

**Universidade:** Estácio de Sá

**Campus:** Estácio Interlagos

**Curso:** Desenvolvimento Full Stack

**Disciplina:** Iniciando o caminho pelo Java

**Número da Turma:** 2023.3

**Semestre Letivo:** 3º Semestre

**Integrantes da Prática:** Lucas Henrique Silva Santos

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

**Título da Prática:** Criação das Entidades e Sistema de Persistência

**Objetivo da Prática:**

1. Utilizar herança e polimorfismo na definição de entidades.
2. Utilizar persistência de objetos em arquivos binários.
3. Implementar uma interface cadastral em modo texto.
4. Utilizar o controle de exceções da plataforma Java.
5. No final do projeto, o aluno terá implementado um sistema cadastral em Java, utilizando os recursos da programação orientada a objetos e a persistência em arquivos binários.

**Códigos da Prática**

Este relatório descreve o desenvolvimento de um sistema cadastral em Java, no qual foram aplicados os conceitos de herança, polimorfismo, persistência de objetos em arquivos binários e a implementação de uma interface cadastral em modo texto. O projeto foi criado como parte de um exercício acadêmico com o objetivo de praticar programação orientada a objetos e persistência de dados em Java.

**1º Procedimento | Criação das Entidades e Sistema de Persistência**

**Herança e Polimorfismo:** No projeto, utilizei herança para criar uma hierarquia de classes que representam entidades no sistema cadastral. A classe base

Pessoa possui os atributos comuns a todas as entidades, como id e nome. As classes PessoaFisica e PessoaJuridica herdam de Pessoa e adicionam campos específicos, como cpf e idade para pessoas físicas e cnpj para pessoas jurídicas. O polimorfismo foi aplicado no método exibir, que é polimórfico nas classes PessoaFisica e PessoaJuridica.

**Persistência em Arquivos Binários:** Para persistir objetos no sistema, implementei repositórios separados para PessoaFisica e PessoaJuridica. Esses repositórios utilizam arquivos binários para armazenar e recuperar os dados das entidades. O método persistir grava os objetos em um arquivo binário, enquanto o método recuperar lê os objetos do arquivo binário e os carrega de volta para a memória.

**Interface Cadastral em Modo Texto:** Desenvolvi uma interface cadastral em modo texto que permite ao usuário interagir com o sistema de forma intuitiva. O programa exibe um menu de opções, incluindo inserção, alteração, exclusão e consulta de entidades. O usuário pode escolher o tipo de entidade (Pessoa Física ou Pessoa Jurídica), fornecer os dados solicitados e realizar as operações desejadas.

**Controle de Exceções em Java:** Implementei tratamento de exceções para lidar com situações imprevistas durante a execução, garantindo que o programa não quebre inesperadamente.

**Resultados da Execução:** Durante os testes, o sistema demonstrou eficácia na criação, edição, exclusão e consulta de entidades. Os dados foram corretamente persistidos em arquivos binários, permitindo a recuperação após a reinicialização do programa. O controle de exceções garantiu uma experiência de usuário robusta e livre de falhas.

Este projeto nos proporcionou uma valiosa experiência de desenvolvimento em Java e a oportunidade de aplicar conceitos teóricos em um contexto prático.

## 2º Procedimento | Criação do Cadastro em Modo Texto

### Implementação da Interface Cadastral em Modo Texto

A segunda fase do projeto envolveu a criação de uma interface de usuário em modo texto, que permite ao usuário interagir com o sistema de forma amigável e intuitiva. O programa oferece um menu de opções que permite ao usuário executar diversas operações no sistema. Aqui estão as principais funcionalidades implementadas:

**Incluir:** O usuário pode adicionar novas entidades ao sistema, escolhendo entre Pessoa Física e Pessoa Jurídica e fornecendo os dados necessários. Esses dados são validados para garantir a integridade das informações.

**Alterar:** É possível modificar os dados de uma entidade existente, identificando-a pelo ID. O programa exibe os dados atuais e solicita novas informações ao usuário.

