 Estácio	Universidade Estácio Campus Belém Curso de Desenvolvimento Full Stack Relatório da Missão Prática 1 - Mundo 3
Disciplina:	RPG0014 - Iniciando o caminho pelo Java
Nome:	Cleyton Isamu Muto
Turma:	2023.1

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

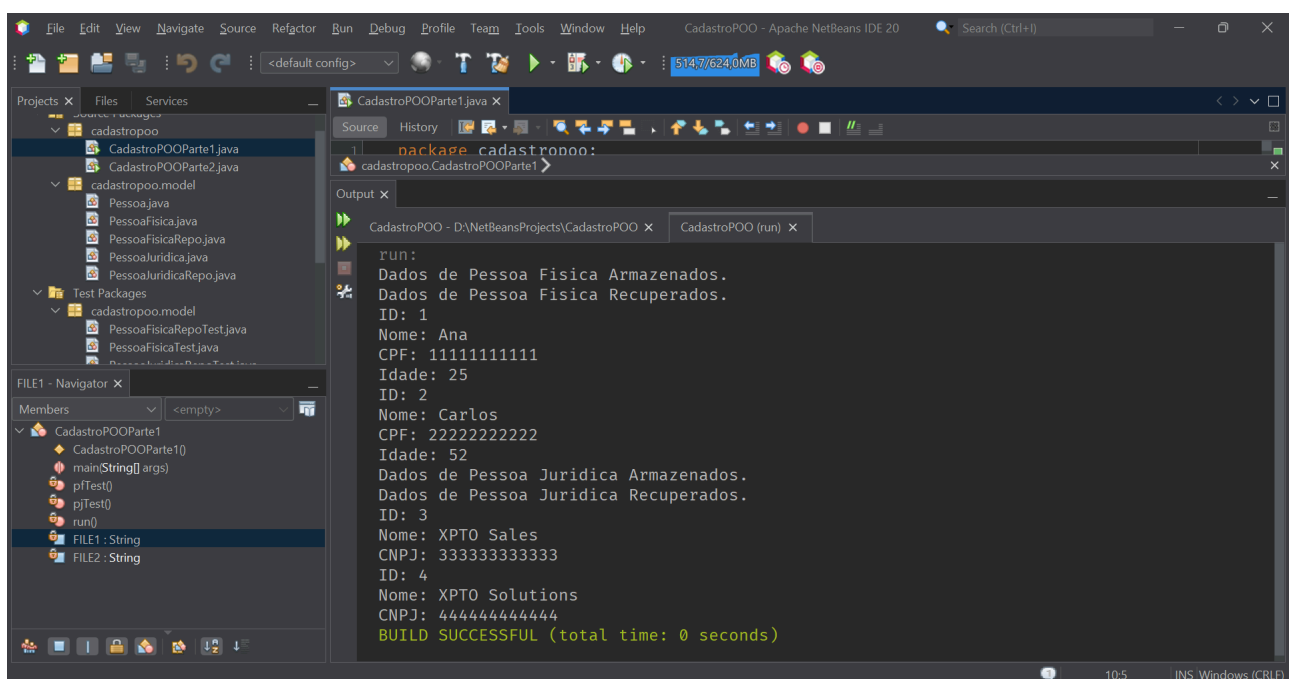
1. Título da Prática: “1º Procedimento | Criação das Entidades e Sistema de Persistência”

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



```

package cadastrapoo;

castrapoo.CadastroPOOParte1

Output
CadastrarPOO - D:\NetBeansProjects\CadastrarPOO \ CadastrarPOO (run) x
Run:
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados.
ID: 1
Nome: Ana
CPF: 111111111111
Idade: 25
ID: 2
Nome: Carlos
CPF: 222222222222
Idade: 52
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados.
ID: 3
Nome: XPTO Sales
CNPJ: 333333333333
ID: 4
Nome: XPTO Solutions
CNPJ: 44444444444444
BUILD SUCCESSFUL (total time: 0 seconds)

```

Figura 1: resultado da execução do 1º procedimento.

5. Análise e Conclusão

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

A principal vantagem da herança é permitir que uma classe pai compartilhe seus atributos e métodos com outras classes filhas; desta forma, naturalmente gera reaproveitamento de código e, portanto, economia de tempo e esforço de desenvolvimento.

Uma possível desvantagem é que, especificamente em Java, não existe o mecanismo de herança múltipla [1], na qual uma classe herda de 2 ou mais classes diferentes. Outra é o forte acoplamento causado entre a classe pai e a classe filha, na qual a classe filha depende diretamente da implementação da classe pai, a ponto de comprometer a facilidade de manutenção entre classes.

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

A implementação da interface `Serializable` por uma classe, em Java, é necessária para que seus objetos possam ser convertidos em uma sequência de bytes, e enviados através de algum canal digital (*stream*); o que inclui arquivos binários ou transferência via rede de dados, por exemplo. Ao chegar no destino, essa sequência de bytes deve ser convertida de volta ao seu formato original apropriado de objeto, para que possa ser devidamente utilizado.

