando o caminho pelo Java
Nome:	Felipe Rocha Santana da Silva
Turma:	2023.1

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

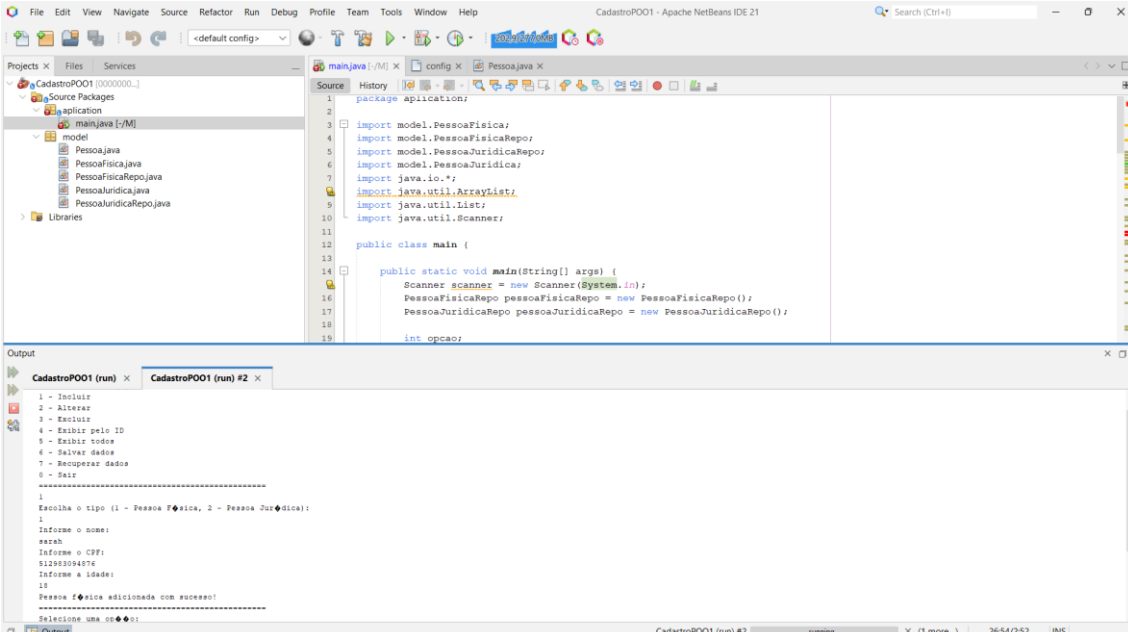
1. Título da Prática: “1º Procedimento | Criação das Entidades e Sistema de Persistência”

2. Objetivo da Prática

- Utilizar herança e polimorfismo na definição de entidades.
- Utilizar persistência de objetos em arquivos binários.
- Implementar uma interface cadastral em modo texto.
- Utilizar o controle de exceções da plataforma Java.
- Implementar um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

3. Códigos solicitados: anexo no final do relatório.

4. Resultados da execução dos códigos.



The screenshot displays the Apache NetBeans IDE environment. The top toolbar includes standard development tools like File, Edit, View, and Run. The left sidebar shows the project structure for 'CadastroPOO1', with a focus on the 'model' package containing classes like 'PessoaFisica' and 'PessoaJuridica'. The main editor window shows the source code of the 'main' class, which imports necessary packages and sets up a scanner for user input. The bottom 'Output' window shows the execution results, including a menu for selecting entity types and a successful message for adding a new physical person.

```

1 package application;
2
3 import model.PessoaFisica;
4 import model.PessoaFisicaRepo;
5 import model.PessoaJuridicaRepo;
6 import model.PessoaJuridica;
7 import java.io.*;
8 import java.util.ArrayList;
9 import java.util.List;
10 import java.util.Scanner;
11
12 public class main {
13
14     public static void main(String[] args) {
15         Scanner scanner = new Scanner(System.in);
16         PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
17         PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();
18
19         int opcao;

```

Output:

```

1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
8 - Sair
=====
1 Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Informe o nome:
sarah
Informe o CPF:
512953094076
Informe a idade:
18
Pessoa Física adicionada com sucesso!
=====
Selecione uma opção:

```

5. Análise e Conclusão

A. Quais as vantagens e desvantagens do uso de herança?

R: As vantagens são reutilizar códigos já existente, o que economiza tempo e deixa o código mais organizado.