**Excluir:** O sistema permite a exclusão de uma entidade específica pelo seu ID, evitando a exclusão acidental.

**Consultar por ID:** O usuário pode consultar uma entidade pelo seu ID, e o programa exibe os detalhes dessa entidade.

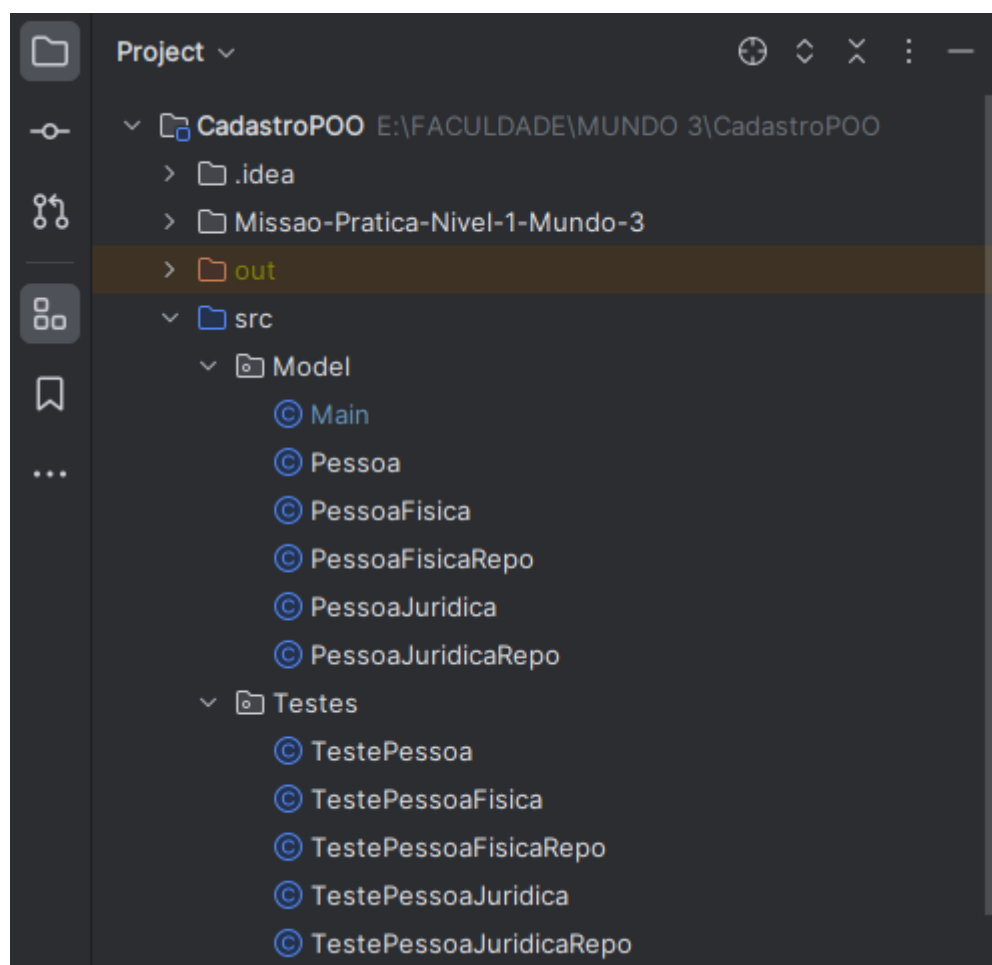
**Listar Todas as Entidades:** O sistema permite listar todas as entidades cadastradas no sistema, seja Pessoa Física ou Pessoa Jurídica.

**Salvar e Recuperar Dados:** Implementei opções para salvar os dados em arquivos binários, permitindo que as informações sejam mantidas entre as execuções do programa. Também é possível recuperar os dados a partir desses arquivos.

## Tratamento de Exceções

Durante a implementação da interface cadastral, foi dado destaque ao tratamento de exceções. Isso garante que o programa funcione de maneira robusta, mesmo em situações de erro. Foram implementados blocos try-catch para capturar exceções e fornecer mensagens de erro adequadas ao usuário, evitando falhas inesperadas no programa.