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

O paradigma funcional é utilizado pela API `stream` no Java [2], através da possibilidade de transformar operações de iterações externas (como laços de repetição `for`, `while`, `do-while`) em iterações internas (*filter*, *map*, *reduce*, *sorted*, entre outras). Essas iterações internas abstraem o processo de repetição, ao permitir que a API tome controle e permita a *loops* mais otimizados. Por exemplo, na classe `PessoaFisicaRepo.java`, há uma implementação desse paradigma, com uso da função lambda (*arrow function*), através de uma sequência de chamadas em *pipe*:

```
public PessoaFisica obter(int id) {  
    return pessoasFisicas.stream()  
        .filter(pf -> pf.getId() == id)  
        .findFirst()  
        .orElse(null);  
}
```

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

Serialização de fluxo, ou em inglês, *stream serialization*; na qual classes nativas da API `java.io.*`, tais como `FileInputStream`, `ObjectInputStream`, entre outras, onde seus objetos são passados como argumento um dentro da instância do outro, em sequência, para inserir dados da aplicação nesses objetos através de métodos de leitura (*read*) e escrita (*write*). Isto ocorre, por exemplo, na implementação do método `recuperar()`

```

public void recuperar(String filename) {
    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
        pessoasFisicas = (ArrayList<PessoaFisica>) ois.readObject();
    }
    catch (ClassNotFoundException e) {
        LOGGER.log(Level.WARNING, e.toString(), e);
    }
    catch (IOException e) {
        LOGGER.log(Level.SEVERE, e.toString(), e);
    }
}
}

```

Obs: neste método, foi utilizada a sintaxe de “try-with-resources” [3], na qual o objeto stream é instanciado dentro da chamada do try(), entre parênteses. Novidade introduzida no Java 8.

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/cleytonmuto/CadastroPOO>

4. Resultados da execução dos códigos

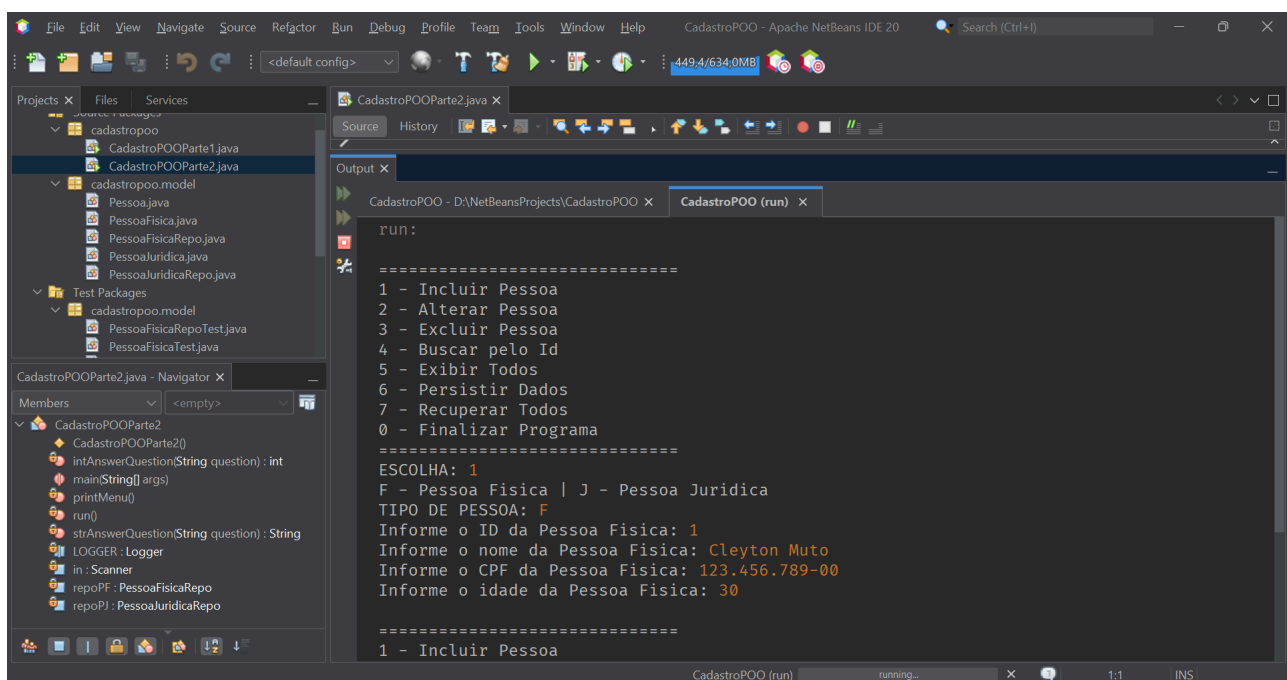


Figura 2: resultado da execução do 2º procedimento - opção incluir pessoa.