E as desvantagens, algumas classes ficam meio instáveis e difícil de se manter, se uma classe pai muda, todos os filhos precisam ser atualizados também.

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

R: A interface Serializable é tipo um passaporte que diz pro Java que um objeto pode ser convertido em uma sequência de bytes, o que é essencial pra gravar o objeto em um arquivo binário. Então, resumindo, a interface Serializable é tipo a chave que abre a porta pra salvar e carregar objetos em arquivos binários no Java. Sem ela, o Java fica meio perdido e não consegue fazer essa mágica acontecer.

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

R: A API Stream faz uso de expressões lambda, que são tipo mini funções que a gente pode passar como parâmetros. Isso dá uma flexibilidade enorme pra manipular os dados da coleção do jeito que a gente quiser. Então, em vez de usar loops tradicionais, a gente pode usar essas operações funcionais da API Stream pra processar os dados de forma mais declarativa e concisa, o que deixa o código mais elegante e fácil de entender.

Então, em vez de usar loops tradicionais, a gente pode usar essas operações funcionais da API Stream pra processar os dados de forma mais declarativa e concisa, o que deixa o código mais elegante e fácil de entender.

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

R: Quando a gente fala de persistência de dados em arquivos com Java, o padrão de desenvolvimento mais comum é o padrão de projeto DAO, que significa "Data Access Object".

Esse padrão separa a lógica de acesso aos dados da lógica de negócios da aplicação. Basicamente, a ideia é ter uma classe (ou interface) para cada tipo de entidade que a gente quer persistir, tipo ClienteDAO, ProdutoDAO.

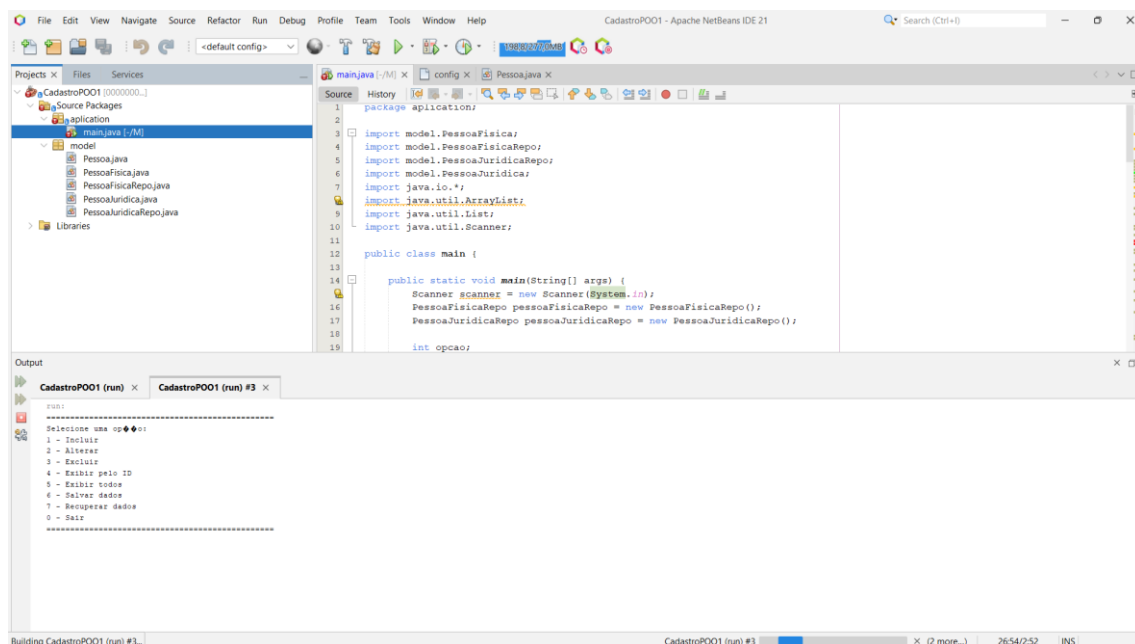
1 Título da Prática: “2º Procedimento | Criação do Cadastro em Modo Texto”

