



UNIVERSIDADE ESTÁCIO DE SÁ POLO IBIRITÉ

CURSO: DESENVOLVIMENTO FULL STACK

DISCIPLINA: INICIANDO O CAMINHO PELO JAVA

EVERTON GOMES COSTA

TURMA: 22.3

3º SEMESTRE

IBIRITÉ-MG

2023

1º Procedimento | Criação das Entidades e Sistema de Persistência

Análise e Conclusão:

Vantagens e Desvantagens do Uso de Herança.

Vantagens:

Reutilização de código: A herança permite que você crie novas classes com base nas classes existentes, aproveitando a estrutura e comportamento das classes pai.

Hierarquia: Ajuda a criar uma hierarquia de classes que reflete as relações do mundo real.

Polimorfismo: permite que objetos de classes derivadas sejam tratados como objetos da classe base, o que facilita a criação de código flexível e genérico.

Desvantagens:

Acoplamento: Herança pode levar a um alto grau de adaptação entre as classes, tornando-as mais dependentes umas das outras.

Complexidade: Hierarquias de herança complexas podem ser difíceis de entender e manter.

Limitações de Múltipla Herança: algumas linguagens, incluindo Java, não suportam herança múltipla direta de classes. Isso pode levar a limitações na reutilização de código.

Interface Serializável para Persistência em Arquivos Binários:

A interface 'Serializable' é necessária ao prosseguimento em arquivos binários porque ela marca uma classe como serializável, permitindo que seus objetos sejam convertidos em uma sequência de bytes para armazenamento ou transmissão. Quando objetos são serializados em arquivos binários, a estrutura e os valores dos campos são armazenados de forma que podem ser recuperados posteriormente. Uma interface 'Serializable' garante que os

objetos possam ser corretamente serializados e desserializados, preservando sua integridade.

Paradigma Funcional na API Stream do Java:

A API Stream no Java utiliza o paradigma funcional para oferecer uma maneira mais concisa e eficiente de trabalhar com coleções de dados. Ela permite realizar operações de transformação e filtragem em coleções de forma mais declarativa, evitando loops explícitos. A API Stream utiliza funções lambda e expressões funcionais para permitir que você defina operações de maneira mais clara e legível.

Padrão de Desenvolvimento na Persistência de Dados em Arquivos em Java:

Um padrão comum de desenvolvimento na persistência de dados em arquivos em Java é usar a serialização para armazenar objetos em formato binário. A serialização é uma forma relativamente simples de salvar e carregar objetos diretamente em arquivos. No entanto, é importante observar que a serialização pode ter limitação em relação à evolução dos objetos ao longo do tempo, o que pode resultar em problemas de compatibilidade entre diferentes versões de classes.

Outras abordagens incluem a gravação e leitura manual dos dados usando fluxos de entrada/saída (por exemplo, `FileInputStream` e `FileOutputStream`) ou a utilização de bibliotecas externas, como o JSON ou XML, para armazenar os dados de forma mais flexível e legível.

CÓDIGO DO PRIMEIRO PROCEDIMENTO:

PESSOA.JAVA =

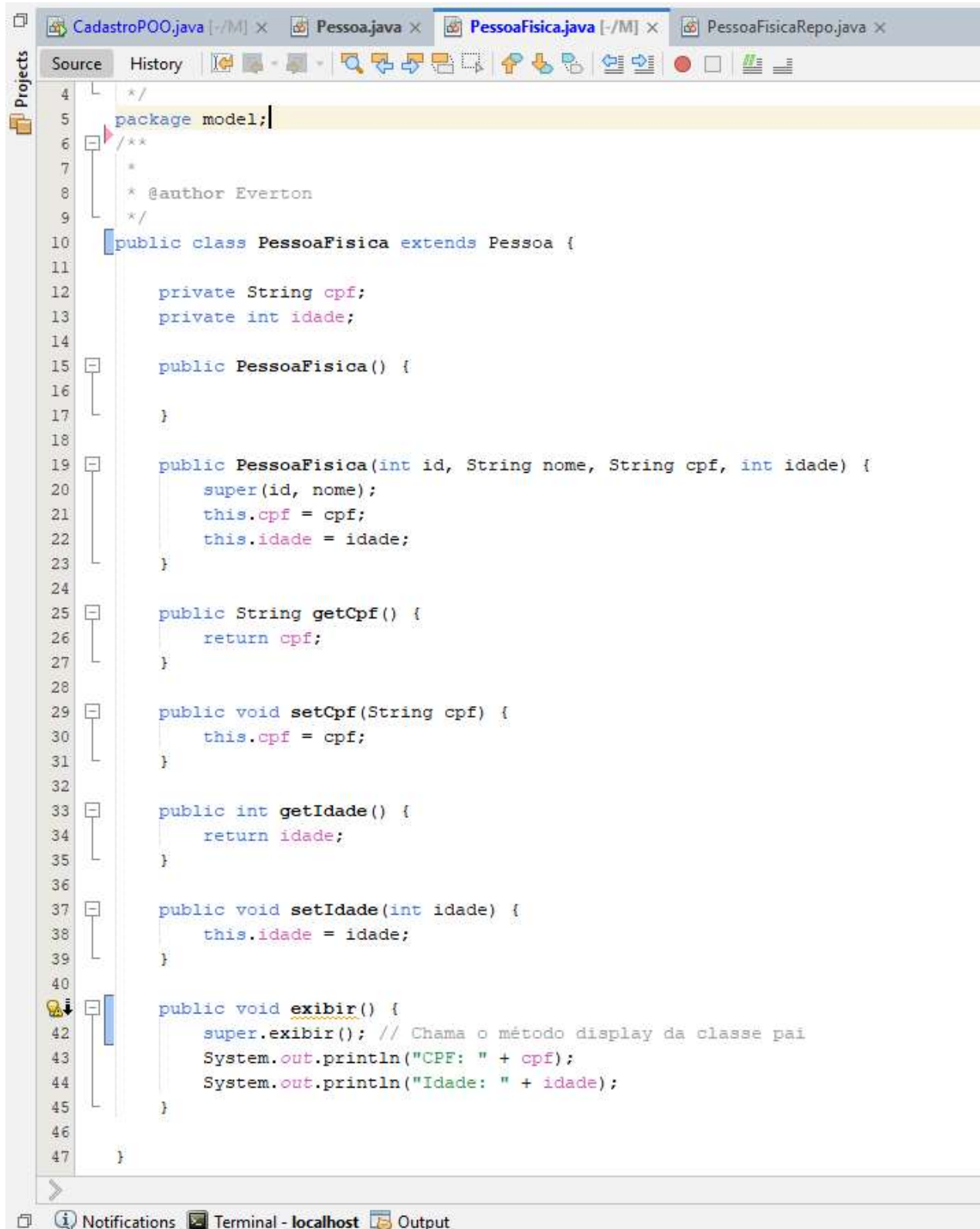


The screenshot shows an IDE with four tabs: CadastroPOO.java, Pessoa.java (active), PessoaFisica.java, and PessoaFisicaRepo.java. The active tab displays the following Java code:

```
4  */
5  package model;
6
7  import java.io.Serializable;
8
9  /**
10   *
11   * @author Everton
12   */
13  public class Pessoa implements Serializable {
14
15      private int id;
16      private String nome;
17
18      public Pessoa() {
19
20      }
21
22      public Pessoa(int id, String nome) {
23          this.id = id;
24          this.nome = nome;
25      }
26
27      public int getId() {
28          return id;
29      }
30
31      public void setId(int id) {
32          this.id = id;
33      }
34
35      public String getNome() {
36          return nome;
37      }
38
39      public void setNome(String nome) {
40          this.nome = nome;
41      }
42
43      public void exibir() {
44          System.out.println("ID: " + id);
45          System.out.println("Nome: " + nome);
46      }
47  }
```

The bottom of the IDE shows a breadcrumb trail 'model.Pessoa' and a status bar with 'Notifications', 'Terminal - localhost', and 'Output'.