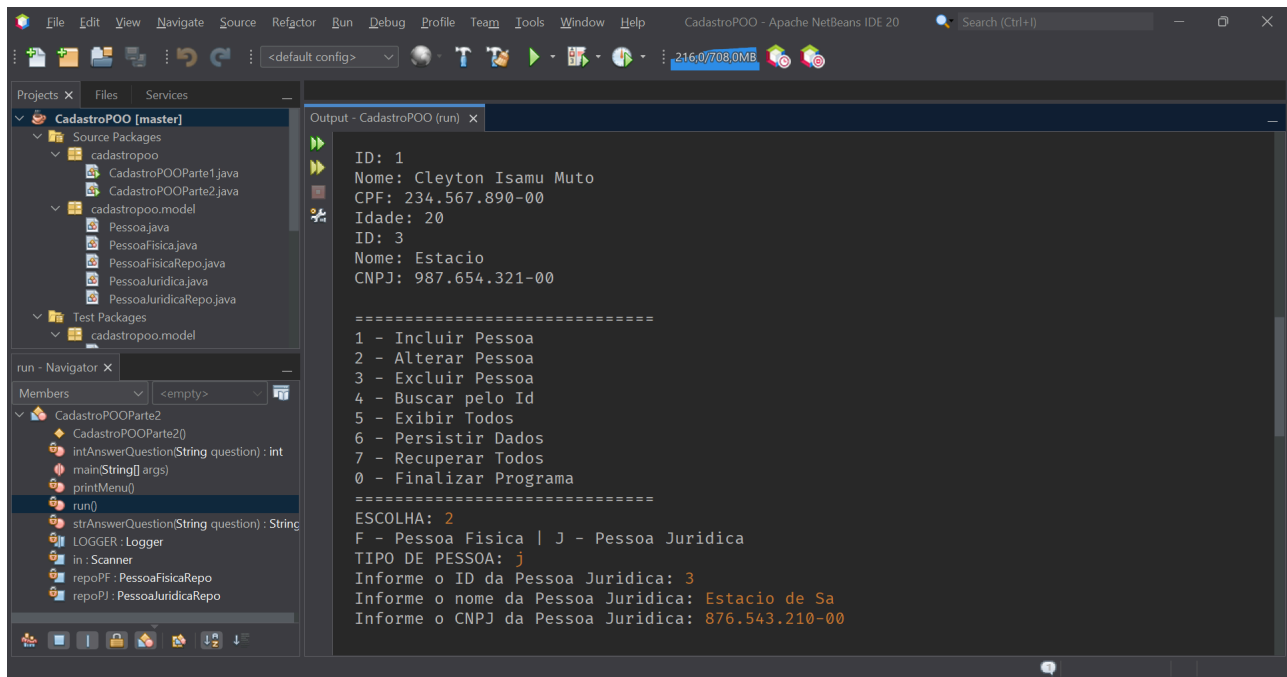


Figura 3: resultado da execução do 2º procedimento - alterar pessoa.

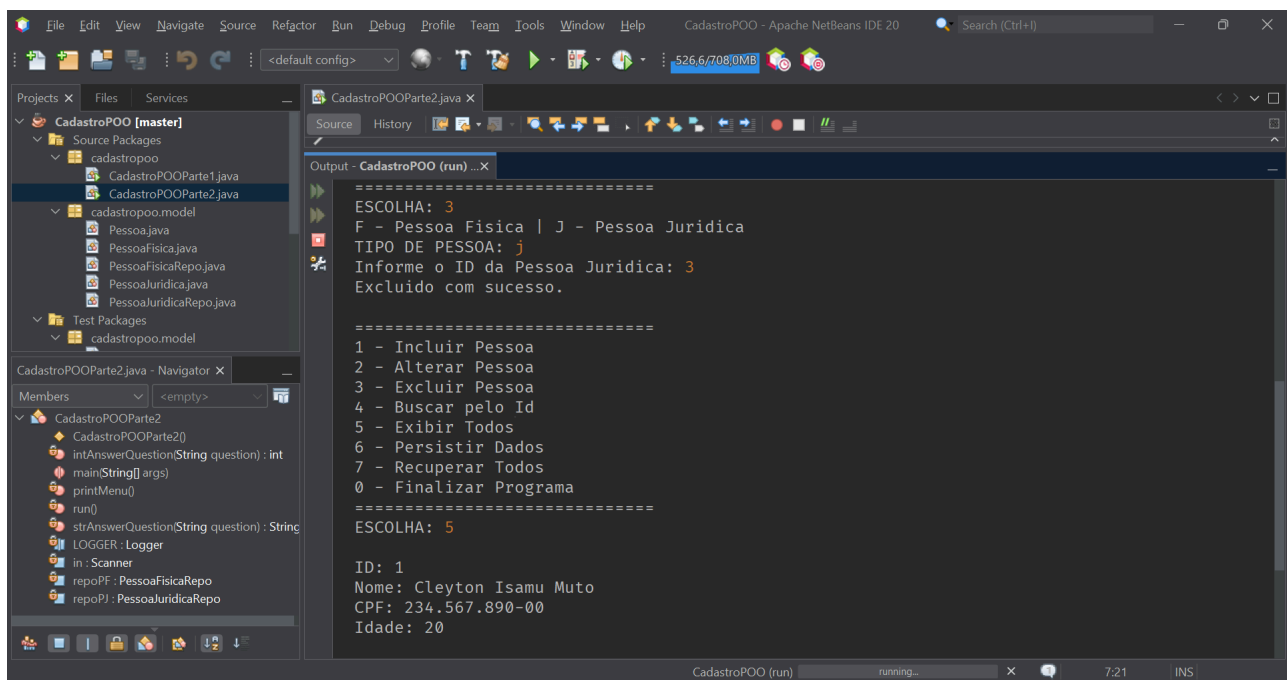


Figura 4: resultado da execução do 2º procedimento - excluir pessoa.