## RESULTADO DOS CÓDIGOS



## Pessoa:

```
© Pessoa.java ×
1 package Model;
2
3 import java.io.Serializable;
4
5 6 usages 2 inheritors 1 Lucas Henrique
6 public class Pessoa implements Serializable {
7     3 usages
8     private String nome;
9     private int id;
10
11     4 usages 1 Lucas Henrique
12     public Pessoa(){
13     }
14
15     3 usages 1 Lucas Henrique
16     public Pessoa(String nome, int id) {
17         this.nome = nome;
18         this.id = id;
19     }
20
21     1 usage 1 Lucas Henrique
22     public String getNome() { return nome; }
23
24     11 usages 1 Lucas Henrique
25     public void setNome(String nome) { this.nome = nome; }
26
27     5 usages 1 Lucas Henrique
28     public int getId() { return id; }
29
30     11 usages 1 Lucas Henrique
31     public void setId(int id) { this.id = id; }
32
33     2 overrides 1 Lucas Henrique
34     public void exibir() {
35         System.out.println("Nome " + getNome());
36         System.out.println("Id " + getId());
37     }
38 }
```

## Pessoa Física:

© PessoaFisica.java ×

```
1 package Model;
2
3 import java.io.Serializable;
4
5 29 usages  🧑 Lucas Henrique
6 public class PessoaFisica extends Pessoa implements Serializable {
7     3 usages
8     private String cpf;
9     3 usages
10    private int idade;
11
12    5 usages  🧑 Lucas Henrique
13    public PessoaFisica(){
14    }
15
16    2 usages  🧑 Lucas Henrique
17    public PessoaFisica(String nome, int id, String cpf, int idade) {
18        super(nome, id);
19        this.cpf = cpf;
20        this.idade = idade;
21    }
22
23    1 usage  🧑 Lucas Henrique
24    public String getCpf() { return cpf; }
25
26    5 usages  🧑 Lucas Henrique
27    public void setCpf(String cpf) { this.cpf = cpf; }
28
29    1 usage  🧑 Lucas Henrique
30    public int getIdade() { return idade; }
31
32    5 usages  🧑 Lucas Henrique
33    public void setIdade(int idade) { this.idade = idade; }
34
35    🧑 Lucas Henrique
36    @Override
37    public void exibir() {
38        super.exibir();
39        System.out.println("Cpf " + getCpf());
40        System.out.println("Idade " + getIdade());
41    }
42 }
```

## Pessoa Física Repositório:

```
© PessoaFisicaRepo.java ×
1  package Model;
2
3  > import ...
7
5 usages  📄 Lucas Henrique
8  public class PessoaFisicaRepo {
9      9 usages
10     private List<PessoaFisica> personasFisicas = new ArrayList<>();
11
12     5 usages  📄 Lucas Henrique
13     > public void inserir(PessoaFisica pessoaFisica) { personasFisicas.add(pessoaFisica); }
14
15     2 usages  📄 Lucas Henrique
16     @ > public void alterar(PessoaFisica pessoaFisica, PessoaFisica novaPessoa) {
17         int index = getIndexById(pessoaFisica.getId());
18         if (index != -1) {
19             personasFisicas.set(index, novaPessoa);
20         }
21     }
22
23     2 usages  📄 Lucas Henrique
24     > public void excluir(int id) {
25         int index = getIndexById(id);
26         if (index != -1) {
27             personasFisicas.remove(index);
28         }
29     }
30
31     4 usages  📄 Lucas Henrique
32     > public PessoaFisica obter(int id) {
33         int index = getIndexById(id);
34
35         if (index != -1) {
36             // personasFisicas.get(index).exibir();
37             return personasFisicas.get(index);
38         } else {
39             //System.out.println("Pessoa Física com o ID " + id + " não foi encontrada.");
40             return null;
41         }
42     }
43 }
```

```

41 > 2 usages  Lucas Henrique
    public List<PessoaFisica> obterTodos() { return new ArrayList<>(pessoasFisicas); }
44
    2 usages  Lucas Henrique
45 public void persistir(String nomeArquivo) throws IOException {
46     try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
47         outputStream.writeObject(pessoasFisicas);
48     }
49 }
50
    2 usages  Lucas Henrique
51 public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
52     try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
53         pessoasFisicas = (List<PessoaFisica>) inputStream.readObject();
54     }
55 }
56
    3 usages  Lucas Henrique
57 private int getIndexById(int id) {
58     return IntStream.range(0, pessoasFisicas.size()) IntStream
59         .filter(i -> pessoasFisicas.get(i).getId() == id)
60         .findFirst() OptionalInt
61         .orElse(other: -1);
62 }
63 }

