 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.

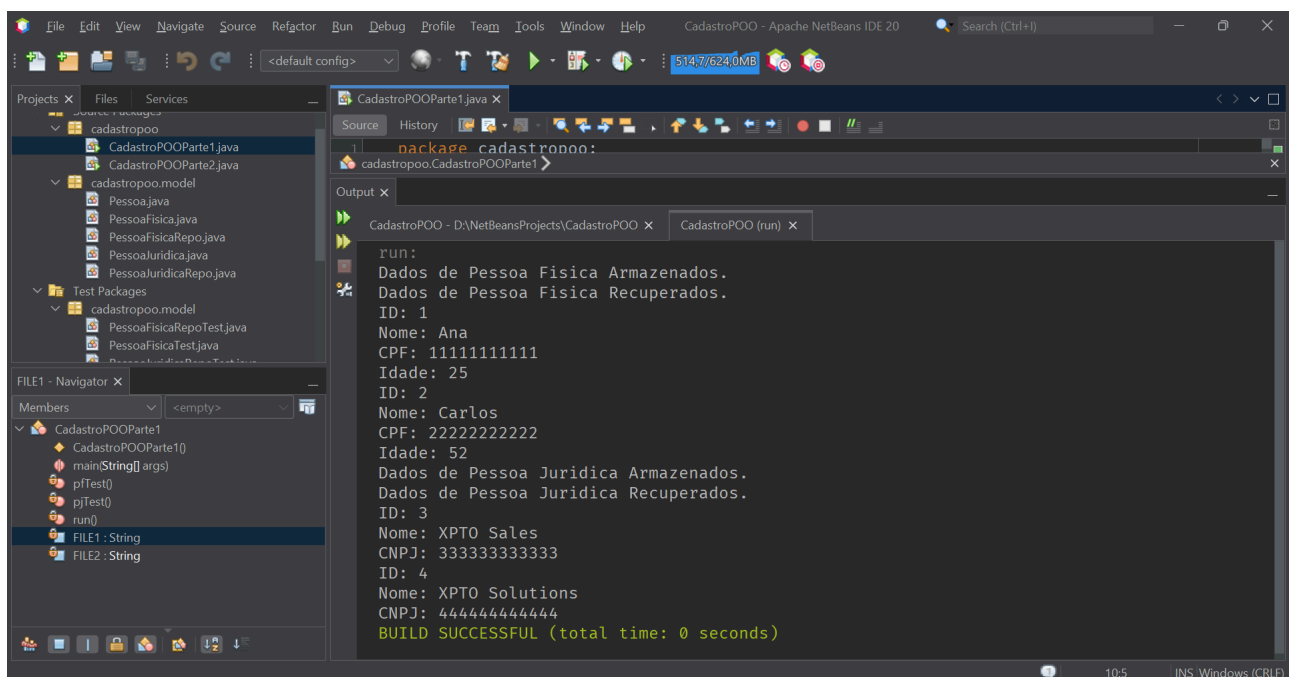
1. Título da Prática: “1º Procedimento | Criação das Entidades e Sistema de Persistência”

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: <https://github.com/cleytonmuto/CadastroPOO>

4. Resultados da execução dos códigos



```

package cadastrapoo;