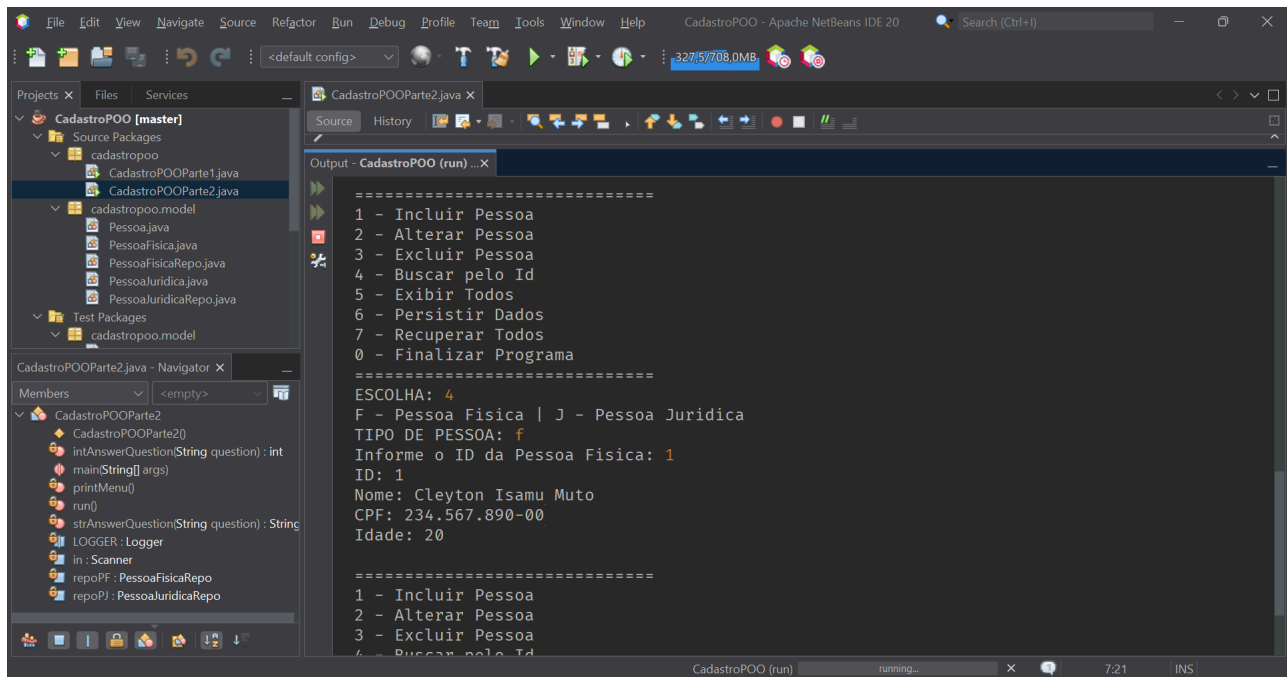


Figura 5: resultado da execução do 2º procedimento - buscar pelo ID.