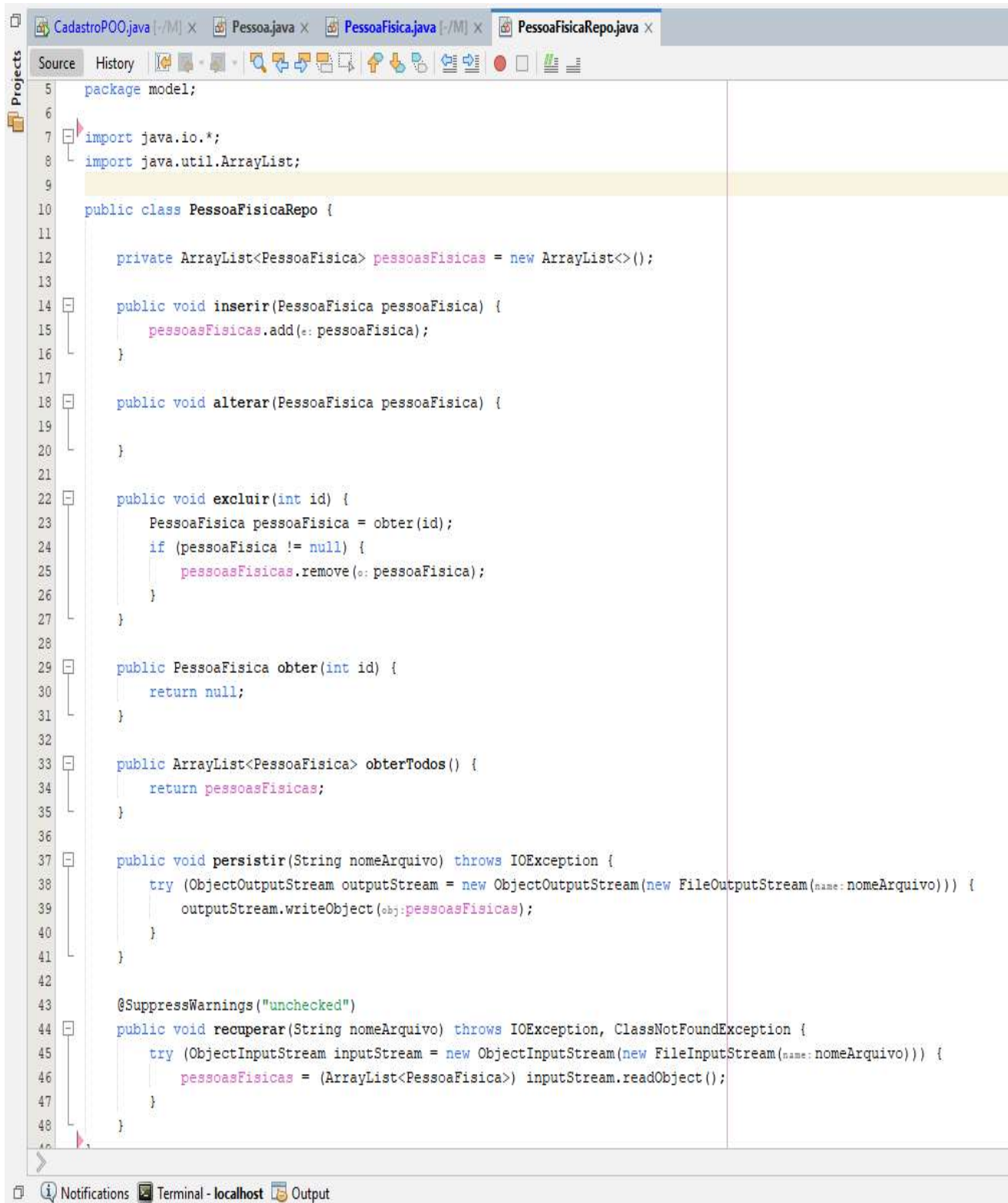
PESSOA FISICA.JAVA =



```
4  */
5  package model;
6  /**
7   *
8   * @author Everton
9   */
10 public class PessoaFisica extends Pessoa {
11
12     private String cpf;
13     private int idade;
14
15     public PessoaFisica() {
16
17     }
18
19     public PessoaFisica(int id, String nome, String cpf, int idade) {
20         super(id, nome);
21         this.cpf = cpf;
22         this.idade = idade;
23     }
24
25     public String getCpf() {
26         return cpf;
27     }
28
29     public void setCpf(String cpf) {
30         this.cpf = cpf;
31     }
32
33     public int getIdade() {
34         return idade;
35     }
36
37     public void setIdade(int idade) {
38         this.idade = idade;
39     }
40
41     public void exibir() {
42         super.exibir(); // Chama o método display da classe pai
43         System.out.println("CPF: " + cpf);
44         System.out.println("Idade: " + idade);
45     }
46
47 }
```

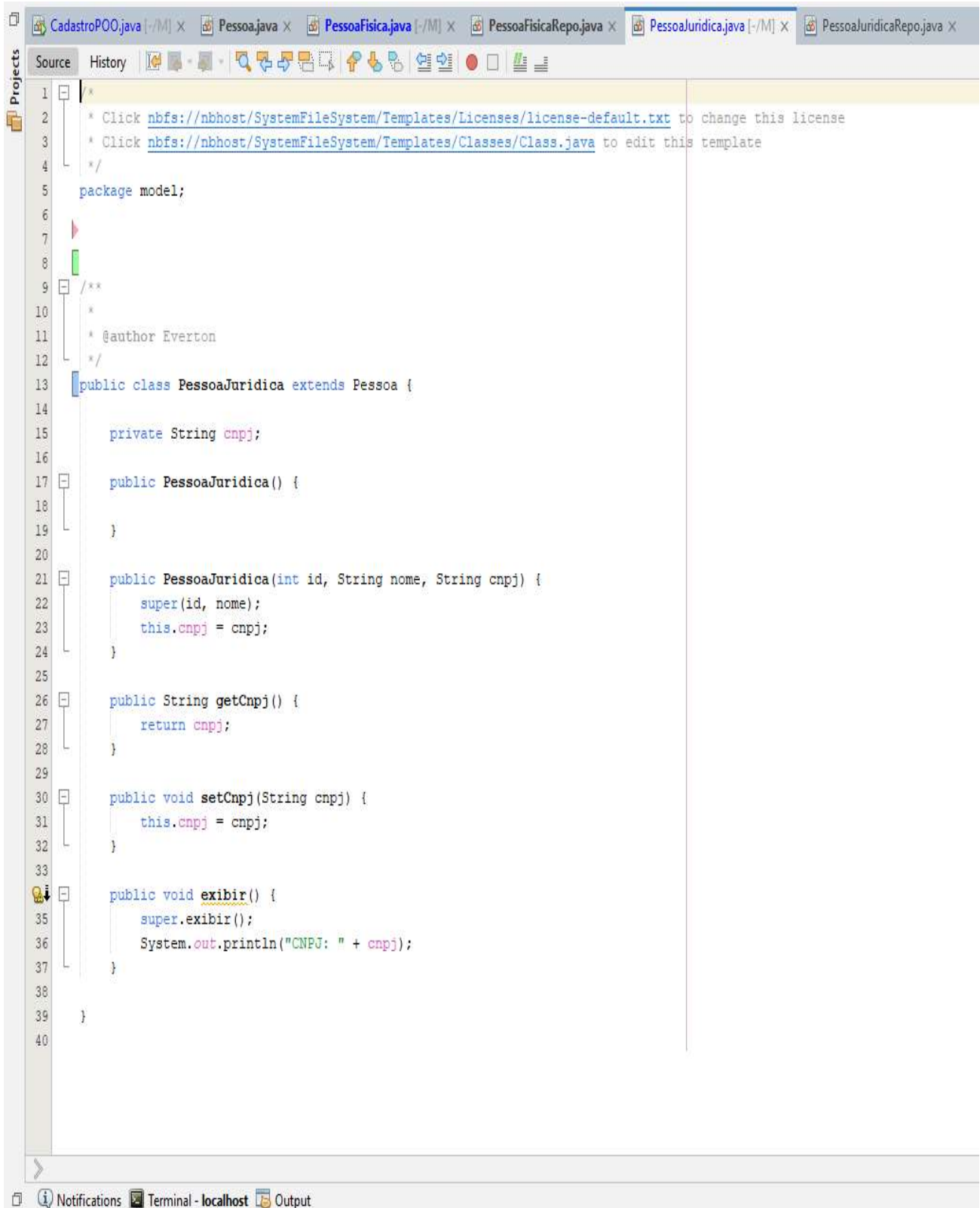
Notifications Terminal - localhost Output