```



## Pessoa Jurídica:

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

## Pessoa Jurídica Repositório:

```
1 package Model;
2
3 > import ...
7
5 usages  Lucas Henrique
8 public class PessoaJuridicaRepo {
9     9 usages
10     private List<PessoaJuridica> personasJuridicas = new ArrayList<>();
11
12     5 usages  Lucas Henrique
13     > public void inserir(PessoaJuridica pessoaJuridica) { personasJuridicas.add(pessoaJuridica); }
14
15     2 usages  Lucas Henrique
16     @ public void alterar(PessoaJuridica pessoaJuridica, PessoaJuridica novaPessoa) {
17         int index = getIndexById(pessoaJuridica.getId());
18         if (index != -1) {
19             personasJuridicas.set(index, novaPessoa);
20         }
21     }
22
23     2 usages  Lucas Henrique
24     public void excluir(int id) {
25         int index = getIndexById(id);
26         if (index != -1) {
27             personasJuridicas.remove(index);
28         }
29     }
30
31     4 usages  Lucas Henrique
32     public PessoaJuridica obter(int id) {
33         int index = getIndexById(id);
34
35         if (index != -1) {
36             // personasJuridicas.get(index).exibir();
37             return personasJuridicas.get(index);
38         } else {
39             //System.out.println("Pessoa Juridica com o ID " + id + " não foi encontrada.");
40             return null;
41         }
42     }
43 }
```

```

41 > public List<PessoaJuridica> obterTodos() { return new ArrayList<>(pessoasJuridicas); }
44
2 usages Lucas Henrique
45 public void persistir(String nomeArquivo) throws IOException {
46     try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
47         outputStream.writeObject(pessoasJuridicas);
48     }
49 }
50
2 usages Lucas Henrique
51 public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
52     try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
53         pessoasJuridicas = (List<PessoaJuridica>) inputStream.readObject();
54     }
55 }
56
3 usages Lucas Henrique
57 private int getIndexById(int id) {
58     return IntStream.range(0, pessoasJuridicas.size()) IntStream
59         .filter(i -> pessoasJuridicas.get(i).getId() == id)
60         .findFirst() OptionalInt
61         .orElse( other: -1);
62 }
63 }

```

## TESTES REALIZADOS






### Teste Pessoa:

```

© TestePessoa.java x
1 package Testes;
2
3 import Model.Pessoa;
4
5 public class TestePessoa {
6     public static void main(String[] args) {
7         Pessoa pessoa = new Pessoa();
8         pessoa.setNome("João");
9         pessoa.setId(123);
10        pessoa.exibir();
11    }
12 }
13

```

## Teste Pessoa Física:

```
© TestePessoaFisica.java ×  
1 package Testes;  
2  
3 import Model.PessoaFisica;  
4  
5  Lucas Henrique  
6  public class TestePessoaFisica {  
7  Lucas Henrique  
8  public static void main(String[] args) {  
9     PessoaFisica pessoaFisica = new PessoaFisica();  
10    pessoaFisica.setNome("Lucas");  
11    pessoaFisica.setId(20);  
12    pessoaFisica.setCpf("123487985");  
13     pessoaFisica.setIdade(26);  
14    pessoaFisica.exibir();  
15 }  
16 }
```