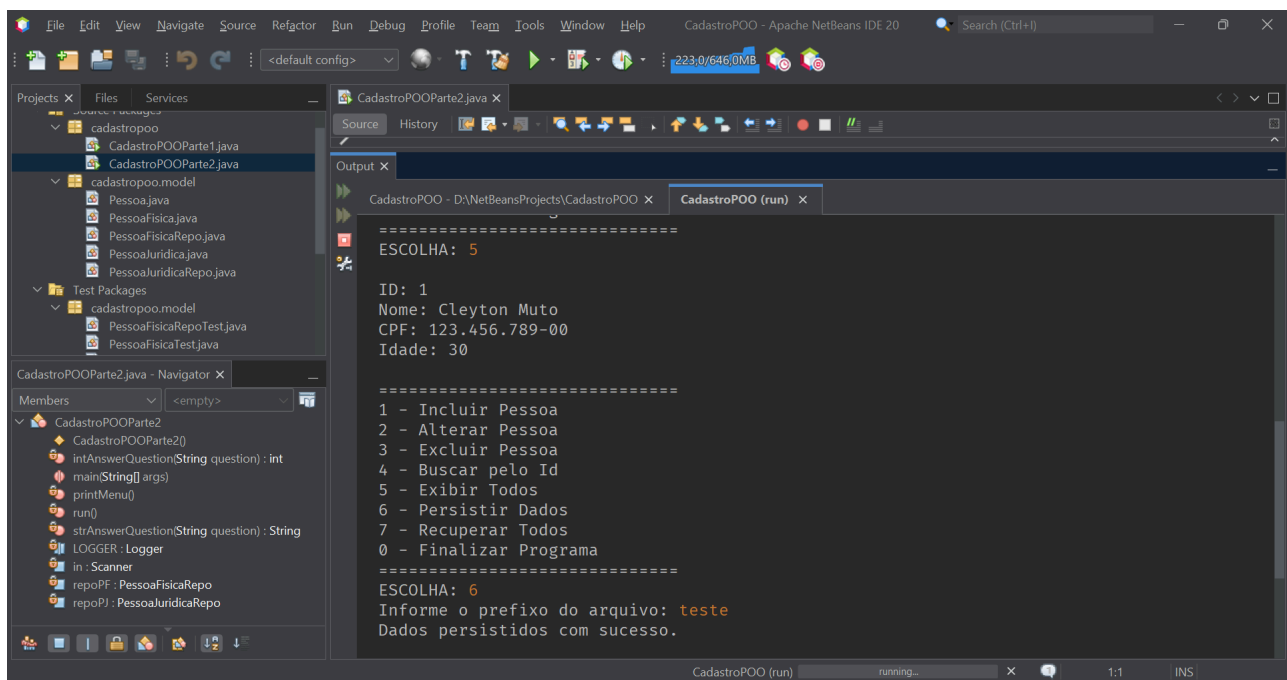


Figura 6: resultado da execução do 2º procedimento - exibir e persistir.

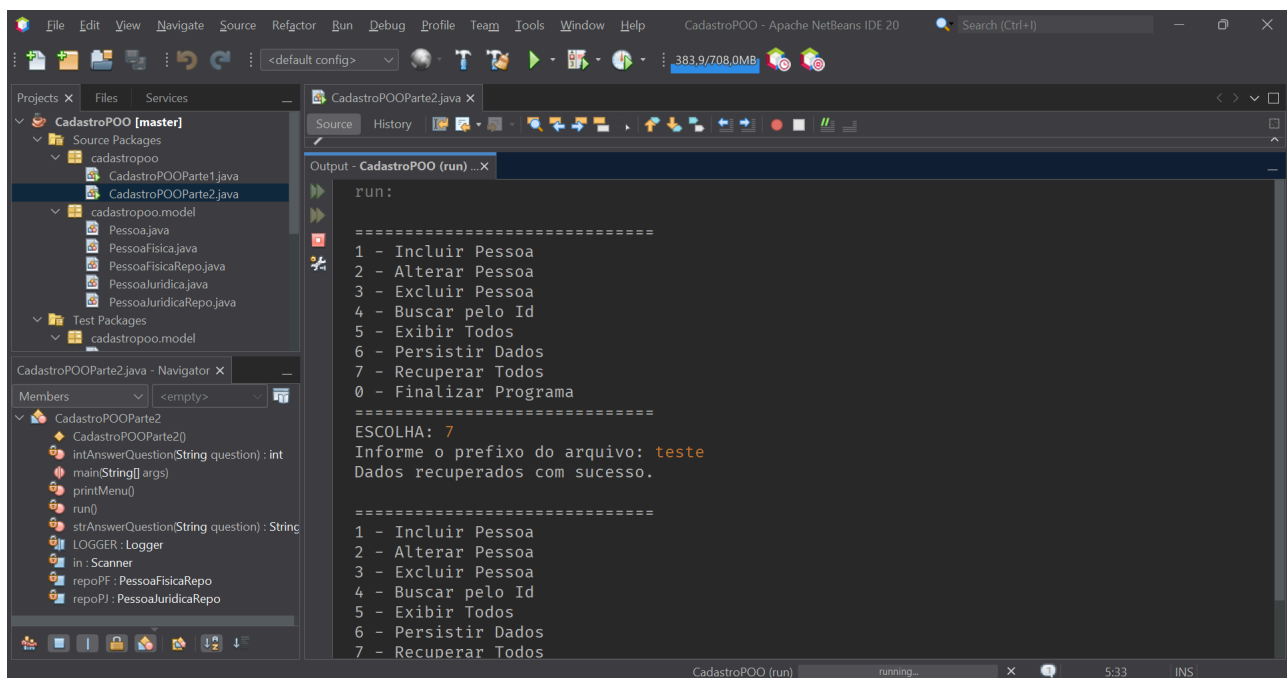


Figura 7: resultado da execução do 2º procedimento - recuperar.

5. Análise e Conclusão:

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

Elementos estáticos são aqueles pertencentes à classe, não necessariamente ao objeto instanciado. Assim, podem ser invocados e utilizados sem a obrigação do uso de um objeto de referência, mas também podem ser usados por objetos.

O método principal “main” adota esse modificador, para que possa ser invocado diretamente pela JVM (*Java Virtual Machine*) sem necessidade de instanciar um objeto do tipo da classe.

(b) Para que serve a classe Scanner?

A classe Scanner serve para efetuar a leitura de dados a partir de dispositivos de entrada como, por exemplo, o teclado.

(c) Como o uso de classes de repositório impactou na organização do código?

Impactou positivamente, por permitir a separação entre as funcionalidades de acesso aos dados (inserir, alterar, excluir, exibir), das classes de definição das entidades (Pessoa, PessoaFisica, PessoaJuridica), e da classe principal CadastroPOO, responsável pela interface textual (menu) com o usuário.

Referências

[1] ***“Multiple inheritance (C++ only)”***

Disponível em <https://www.ibm.com/docs/en/i/7.2?topic=only-multiple-inheritance-c>

Acesso em 11 de fevereiro de 2024.