PESSOA FISICA REPO.JAVA =



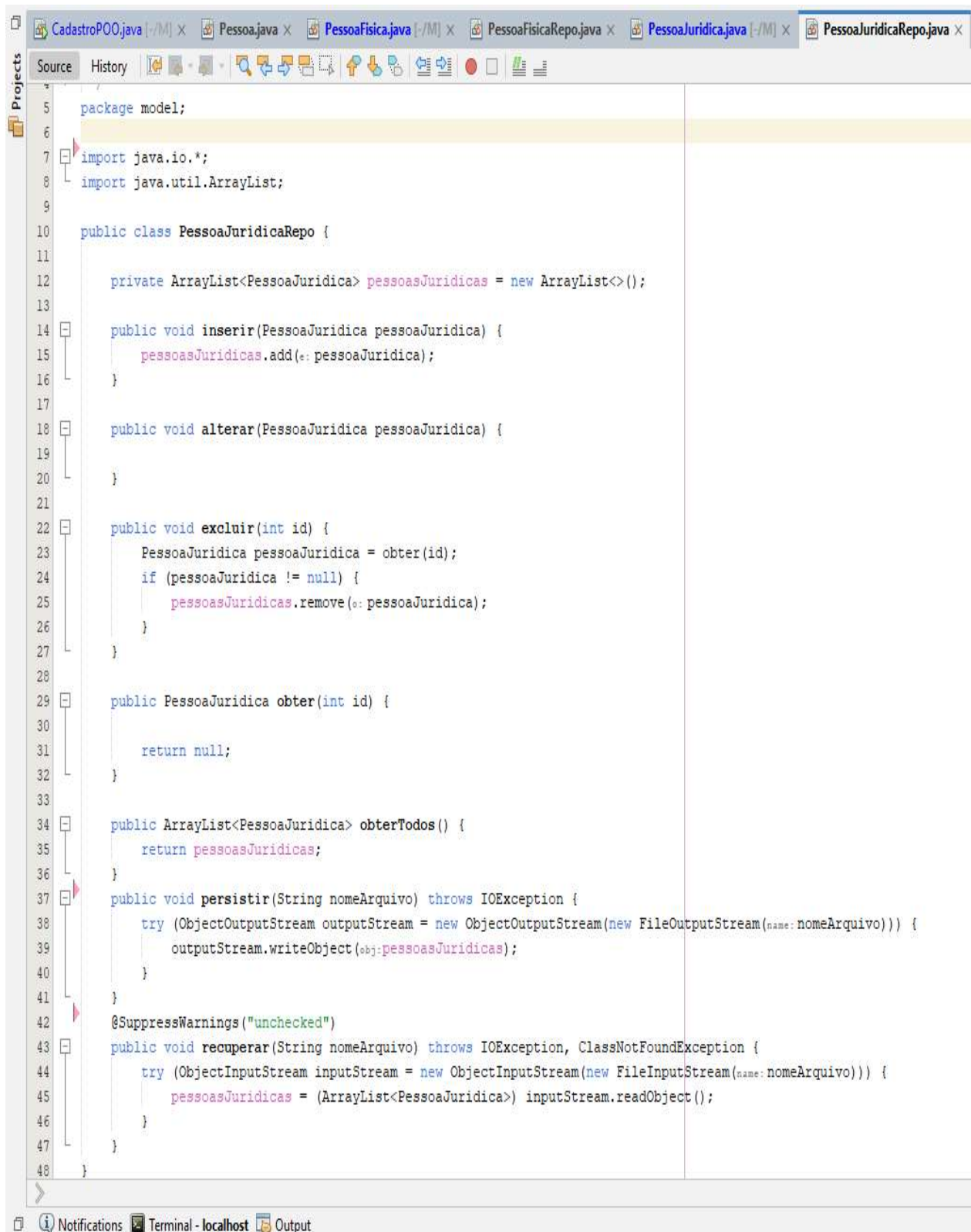
```
5 package model;
6
7 import java.io.*;
8 import java.util.ArrayList;
9
10 public class PessoaFisicaRepo {
11
12     private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();
13
14     public void inserir(PessoaFisica pessoaFisica) {
15         pessoasFisicas.add(pessoaFisica);
16     }
17
18     public void alterar(PessoaFisica pessoaFisica) {
19
20     }
21
22     public void excluir(int id) {
23         PessoaFisica pessoaFisica = obter(id);
24         if (pessoaFisica != null) {
25             pessoasFisicas.remove(pessoaFisica);
26         }
27     }
28
29     public PessoaFisica obter(int id) {
30         return null;
31     }
32
33     public ArrayList<PessoaFisica> obterTodos() {
34         return pessoasFisicas;
35     }
36
37     public void persistir(String nomeArquivo) throws IOException {
38         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
39             outputStream.writeObject(pessoasFisicas);
40         }
41     }
42
43     @SuppressWarnings("unchecked")
44     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
45         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
46             pessoasFisicas = (ArrayList<PessoaFisica>) inputStream.readObject();
47         }
48     }
49 }
```

PESSOA JURIDICA.JAVA =



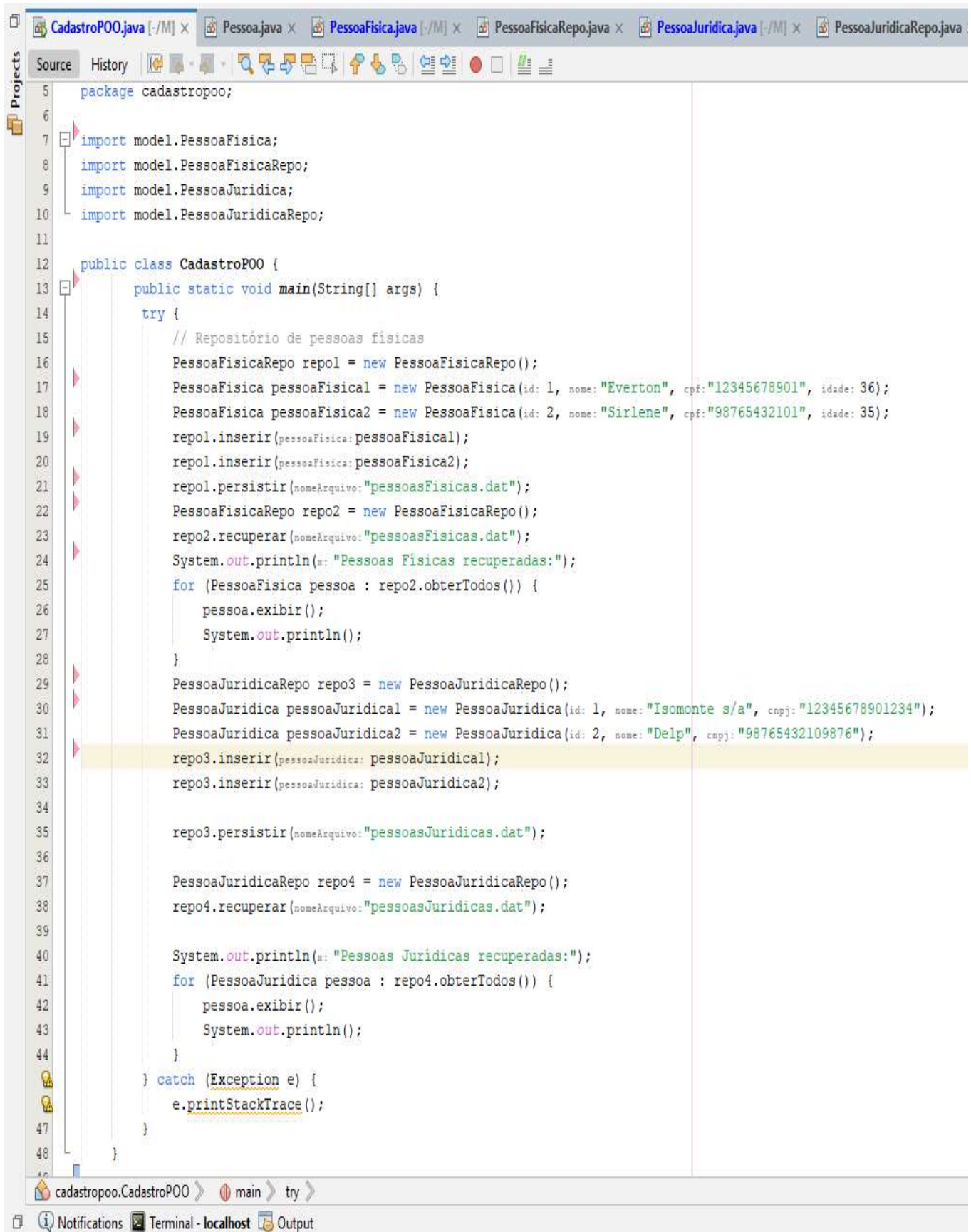
```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package model;
6
7
8
9   /**
10    *
11    * @author Everton
12    */
13   public class PessoaJuridica extends Pessoa {
14
15       private String cnpj;
16
17       public PessoaJuridica() {
18
19       }
20
21       public PessoaJuridica(int id, String nome, String cnpj) {
22           super(id, nome);
23           this.cnpj = cnpj;
24       }
25
26       public String getCnpj() {
27           return cnpj;
28       }
29
30       public void setCnpj(String cnpj) {
31           this.cnpj = cnpj;
32       }
33
34       public void exibir() {
35           super.exibir();
36           System.out.println("CNPJ: " + cnpj);
37       }
38
39   }
```


PESSOA JURÍDICA REPO.JAVA =



```
5 package model;
6
7 import java.io.*;
8 import java.util.ArrayList;
9
10 public class PessoaJuridicaRepo {
11
12     private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
13
14     public void inserir(PessoaJuridica pessoaJuridica) {
15         pessoasJuridicas.add(pessoaJuridica);
16     }
17
18     public void alterar(PessoaJuridica pessoaJuridica) {
19
20     }
21
22     public void excluir(int id) {
23         PessoaJuridica pessoaJuridica = obter(id);
24         if (pessoaJuridica != null) {
25             pessoasJuridicas.remove(pessoaJuridica);
26         }
27     }
28
29     public PessoaJuridica obter(int id) {
30
31         return null;
32     }
33
34     public ArrayList<PessoaJuridica> obterTodos() {
35         return pessoasJuridicas;
36     }
37
38     public void persistir(String nomeArquivo) throws IOException {
39         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
40             outputStream.writeObject(pessoasJuridicas);
41         }
42     }
43
44     @SuppressWarnings("unchecked")
45     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
46         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
47             pessoasJuridicas = (ArrayList<PessoaJuridica>) inputStream.readObject();
48         }
49     }
50 }
```