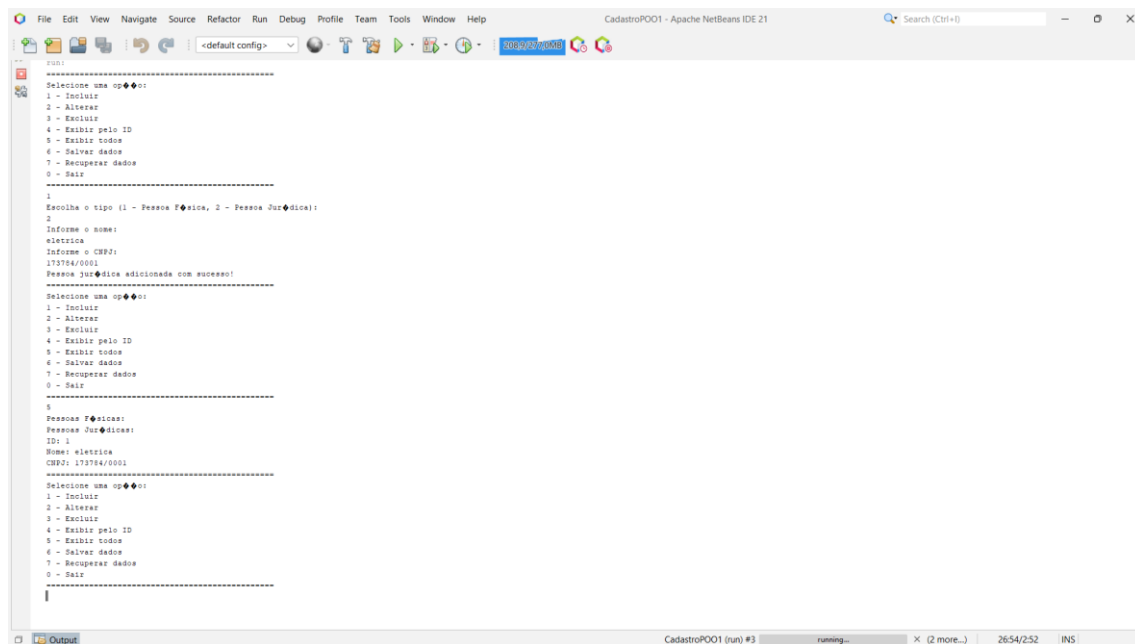
2. Objetivo da Prática

- Utilizar herança e polimorfismo na definição de entidades.
- Utilizar persistência de objetos em arquivos binários.
- Implementar uma interface cadastral em modo texto.
- Utilizar o controle de exceções da plataforma Java.
- Implementar um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