castradopoo.CadastroPOOParte1

Output
CadastrarPOO - D:\NetBeansProjects\CadastroPOO \ CadastroPOO (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

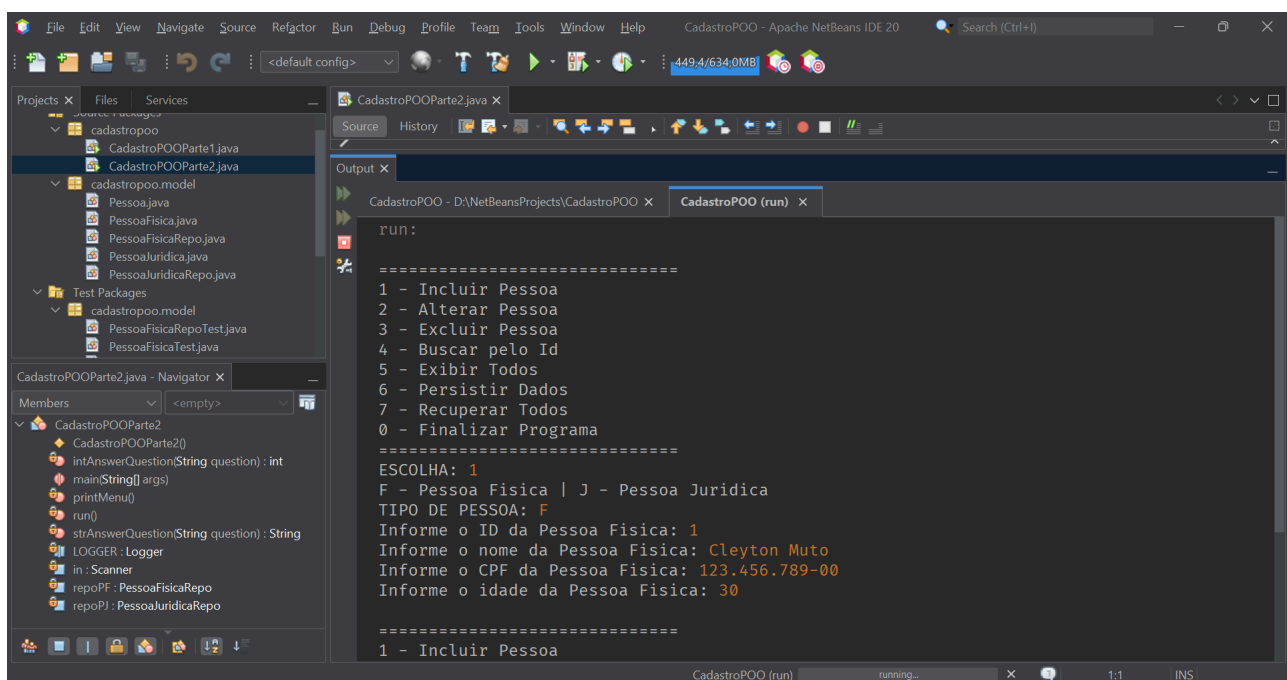


Figura 2: resultado da execução do 2º procedimento - opção incluir pessoa.

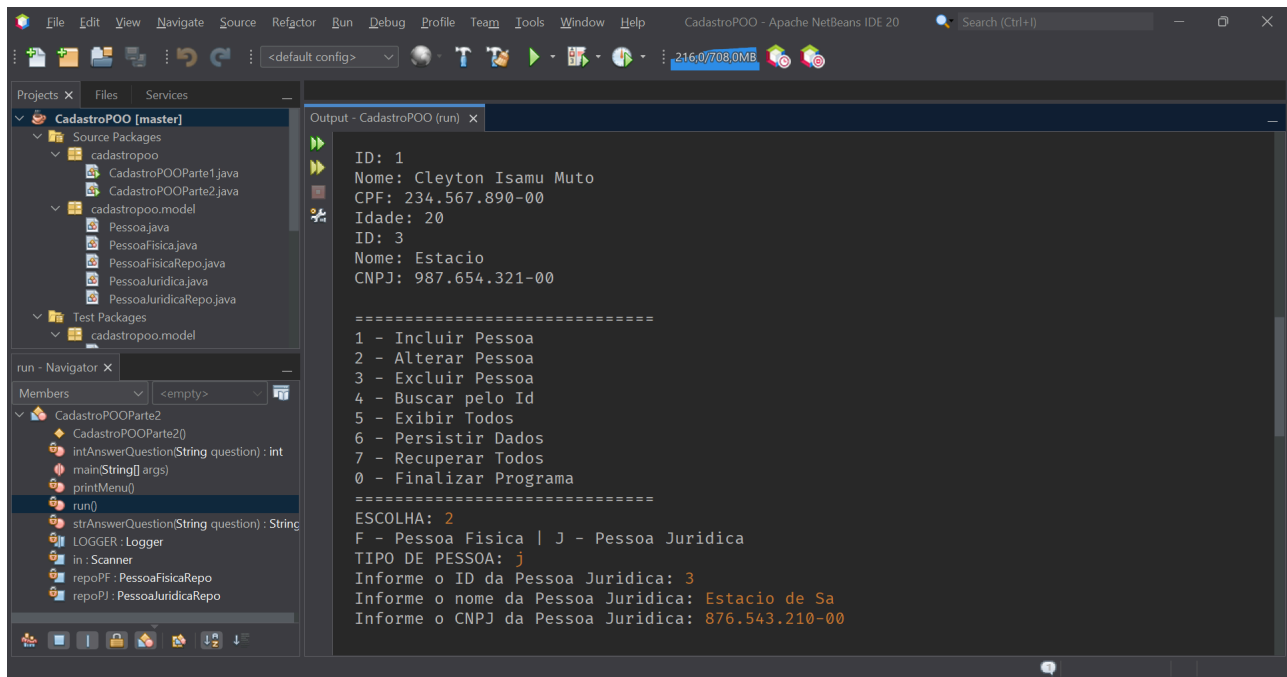


Figura 3: resultado da execução do 2º procedimento - alterar pessoa.