CADASTROPOO.JAVA =



```

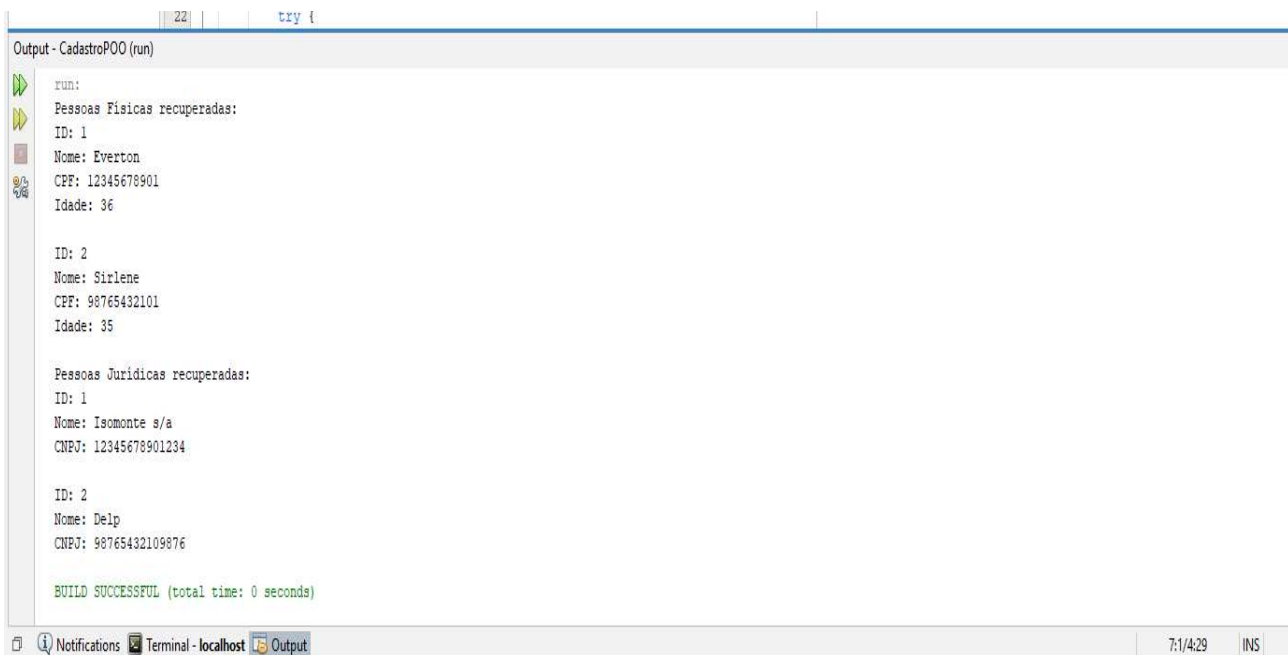
5  package cadastrapoo;
6
7  import model.PessoaFisica;
8  import model.PessoaFisicaRepo;
9  import model.PessoaJuridica;
10 import model.PessoaJuridicaRepo;
11
12 public class CadastroPOO {
13     public static void main(String[] args) {
14         try {
15             // Repositório de pessoas físicas
16             PessoaFisicaRepo repol = new PessoaFisicaRepo();
17             PessoaFisica pessoaFisical = new PessoaFisica(id: 1, nome: "Everton", cpf: "12345678901", idade: 36);
18             PessoaFisica pessoaFisica2 = new PessoaFisica(id: 2, nome: "Sirlene", cpf: "98765432101", idade: 35);
19             repol.inserir(pessoaFisica: pessoaFisical);
20             repol.inserir(pessoaFisica: pessoaFisica2);
21             repol.persistir(nomeArquivo: "pessoasFisicas.dat");
22             PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
23             repo2.recuperar(nomeArquivo: "pessoasFisicas.dat");
24             System.out.println("Pessoas Físicas recuperadas:");
25             for (PessoaFisica pessoa : repo2.obterTodos()) {
26                 pessoa.exibir();
27                 System.out.println();
28             }
29             PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
30             PessoaJuridica pessoaJuridical = new PessoaJuridica(id: 1, nome: "Isomonte s/a", cnpj: "12345678901234");
31             PessoaJuridica pessoaJuridica2 = new PessoaJuridica(id: 2, nome: "Delp", cnpj: "98765432109876");
32             repo3.inserir(pessoaJuridica: pessoaJuridical);
33             repo3.inserir(pessoaJuridica: pessoaJuridica2);
34
35             repo3.persistir(nomeArquivo: "pessoasJuridicas.dat");
36
37             PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
38             repo4.recuperar(nomeArquivo: "pessoasJuridicas.dat");
39
40             System.out.println("Pessoas Jurídicas recuperadas:");
41             for (PessoaJuridica pessoa : repo4.obterTodos()) {
42                 pessoa.exibir();
43                 System.out.println();
44             }
45         } catch (Exception e) {
46             e.printStackTrace();
47         }
48     }
49 }

```

cadastropoo.CadastroPOO > main > try >

Notifications Terminal - localhost Output

RESULTADO DA EXECUÇÃO:



```
Output - CadastroPOO (run)

run:
Pessoas Fisicas recuperadas:
ID: 1
Nome: Everton
CPF: 12345678901
Idade: 36

ID: 2
Nome: Sirlene
CPF: 98765432101
Idade: 35

Pessoas Juridicas recuperadas:
ID: 1
Nome: Isomonte s/a
CNPJ: 12345678901234

ID: 2
Nome: Delp
CNPJ: 98765432109876

BUILD SUCCESSFUL (total time: 0 seconds)
```

Caminho para o código = <https://github.com/Timhto/M3N1.git>

2º Procedimento | Criação do Cadastro em Modo Texto

Elementos estáticos:

Elementos estáticos em Java, como métodos e campos, propriedade à classe em si, em vez de pertencer a instâncias individuais dessa classe. Isso significa que eles são compartilhados por todas as instâncias da classe e podem ser acessados diretamente através do nome da classe, em vez de precisarem ser instanciados.

O método main é definido como estático porque ele serve como o ponto de entrada do programa, e o Java precisa invocá-lo antes de criar qualquer instância da classe. Isso permite que o método main seja chamado sem a necessidade de criar um objeto da classe principal.

Classe Scanner:

Uma classe Scanner em Java é usada para obter entradas do usuário a partir do console. Ela fornece métodos para ler diferentes tipos de dados, como inteiros, números de ponto flutuante, strings, entre outros, de forma interativa. O Scanner é frequentemente usado para interações com o usuário, como quando você precisa ler valores do teclado durante a execução de um programa.

Uso de classes de repositório:

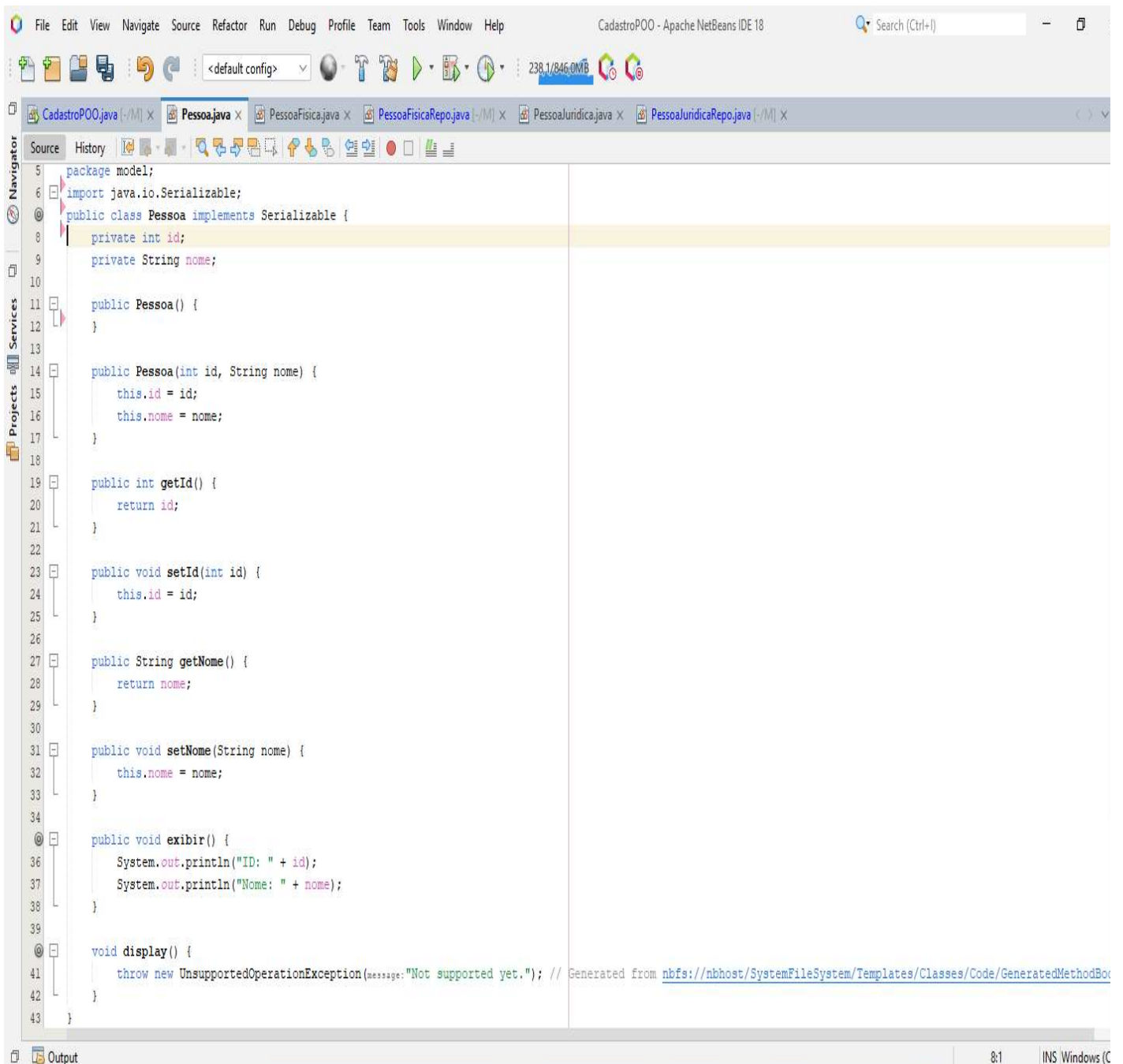
O uso de classes de repositório, como PessoaFisicaRepo e PessoaJuridicaRepo, teve um impacto significativo na organização do código. Essas classes foram criadas para encapsular a lógica de gerenciamento de dados e oferecer uma interface mais limpa e especializada para as operações relacionadas às entidades. Isso leva a uma separação de preocupações e promove a modularidade, o que torna o código mais legível, manutenível e escalável.

As classes de repositório também centralizam as operações de adição, exclusão, alteração e recuperação de dados relacionados a entidades específicas, o que simplifica o código na classe principal e evita duplicação de código. Isso torna mais fácil realizar mudanças ou adicionar novas funcionalidades no futuro.

Em resumo, as classes de repositório melhoram a organização do código, promovem a reutilização e tornam o processo de gerenciamento de dados mais seguro e modular.

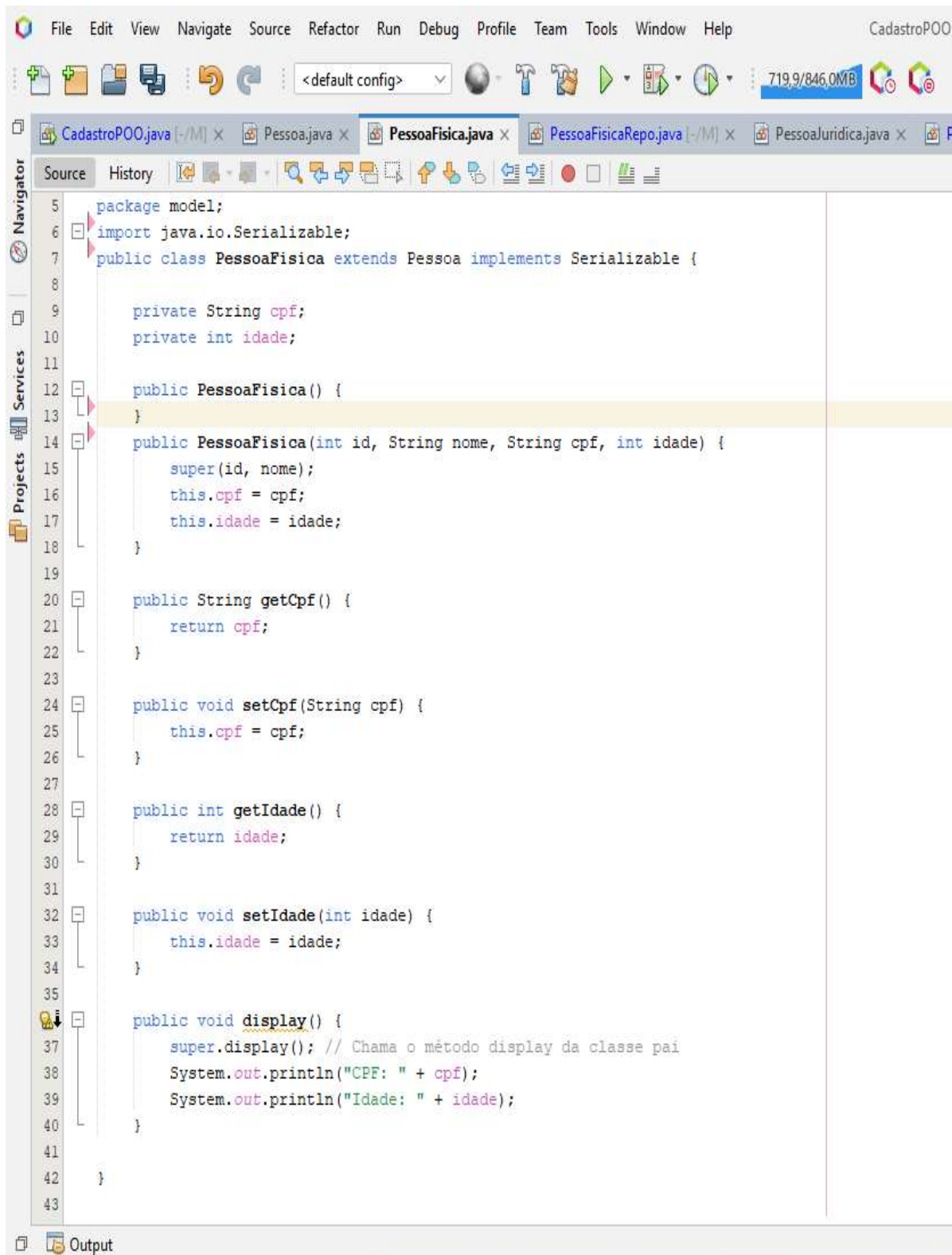
CÓDIGO DO SEGUNDO PROCEDIMENTO:

PESSOA.JAVA

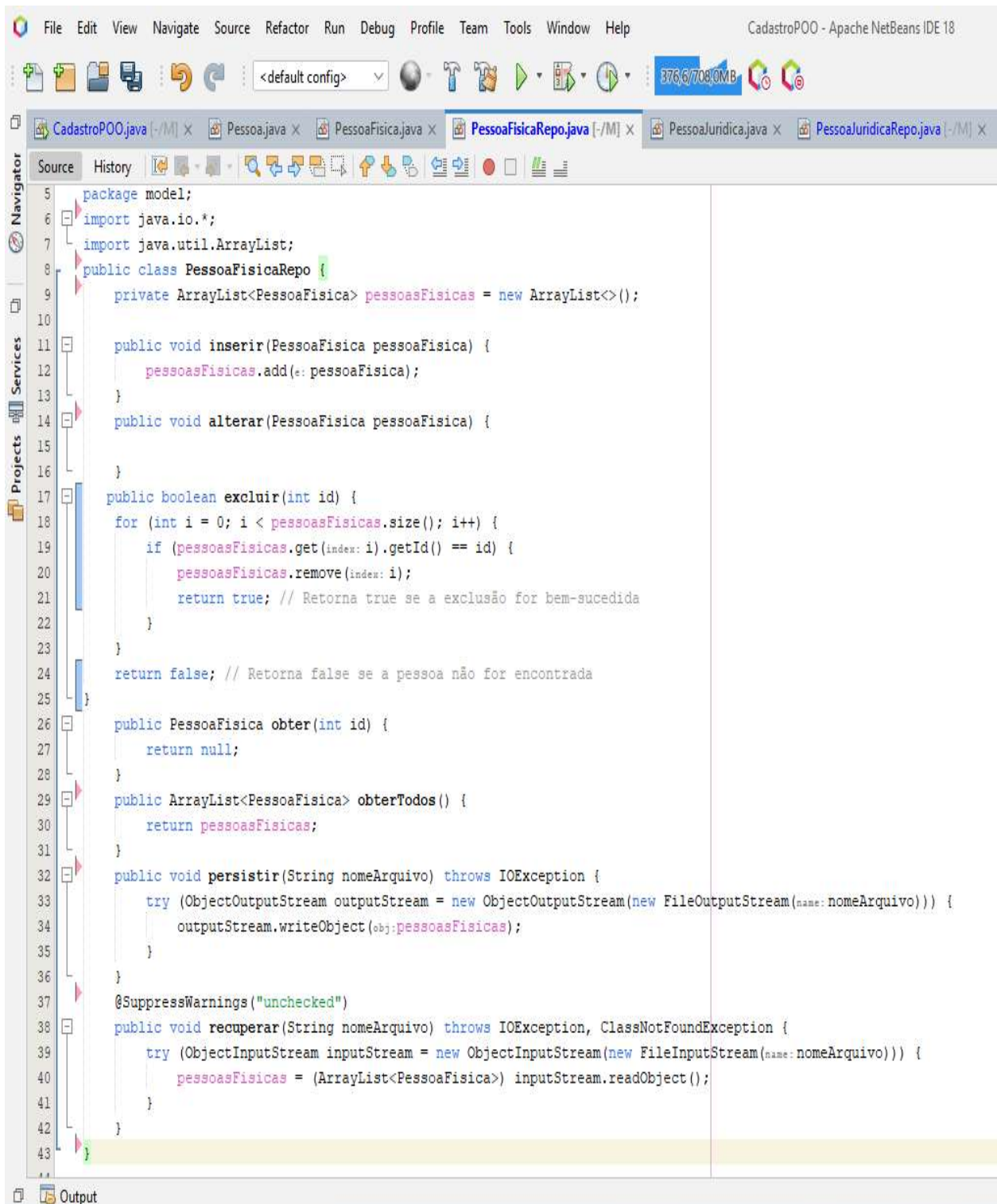


```
5 package model;
6 import java.io.Serializable;
7
8 public class Pessoa implements Serializable {
9     private int id;
10    private String nome;
11
12    public Pessoa() {
13    }
14
15    public Pessoa(int id, String nome) {
16        this.id = id;
17        this.nome = nome;
18    }
19
20    public int getId() {
21        return id;
22    }
23
24    public void setId(int id) {
25        this.id = id;
26    }
27
28    public String getNome() {
29        return nome;
30    }
31
32    public void setNome(String nome) {
33        this.nome = nome;
34    }
35
36    public void exibir() {
37        System.out.println("ID: " + id);
38        System.out.println("Nome: " + nome);
39    }
40
41    void display() {
42        throw new UnsupportedOperationException(message: "Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
43    }
44 }
```