[2] ***“The Power of Java Stream API”***

Disponível em

<https://medium.com/@AlexanderObregon/the-power-of-java-stream-api-d7c0ab7e4c5a>

Acesso em 11 de fevereiro de 2024.

[3] ***“The try-with-resources statement”***

Disponível em <https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html>

Acesso em 11 de fevereiro de 2024.

Anexo I: códigos do projeto

```
package cadastrpoo;

import cadastrpoo.model.*;

/**
 *
 * @author Cleyton
 */
public class CadastroP00Parte1 {

    private final String FILE1;
    private final String FILE2;

    public CadastroP00Parte1() {
        FILE1 = "resources/pf.dat";
        FILE2 = "resources/pj.dat";
    }

    private void run() {
        pfTest();
        pjTest();
    }

    private void pfTest() {
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        PessoaFisica pf1 = new PessoaFisica(1, "Ana", "1111111111", 25);
        PessoaFisica pf2 = new PessoaFisica(2, "Carlos", "2222222222", 52);
        repo1.inserir(pf1);
        repo1.inserir(pf2);
        repo1.persistir(FILE1);
        System.out.println("Dados de Pessoa Fisica Armazenados.");
        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        repo2.recuperar(FILE1);
        System.out.println("Dados de Pessoa Fisica Recuperados.");
        for (PessoaFisica pf : repo2.obterTodos()) {
            pf.exibir();
        }
    }

    private void pjTest() {
        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        PessoaJuridica pj1 = new PessoaJuridica(3, "XPTO Sales", "333333333333");
        PessoaJuridica pj2 = new PessoaJuridica(4, "XPTO Solutions", "444444444444");
        repo3.inserir(pj1);
        repo3.inserir(pj2);
        repo3.persistir(FILE2);
        System.out.println("Dados de Pessoa Juridica Armazenados.");
        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        repo4.recuperar(FILE2);
        System.out.println("Dados de Pessoa Juridica Recuperados.");
        for (PessoaJuridica pj : repo4.obterTodos()) {
            pj.exibir();
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        new CadastroP00Parte1().run();
    }
}
```



```

package cadastrpoo;

import cadastrpoo.model.*;
import java.util.Scanner;
import java.util.logging.Logger;
import java.util.logging.Level;
import java.io.File;

/**
 *
 * @author Cleyton
 */
public class CadastroP00Parte2 {

    private static final Logger LOGGER =
Logger.getLogger(CadastroP00Parte2.class.getName());
    private Scanner in;
    private PessoaFisicaRepo repoPF;
    private PessoaJuridicaRepo repoPJ;

    public CadastroP00Parte2() {
        in = new Scanner(System.in);
        repoPF = new PessoaFisicaRepo();
        repoPJ = new PessoaJuridicaRepo();
    }

    private String strAnswerQuestion(String question) {
        System.out.print(question);
        return in.nextLine();
    }

    private int intAnswerQuestion(String question) {
        System.out.print(question);
        String strValue = in.nextLine();
        int intValue = 0;
        try {
            intValue = Integer.valueOf(strValue);
        }
        catch (NumberFormatException e) {
            LOGGER.log(Level.SEVERE, e.toString(), e);
        }
        return intValue;
    }

    private void run() {
        String opcao = "";
        while (!opcao.equals("0")) {
            printMenu();
            opcao = strAnswerQuestion("ESCOLHA: ");
            switch (opcao) {
                case "1": {
                    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
                    String escolhaIncluir = strAnswerQuestion("TIPO DE PESSOA:
").toUpperCase();
                    if (escolhaIncluir.equals("F")) {
                        int id = intAnswerQuestion("Informe o ID da Pessoa Fisica: ");
                        String nome = strAnswerQuestion("Informe o nome da Pessoa
Fisica: ");
                        String cpf = strAnswerQuestion("Informe o CPF da Pessoa
Fisica: ");
                        int idade = intAnswerQuestion("Informe o idade da Pessoa
Fisica: ");
                        repoPF.inserir(new PessoaFisica(id, nome, cpf, idade));
                    }
                }
            }
        }
    }
}