3. Códigos solicitados: <https://github.com/feeeehhhh/CadastroPoo>

4. Resultados da execução dos códigos





Anexo I: códigos do projeto

```
package model;

import java.io.Serializable;
import java.rmi.server.UID;
import java.util.UUID;

public class Pessoa implements Serializable {

    private static int contador = 0;
    private int id;
    private String nome;

    //Métodos

    public Pessoa(String nome){
        this.setNome(nome);
    }
}
```

```
        contador++;
        id = contador;

    }
    public void exibirInformacoes() {
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
    }
    private void setNome(String nome){
        this.nome = nome;
    }
    public String getNome(){
        return this.nome;
    }

    public int getId(){
        return this.id;
    }

}
```

```
package model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements
Serializable{

    private String cpf;
    private int idade;

    public PessoaFisica (String nome, String cpf, int
idade){
```

```

        super(nome);
        this.cpf = cpf;
        this.idade = idade;
    }

    private void setCpf(String cpf){
        this.cpf = cpf;
    }

    public String getCpf(){
        return cpf;
    }

    private void setIdade(int idade){
        this.idade = idade;
    }
    public int getIdade(){
        return idade;
    }

    @Override
    public void exibirInformacoes() {
        super.exibirInformacoes();
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
    }
}

```

```

package model;

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

```

```
public class PessoaFisicaRepo implements Serializable {

    private List<PessoaFisica> pessoasFisicas;

    public PessoaFisicaRepo() {
        this.pessoasFisicas = new ArrayList<>();
    }

    public void inserir(PessoaFisica pessoaFisica) {
        this.pessoasFisicas.add(pessoaFisica);
    }

    public void alterar(PessoaFisica pessoaFisica) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(i).getId() ==
pessoaFisica.getId()) {
                pessoasFisicas.set(i, pessoaFisica);
                return;
            }
        }
    }

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

    public PessoaFisica obter(int id) {
        for (PessoaFisica pessoaFisica : pessoasFisicas)
        {
            if (pessoaFisica.getId() == id) {
                return pessoaFisica;
            }
        }
        return null;
    }

    public List<PessoaFisica> obterTodos() {
```

```

        return this.pessoasFisicas;
    }

    public void persistir(String nomeArquivo) throws
IOException {
        try (ObjectOutputStream out = new
ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            out.writeObject(this.pessoasFisicas);
        } catch (IOException e) {
            throw e;
        }
    }

    public void recuperar(String nomeArquivo) throws
IOException, ClassNotFoundException {
        try (ObjectInputStream in = new
ObjectInputStream(new FileInputStream(nomeArquivo))) {
            this.pessoasFisicas = (List<PessoaFisica>)
in.readObject();
        } catch (IOException e) {
            throw e;
        } catch (ClassNotFoundException e) {
            throw e;
        }
    }
}

```

```

package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements
Serializable{

```



```

private String cnpj;

public PessoaJuridica(String nome,String cnpj){
    super(nome);
    this.cnpj=cnpj;
}
private void setCnpj(String cnpj){
    this.cnpj=cnpj;
}
public String getCnpj(){
    return cnpj;
}
@Override
public void exibirInformacoes() {
    super.exibirInformacoes();
    System.out.println("CNPJ: " + cnpj);
}
}

```

```

package model;

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

public class PessoaJuridicaRepo implements Serializable {
    private List<PessoaJuridica> pessoasJuridicas;

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

    public void inserir(PessoaJuridica pessoaJuridica) {

```

```

        this.pessoasJuridicas.add(pessoaJuridica);
    }

    public void alterar(PessoaJuridica pessoaJuridica) {
        for (int i = 0; i < pessoasJuridicas.size(); i++)
        {
            if (pessoasJuridicas.get(i).getId() ==
pessoaJuridica.getId()) {
                pessoasJuridicas.set(i, pessoaJuridica);
                return;
            }
        }
    }

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

    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pessoaJuridica :
pessoasJuridicas) {
            if (pessoaJuridica.getId() == id) {
                return pessoaJuridica;
            }
        }
        return null;
    }

    public List<PessoaJuridica> obterTodos() {
        return this.pessoasJuridicas;
    }

    public void persistir(String nomeArquivo) throws
IOException {
        try (ObjectOutputStream out = new
ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            out.writeObject(this.pessoasJuridicas);
        }
    }

```

```
        } catch (IOException e) {
            throw e;
        }
    }

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