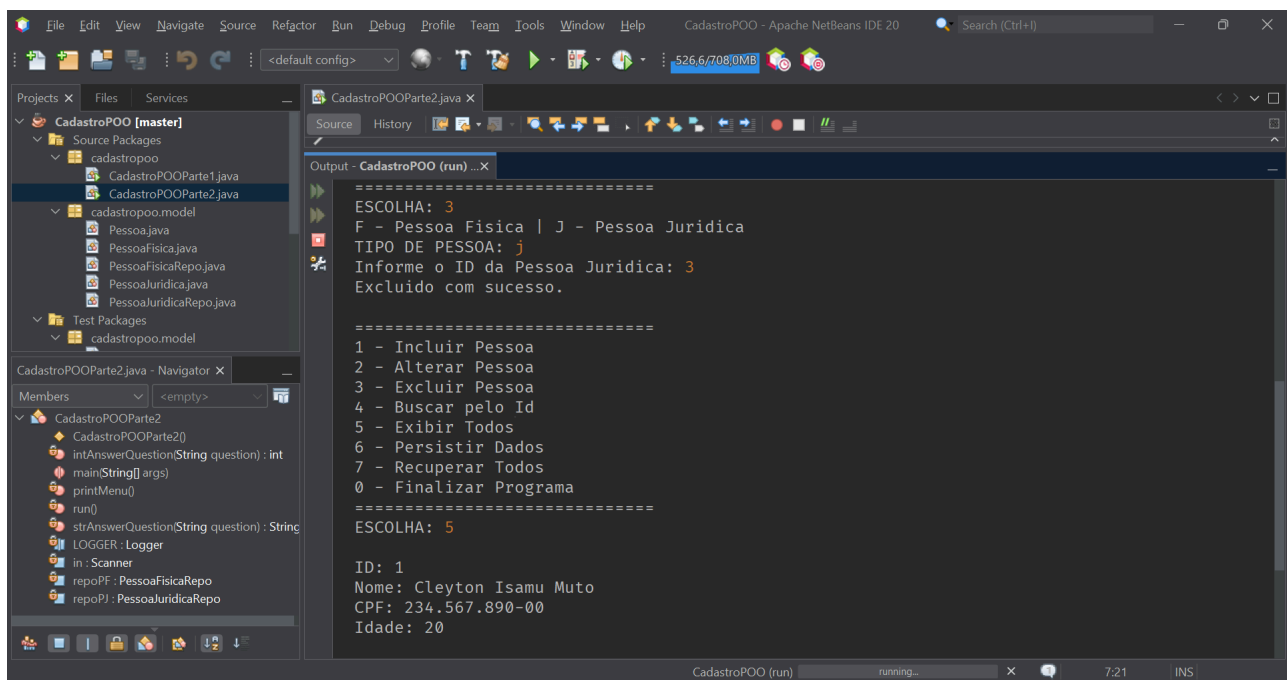


Figura 4: resultado da execução do 2º procedimento - excluir pessoa.