```

```

        else if (escolhaIncluir.equals("J")) {
            int id = intAnswerQuestion("Informe o ID da Pessoa Juridica:
");
            String nome = strAnswerQuestion("Informe o nome da Pessoa
Juridica: ");
            String cnpj = strAnswerQuestion("Informe o CNPJ da Pessoa
Juridica: ");
            repoPJ.inserir(new PessoaJuridica(id, nome, cnpj));
        }
        else {
            System.out.println("Erro: Escolha Invalida!");
        }
    }; break;
case "2": {
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    String escolhaAlterar = strAnswerQuestion("TIPO DE PESSOA:
").toUpperCase();
    if (escolhaAlterar.equals("F")) {
        try {
            int id = intAnswerQuestion("Informe o ID da Pessoa Fisica:
");
            PessoaFisica pf = repoPF.obter(id);
            if (pf != null) {
                pf.setNome(strAnswerQuestion("Informe o nome da Pessoa
Fisica: "));
                pf.setCpf(strAnswerQuestion("Informe o CPF da Pessoa
Fisica: "));
                pf.setIdade(intAnswerQuestion("Informe o idade da
Pessoa Fisica: "));
                repoPF.alterar(pf);
            }
            else {
                System.out.println("ID nao encontrado!");
            }
        }
        catch (NullPointerException e){
            System.err.println("ID nao encontrado!");
            LOGGER.log(Level.SEVERE, e.toString(), e);
        }
    }
    else if (escolhaAlterar.equals("J")) {
        try {
            int id = intAnswerQuestion("Informe o ID da Pessoa
Juridica: ");
            PessoaJuridica pj = repoPJ.obter(id);
            if (pj != null) {
                pj.setNome(strAnswerQuestion("Informe o nome da Pessoa
Juridica: "));
                pj.setCnpj(strAnswerQuestion("Informe o CNPJ da Pessoa
Juridica: "));
                repoPJ.alterar(pj);
            }
            else {
                System.out.println("ID nao encontrado!");
            }
        }
        catch (NullPointerException e){
            System.err.println("ID nao encontrado!");
            LOGGER.log(Level.SEVERE, e.toString(), e);
        }
    }
    else {
        System.out.println("Erro: Escolha Invalida!");
    }
}; break;
case "3": {

```

```

        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String escolhaExcluir = strAnswerQuestion("TIPO DE PESSOA:
").toUpperCase();
        if (escolhaExcluir.equals("F")) {
            System.out.println(repoPF.excluir(intAnswerQuestion("Informe o
ID da Pessoa Fisica: ")) ?
                "Excluido com sucesso." : "ID nao encontrado." );
        }
        else if (escolhaExcluir.equals("J")) {
            System.out.println(repoPJ.excluir(intAnswerQuestion("Informe o
ID da Pessoa Juridica: ")) ?
                "Excluido com sucesso." : "ID nao encontrado." );
        }
        else {
            System.out.println("Erro: Escolha Invalida!");
        }
    }; break;
    case "4": {
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String escolhaExibir = strAnswerQuestion("TIPO DE PESSOA:
").toUpperCase();
        if (escolhaExibir.equals("F")) {
            try {
                repoPF.obter(intAnswerQuestion("Informe o ID da Pessoa
Fisica: ")).exibir();
            }
            catch (NullPointerException e){
                System.err.println("Pessoa nao encontrada!");
                LOGGER.log(Level.SEVERE, e.toString(), e);
            }
        }
        else if (escolhaExibir.equals("J")) {
            try {
                repoPJ.obter(intAnswerQuestion("Informe o ID da Pessoa
Juridica: ")).exibir();
            }
            catch (NullPointerException e){
                System.err.println("Pessoa nao encontrada!");
                LOGGER.log(Level.SEVERE, e.toString(), e);
            }
        }
        else {
            System.out.println("Erro: Escolha Invalida!");
        }
    }; break;
    case "5": {
        System.out.println();
        for (PessoaFisica pf : repoPF.obterTodos()) {
            pf.exibir();
        }
        for (PessoaJuridica pj : repoPJ.obterTodos()) {
            pj.exibir();
        }
    }; break;
    case "6": {
        String prefixo = strAnswerQuestion("Informe o prefixo do arquivo:
");
        String pfFilename =
"resources/".concat(prefixo.concat(".fisica.bin"));
        String pjFilename =
"resources/".concat(prefixo.concat(".juridica.bin"));
        repoPF.persistir(pfFilename);
        repoPJ.persistir(pjFilename);
        System.out.println("Dados persistidos com sucesso.");
    }; break;
    case "7": {

```

```

        String prefixo = strAnswerQuestion("Informe o prefixo do arquivo:
");
        String pfFilename =
"resources/".concat(prefixo.concat(".fisica.bin"));
        String pjFilename =
"resources/".concat(prefixo.concat(".juridica.bin"));
        File pfFile = new File(pfFilename);
        File pjFile = new File(pjFilename);
        if (pfFile.exists() && pjFile.exists()) {
            repoPF.recuperar(pfFilename);
            repoPJ.recuperar(pjFilename);
            System.out.println("Dados recuperados com sucesso.");
        }
        else {
            System.out.println("Prefixo invalido. Arquivo nao
encontrado.");
        }
    }; break;
    default: {
        System.out.println("Escolha invalida!");
    };
}
}

private void printMenu() {
    System.out.println("\n=====");
    System.out.println("1 - Incluir Pessoa");
    System.out.println("2 - Alterar Pessoa");
    System.out.println("3 - Excluir Pessoa");
    System.out.println("4 - Buscar pelo Id");
    System.out.println("5 - Exibir Todos");
    System.out.println("6 - Persistir Dados");
    System.out.println("7 - Recuperar Todos");
    System.out.println("0 - Finalizar Programa");
    System.out.println("=====");
}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    new CadastroPOOParte2().run();
}
}

```

```

package cadastrpoo.model;

import java.io.Serializable;

/**
 *
 * @author Cleyton
 */
public class Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;

    private int id;
    private String nome;

```

```

public Pessoa() {
    id = 0;
    nome = "";
}

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: " + getId());
    System.out.println("Nome: " + getNome());
}

public long getSerialVersionUID() {
    return serialVersionUID;
}
}