PESSOAFISICA.JAVA

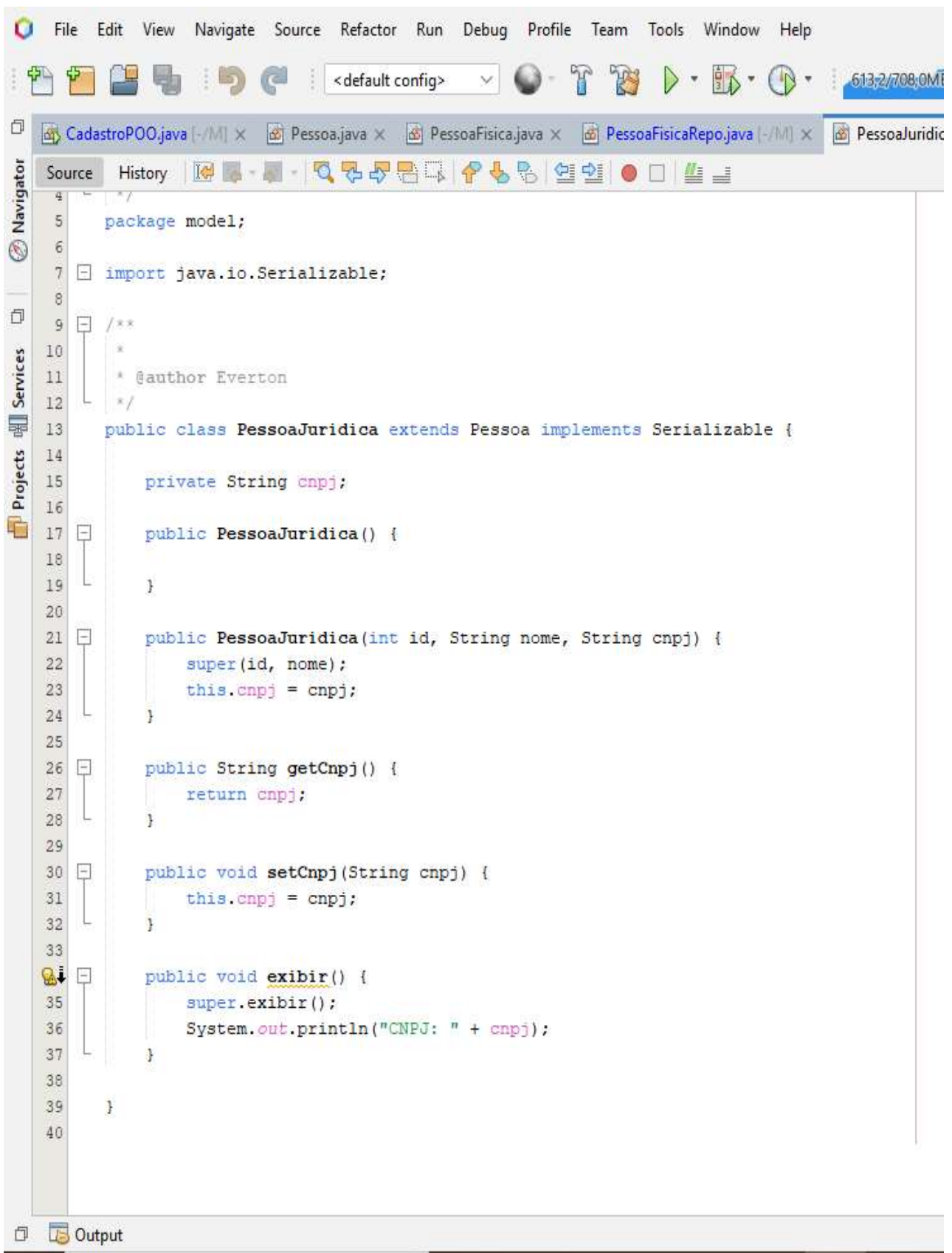


PESSOAFISICAREPO.JAVA

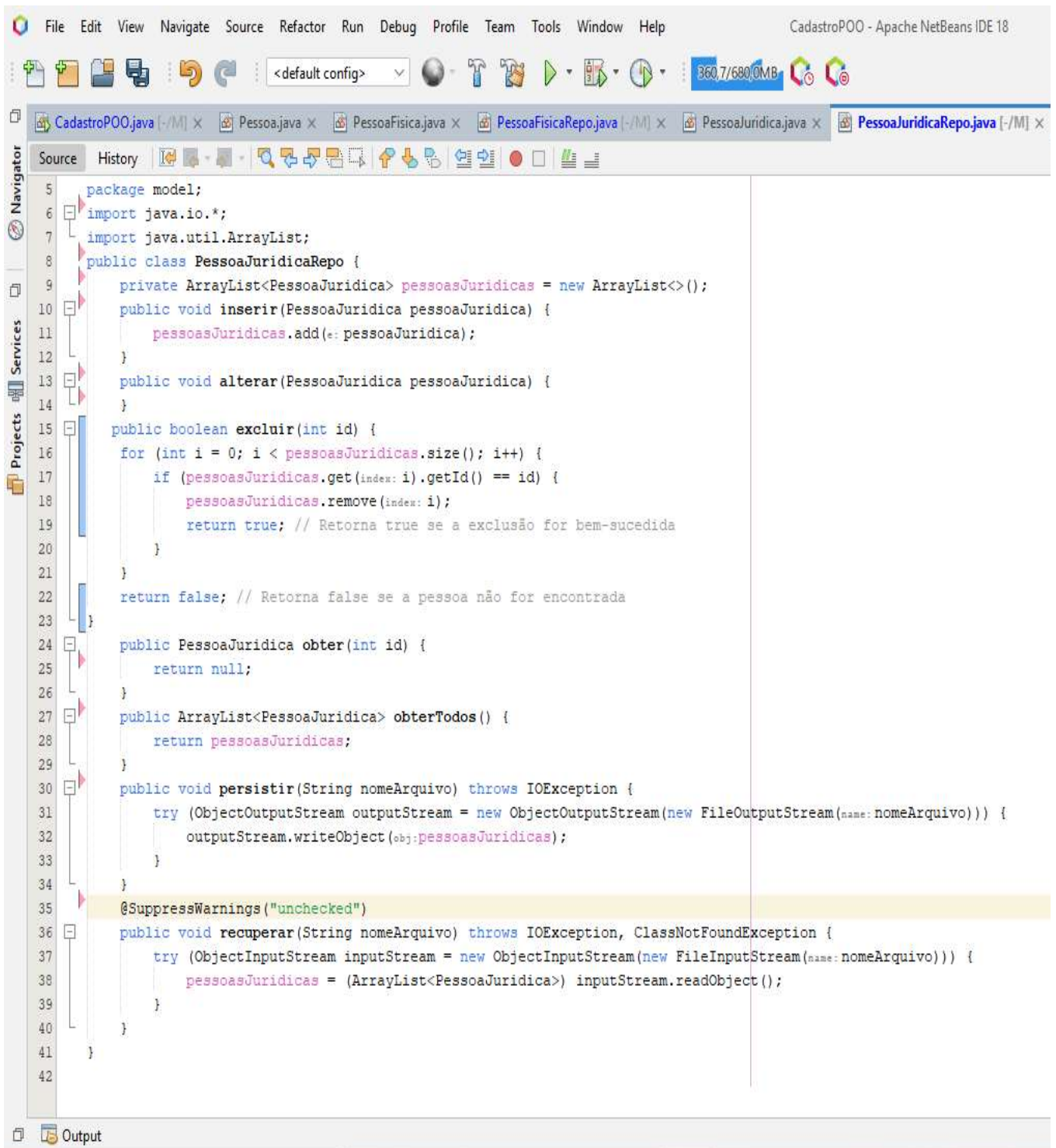


```
5 package model;
6 import java.io.*;
7 import java.util.ArrayList;
8 public class PessoaFisicaRepo {
9     private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();
10
11     public void inserir(PessoaFisica pessoaFisica) {
12         pessoasFisicas.add(pessoaFisica);
13     }
14     public void alterar(PessoaFisica pessoaFisica) {
15
16     }
17     public boolean excluir(int id) {
18         for (int i = 0; i < pessoasFisicas.size(); i++) {
19             if (pessoasFisicas.get(i).getId() == id) {
20                 pessoasFisicas.remove(i);
21                 return true; // Retorna true se a exclusão for bem-sucedida
22             }
23         }
24         return false; // Retorna false se a pessoa não for encontrada
25     }
26     public PessoaFisica obter(int id) {
27         return null;
28     }
29     public ArrayList<PessoaFisica> obterTodos() {
30         return pessoasFisicas;
31     }
32     public void persistir(String nomeArquivo) throws IOException {
33         try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
34             outputStream.writeObject(pessoasFisicas);
35         }
36     }
37     @SuppressWarnings("unchecked")
38     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
39         try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
40             pessoasFisicas = (ArrayList<PessoaFisica>) inputStream.readObject();
41         }
42     }
43 }
```