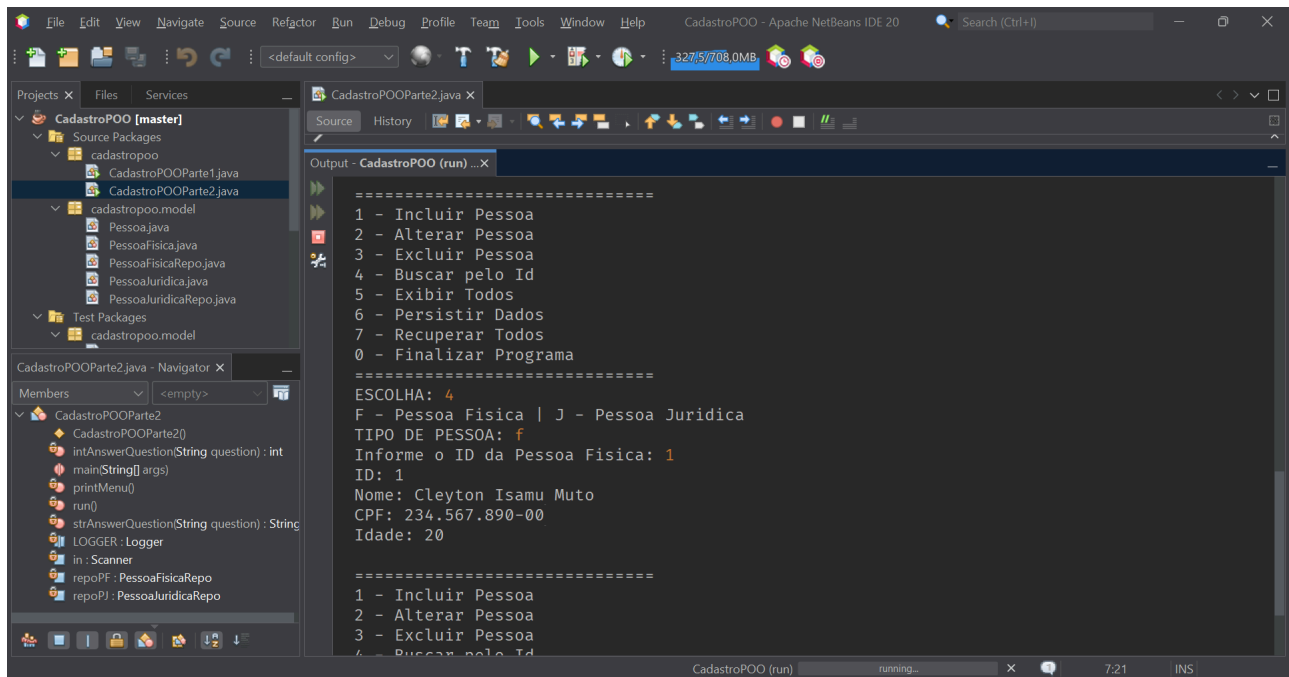


Figura 5: resultado da execução do 2º procedimento - buscar pelo ID.

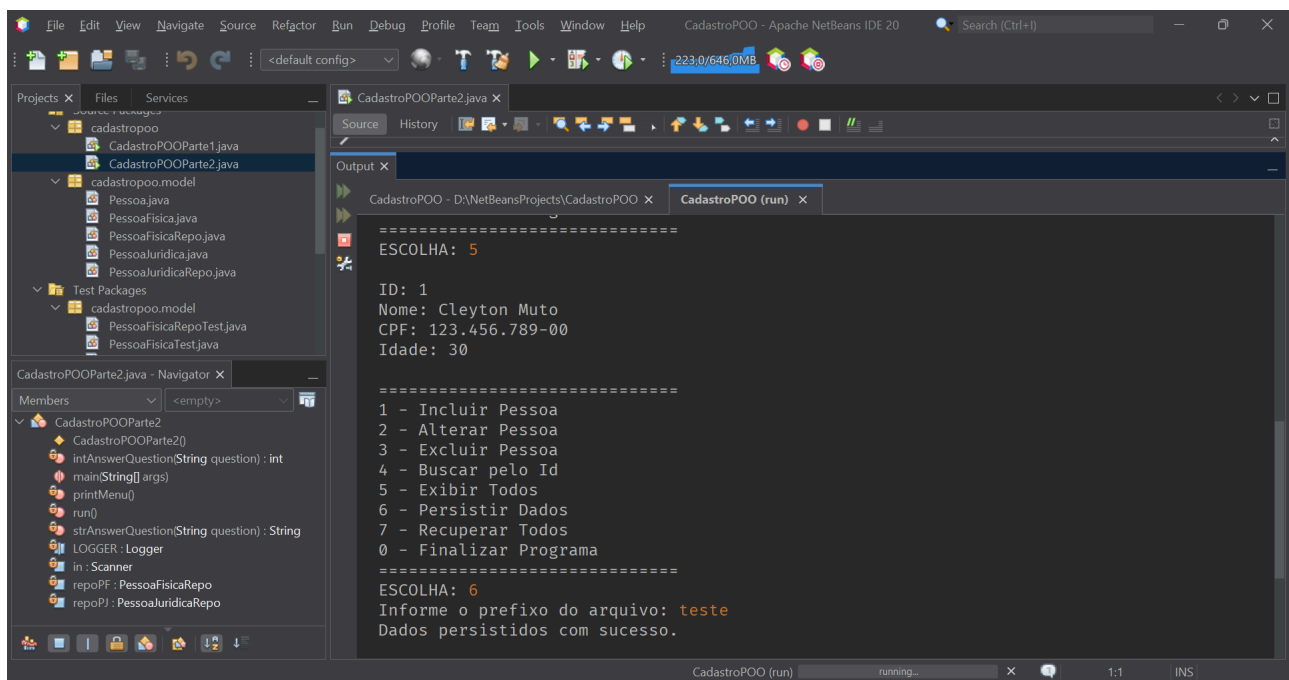


Figura 6: resultado da execução do 2º