```

```

package cadastrpoo.model;

import java.io.Serializable;

/**
 *
 * @author Cleyton
 */
public class PessoaFisica extends Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;

    private String cpf;
    private int idade;

    public PessoaFisica() {
        cpf = "";
        idade = 0;
    }

    public PessoaFisica(String cpf, int idade) {
        this.cpf = cpf;
        this.idade = idade;
    }

    public PessoaFisica(int id, String nome, String cpf, int idade) {
        setId(id);
        setNome(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
    /**
     * overridden method exibir()
     */
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + getCpf());
        System.out.println("Idade: " + getIdade());
    }

    @Override
    /**
     * overridden method getSerialVersionUID()
     */
    public long getSerialVersionUID() {
        return serialVersionUID;
    }
}

```

```

package cadastrpoo.model;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.IOException;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.logging.Logger;
import java.util.logging.Level;

/**
 *
 * @author Cleyton
 */
public class PessoaFisicaRepo implements Serializable {

    private static final long serialVersionUID = 1L;
    private static final Logger LOGGER = Logger.getLogger(PessoaFisicaRepo.class.getName());

    private ArrayList<PessoaFisica> pessoasFisicas;

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

    public boolean inserir(PessoaFisica pf) {
        return pessoasFisicas.contains(pf) ? false : pessoasFisicas.add(pf);
    }

    public boolean alterar(PessoaFisica pf) {
        int position = pessoasFisicas.indexOf(pf);
    }
}

```

```

        if (position != -1) {
            pessoasFisicas.set(position, pf);
            return true;
        }
        return false;
    }

    public boolean excluir(int id) {
        return pessoasFisicas.remove(obter(id));
    }

    public PessoaFisica obter(int id) {
        return pessoasFisicas.stream()
            .filter(pf -> pf.getId() == id)
            .findFirst()
            .orElse(null);
    }

    public ArrayList<PessoaFisica> obterTodos() {
        return pessoasFisicas;
    }

    public void persistir(String filename) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filename))) {
            oos.writeObject(pessoasFisicas);
        }
        catch (IOException e) {
            LOGGER.log(Level.SEVERE, e.toString(), e);
        }
    }

    public void recuperar(String filename) {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
            pessoasFisicas = (ArrayList<PessoaFisica>) ois.readObject();
        }
        catch (ClassNotFoundException e) {
            LOGGER.log(Level.WARNING, e.toString(), e);
        }
        catch (IOException e) {
            LOGGER.log(Level.SEVERE, e.toString(), e);
        }
    }

    public long getSerialVersionUID() {
        return serialVersionUID;
    }
}

```

```

package cadastrapoo.model;

import java.io.Serializable;

/**
 * @author Cleyton
 */
public class PessoaJuridica extends Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;

    private String cnpj;

    public PessoaJuridica() {
        cnpj = "";
    }

    public PessoaJuridica(String cnpj) {
        this.cnpj = cnpj;
    }
}

```

```

    }

    public PessoaJuridica(int id, String nome, String cnpj) {
        setId(id);
        setNome(nome);
        this.cnpj = cnpj;
    }

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    @Override
    /**
     * overridden method exibir()
     */
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + getCnpj());
    }

    @Override
    /**
     * overridden method getSerialVersionUID()
     */
    public long getSerialVersionUID() {
        return serialVersionUID;
    }
}

```

```

package cadastrpoo.model;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.IOException;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.logging.Logger;
import java.util.logging.Level;

/**
 *
 * @author Cleyton
 */
public class PessoaJuridicaRepo implements Serializable {

    private static final long serialVersionUID = 1L;
    private static final Logger LOGGER = Logger.getLogger(PessoaJuridicaRepo.class.getName());

    private ArrayList<PessoaJuridica> pessoasJuridicas;

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

    public boolean inserir(PessoaJuridica pj) {
        return pessoasJuridicas.contains(pj) ? false : pessoasJuridicas.add(pj);
    }

    public boolean alterar(PessoaJuridica pj) {
        int position = pessoasJuridicas.indexOf(pj);
        if (position != -1) {

```



```

        pessoasJuridicas.set(position, pj);
        return true;
    }
    return false;
}

public boolean excluir(int id) {
    return pessoasJuridicas.remove(obter(id));
}

public PessoaJuridica obter(int id) {
    return pessoasJuridicas.stream()
        .filter(pj -> pj.getId() == id)
        .findFirst()
        .orElse(null);
}

public ArrayList<PessoaJuridica> obterTodos() {
    return pessoasJuridicas;
}

public void persistir(String filename) {
    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filename))) {
        oos.writeObject(pessoasJuridicas);
    }
    catch (IOException e) {
        LOGGER.log(Level.SEVERE, e.toString(), e);
    }
}

public void recuperar(String filename) {
    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
        pessoasJuridicas = (ArrayList<PessoaJuridica>) ois.readObject();
    }
    catch (ClassNotFoundException e) {
        LOGGER.log(Level.WARNING, e.toString(), e);
    }
    catch (IOException e) {
        LOGGER.log(Level.SEVERE, e.toString(), e);
    }
}

public long getSerialVersionUID() {
    return serialVersionUID;
}
}

```