PESSOAJURIDICA.JAVA



PESSOAJURIDICAREPO.JAVA

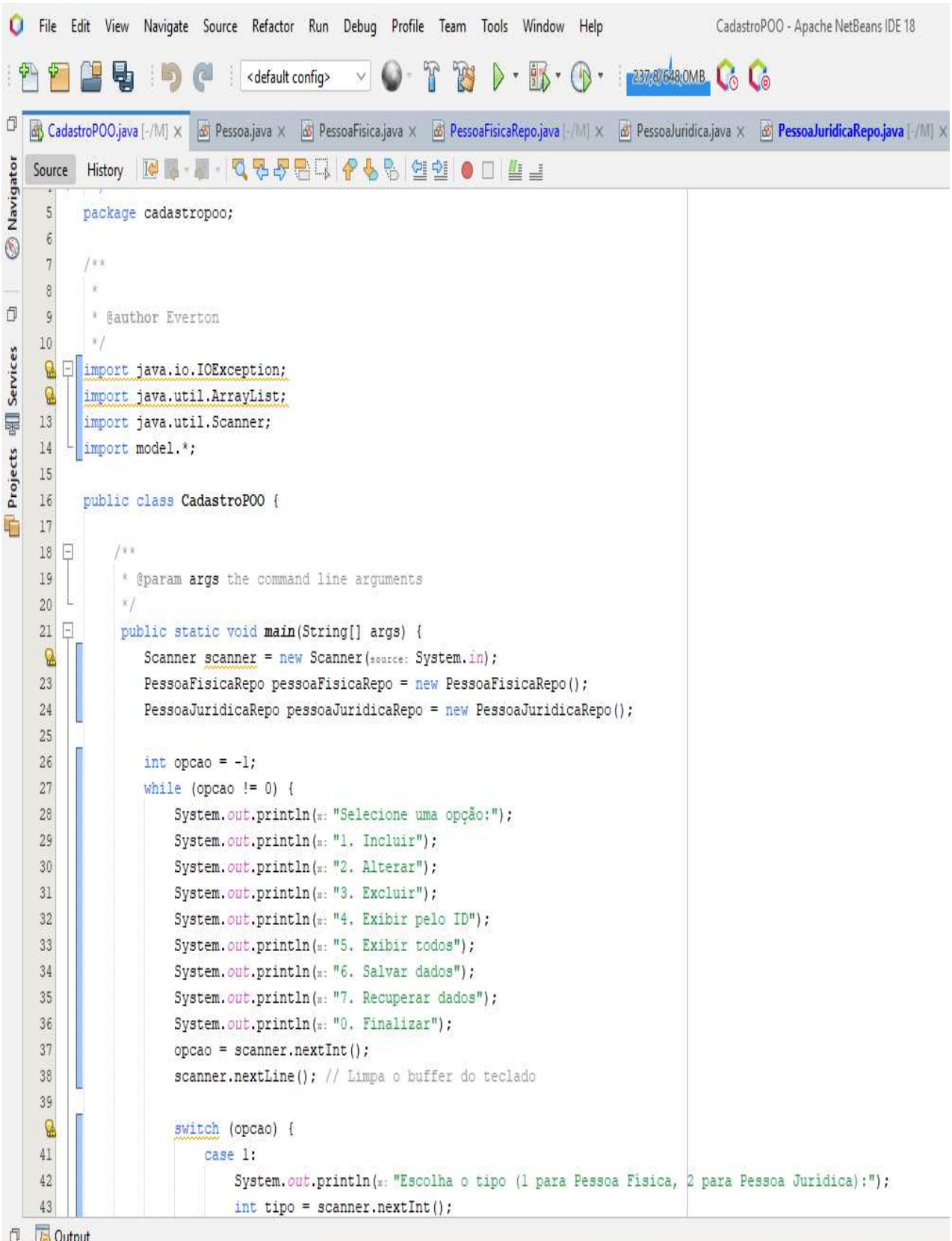


The screenshot displays the Apache NetBeans IDE with the following components:

- Menu Bar:** File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help.
- Toolbar:** Includes icons for file operations, running, and debugging. A status bar at the top right shows "CadastroPOO - Apache NetBeans IDE 18" and "360.7/680.0MB".
- Project Navigator:** Located on the left, it shows the project structure with files: CadastroPOO.java, Pessoa.java, PessoaFisica.java, PessoaFisicaRepo.java, PessoaJuridica.java, and PessoaJuridicaRepo.java.
- Source Editor:** The main area showing the code for `PessoaJuridicaRepo.java`. The code is as follows:

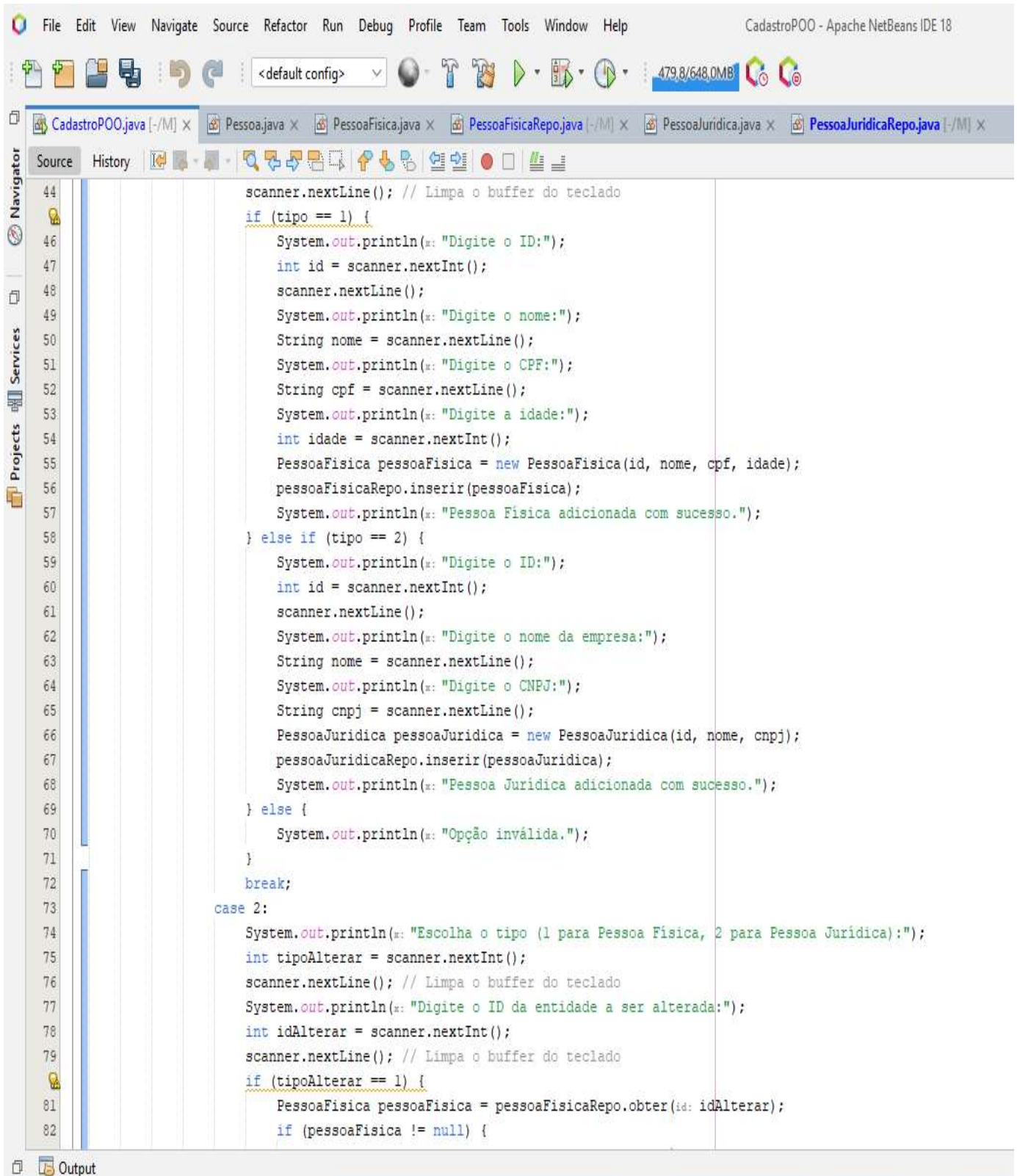
```
5 package model;
6 import java.io.*;
7 import java.util.ArrayList;
8 public class PessoaJuridicaRepo {
9     private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
10    public void inserir(PessoaJuridica pessoaJuridica) {
11        pessoasJuridicas.add(pessoaJuridica);
12    }
13    public void alterar(PessoaJuridica pessoaJuridica) {
14    }
15    public boolean excluir(int id) {
16        for (int i = 0; i < pessoasJuridicas.size(); i++) {
17            if (pessoasJuridicas.get(i).getId() == id) {
18                pessoasJuridicas.remove(i);
19                return true; // Retorna true se a exclusão for bem-sucedida
20            }
21        }
22        return false; // Retorna false se a pessoa não for encontrada
23    }
24    public PessoaJuridica obter(int id) {
25        return null;
26    }
27    public ArrayList<PessoaJuridica> obterTodos() {
28        return pessoasJuridicas;
29    }
30    public void persistir(String nomeArquivo) throws IOException {
31        try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
32            outputStream.writeObject(pessoasJuridicas);
33        }
34    }
35    @SuppressWarnings("unchecked")
36    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
37        try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
38            pessoasJuridicas = (ArrayList<PessoaJuridica>) inputStream.readObject();
39        }
40    }
41 }
42
```
- Output Window:** Located at the bottom, it is currently empty and labeled "Output".

CADASTROPOO.JAVA



The screenshot shows the Apache NetBeans IDE 18.0 interface. The title bar indicates the project is "CadastroPOO - Apache NetBeans IDE 18". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar contains various icons for file operations, running, and debugging. The sidebar on the left has three tabs: Navigator, Services, and Projects. The main editor window displays the source code of the file "CadastroPOO.java".

```
5 package cadastrupo;
6
7 /**
8  *
9  * @author Everton
10 */
11 import java.io.IOException;
12 import java.util.ArrayList;
13 import java.util.Scanner;
14 import model.*;
15
16 public class CadastroPOO {
17
18     /**
19      * @param args the command line arguments
20      */
21     public static void main(String[] args) {
22         Scanner scanner = new Scanner(System.in);
23         PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
24         PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();
25
26         int opcao = -1;
27         while (opcao != 0) {
28             System.out.println("Selecione uma opção:");
29             System.out.println("1. Incluir");
30             System.out.println("2. Alterar");
31             System.out.println("3. Excluir");
32             System.out.println("4. Exibir pelo ID");
33             System.out.println("5. Exibir todos");
34             System.out.println("6. Salvar dados");
35             System.out.println("7. Recuperar dados");
36             System.out.println("0. Finalizar");
37             opcao = scanner.nextInt();
38             scanner.nextLine(); // Limpa o buffer do teclado
39
40             switch (opcao) {
41                 case 1:
42                     System.out.println("Escolha o tipo (1 para Pessoa Fisica, 2 para Pessoa Juridica:");
43                     int tipo = scanner.nextInt();
```



File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

CadastroPOO - Apache NetBeans IDE 18

195.0/648.0MB

CadastroPOO.java [-/M] x Pessoa.java x PessoaFisica.java x PessoaFisicaRepo.java [-/M] x PessoaJuridica.java x PessoaJuridicaRepo.java [-/M]

Source History

```
83      System.out.println("Dados atuais da Pessoa Fisica:");
84      pessoaFisica.exibir();
85      System.out.println("Digite o novo nome:");
86      String novoNome = scanner.nextLine();
87      System.out.println("Digite o novo CPF:");
88      String novoCpf = scanner.nextLine();
89      System.out.println("Digite a nova idade:");
90      int novaIdade = scanner.nextInt();
91      pessoaFisica.setNome(nome: novoNome);
92      pessoaFisica.setCpf(cpf: novoCpf);
93      pessoaFisica.setIdade(idade: novaIdade);
94      System.out.println("Pessoa Fisica alterada com sucesso.");
95  } else {
96      System.out.println("Pessoa Fisica não encontrada.");
97  }
98  } else if (tipoAlterar == 2) {
99      PessoaJuridica pessoaJuridica = pessoaJuridicaRepo.obter(id: idAlterar);
100     if (pessoaJuridica != null) {
101         System.out.println("Dados atuais da Pessoa Juridica:");
102         pessoaJuridica.exibir();
103         System.out.println("Digite o novo nome da empresa:");
104         String novoNome = scanner.nextLine();
105         System.out.println("Digite o novo CNPJ:");
106         String novoCnpj = scanner.nextLine();
107         pessoaJuridica.setNome(nome: novoNome);
108         pessoaJuridica.setCnpj(cnpj: novoCnpj);
109         System.out.println("Pessoa Juridica alterada com sucesso.");
110     } else {
111         System.out.println("Pessoa Juridica não encontrada.");
112     }
113 } else {
114     System.out.println("Opção inválida.");
115 }
116 break;
117 case 3:
118     System.out.println("Escolha o tipo (1 para Pessoa Fisica, 2 para Pessoa Juridica):");
119     int tipoExcluir = scanner.nextInt();
120     scanner.nextLine(); // Limpa o buffer do teclado
121     System.out.println("Digite o ID da entidade a ser excluída:");
```

Output

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help CadastroPOO - Apache NetBeans IDE 18

529.9/648.0MB

CadastroPOO.java [-/M] x Pessoa.java x PessoaFisica.java x PessoaFisicaRepo.java [-/M] x PessoaJuridica.java x PessoaJuridicaRepo.java [-/M] x

Source History

```
122     int idExcluir = scanner.nextInt();
123     scanner.nextLine(); // Limpa o buffer do teclado
124     boolean excluido = false;
125     if (tipoExcluir == 1) {
126         excluido = pessoaFisicaRepo.excluir(id: idExcluir);
127     } else if (tipoExcluir == 2) {
128         excluido = pessoaJuridicaRepo.excluir(id: idExcluir);
129     } else {
130         System.out.println(x: "Opção inválida.");
131     }
132     if (excluido) {
133         System.out.println(x: "Entidade excluída com sucesso.");
134     } else {
135         System.out.println(x: "Entidade não encontrada.");
136     }
137     break;
138     // Demais casos omitidos para evitar superar o limite de caracteres
139     // ...
140     case 0:
141         System.out.println(x: "Finalizando o programa.");
142         break;
143     default:
144         System.out.println(x: "Opção inválida.");
145 }
146 }
147 scanner.close();
148 }
149 }
```

Projects Services

Output

RESULTADO:

The screenshot displays the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar shows various icons for file operations and running the application. The main editor area shows the source code of 'CadastroPOO.java' with the following snippet:

```
122 int idExcluir = scanner.nextInt();
123 scanner.nextLine(); // Limpa o buffer do teclado
124 boolean excluido = false;
125 if (tipoExcluir == 1) {
126     excluido = pessoaFisicaRepo.excluir(id: idExcluir);
```

The 'Output - CadastroPOO (run)' window shows the execution of the program, displaying a menu of options and user input for adding a physical person:

```
run:
Selecione uma opção:
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar
1
Escolha o tipo (1 para Pessoa Física, 2 para Pessoa Jurídica):
1
Digite o ID:
120
Digite o nome:
EVERTON
Digite o CPF:
12345678910
Digite a idade:
36
Pessoa Física adicionada com sucesso.
Selecione uma opção:
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar
```

Caminho para o código = <https://github.com/Timhto/M3N1.git>