procedimento - exibir e persistir.

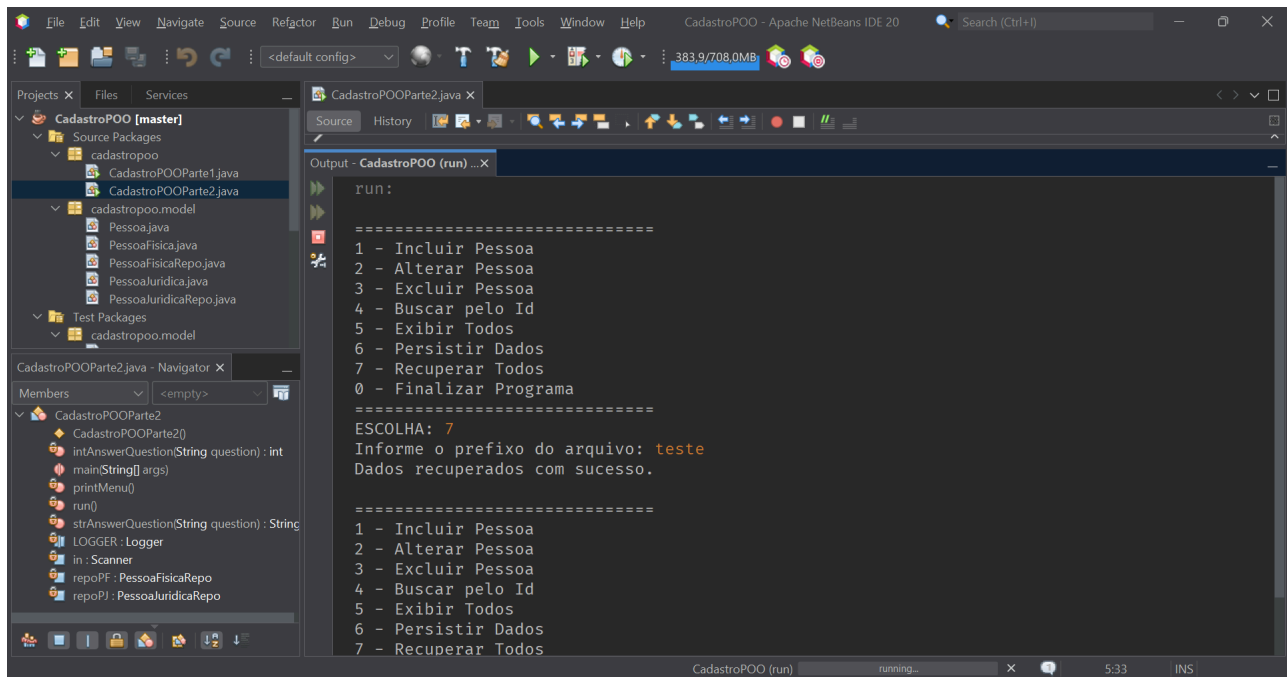


Figura 7: resultado da execução do 2º procedimento - recuperar.

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.