## Teste Pessoa Física Repositório:

```
© TestePessoaFisicaRepo.java x
1 package Testes;
2
3 > import ...
7
8 ▶ public class TestePessoaFisicaRepo {
9 ▶   ▶ public static void main(String[] args) {
10       PessoaFisicaRepo repositorio = new PessoaFisicaRepo();
11       PessoaFisica pessoa = new PessoaFisica();
12       PessoaFisica pessoa1 = new PessoaFisica();
13       PessoaFisica pessoa2 = new PessoaFisica();
14       PessoaFisica pessoa3 = new PessoaFisica();
15       pessoa.setNome("João");
16       pessoa.setId(10);
17       pessoa.setIdade(22);
18       pessoa.setCpf("123456789");
19       pessoa1.setNome("Pedro");
20       pessoa1.setId(11);
21       pessoa1.setIdade(23);
22       pessoa1.setCpf("987654321");
23       pessoa2.setNome("Maria");
24       pessoa2.setId(12);
25       pessoa2.setIdade(24);
26       pessoa2.setCpf("123456");
27       pessoa3.setNome("Marcos");
28       pessoa3.setId(3);
29       pessoa3.setIdade(40);
30       pessoa3.setCpf("654321");
31       repositorio.inserir(pessoa);
32       repositorio.inserir(pessoa1);
33       repositorio.inserir(pessoa2);
34       repositorio.inserir(pessoa3);
35       repositorio.alterar(pessoa,pessoa1);
36       repositorio.excluir(id: 12);
37       repositorio.obter(id: 3);
38
```

```

39         try {
40             System.out.println("Salvo com sucesso");
41             repositório.persistir( nomeArquivo: "pessoasFisicas.bin");
42         } catch (IOException e) {
43             e.printStackTrace();
44         }
45
46         try {
47             repositório.recuperar( nomeArquivo: "pessoasFisicas.bin");
48             System.out.println("Pessoas Físicas recuperadas:");
49             repositório.obterTodos().forEach(PessoaFisica::exibir);
50         } catch (IOException | ClassNotFoundException e) {
51             throw new RuntimeException(e);
52         }
53     }
54 }
55 }

```

## Teste Pessoa Jurídica:

© TestePessoaJuridica.java ×

```

1  package Testes;
2
3  import Model.PessoaJuridica;
4
5  Lucas Henrique
6  ▶ public class TestePessoaJuridica {
7      Lucas Henrique
8      ▶ public static void main(String[] args) {
9          PessoaJuridica pessoaJuridica = new PessoaJuridica();
10         pessoaJuridica.setNome("Mestre Kame");
11         pessoaJuridica.setId(255);
12         pessoaJuridica.setCnpj("21511248");
13         pessoaJuridica.exibir();
14     }
15 }

```

## Teste Pessoa Jurídica Repositório:

```
© TestePessoaJuridicaRepo.java ×
1 package Testes;
2
3 > import ...
7
  Lucas Henrique
8 ▶ public class TestePessoaJuridicaRepo {
  Lucas Henrique
9 ▶   public static void main(String[] args) {
10     PessoaJuridicaRepo repositorio = new PessoaJuridicaRepo();
11     PessoaJuridica pessoa = new PessoaJuridica();
12     PessoaJuridica pessoa1 = new PessoaJuridica();
13     PessoaJuridica pessoa2 = new PessoaJuridica();
14     PessoaJuridica pessoa3 = new PessoaJuridica();
15     pessoa.setNome("João");
16     pessoa.setId(10);
17     pessoa.setCnpj("123456");
18     pessoa1.setNome("Pedro");
19     pessoa1.setId(11);
20     pessoa1.setCnpj("654321");
21     pessoa2.setNome("Maria");
22     pessoa2.setId(12);
23     pessoa2.setCnpj("123456789");
24     pessoa3.setNome("Marcos");
25     pessoa3.setId(3);
26     pessoa3.setCnpj("321456");
27     repositorio.inserir(pessoa);
28     repositorio.inserir(pessoa1);
29     repositorio.inserir(pessoa2);
30     repositorio.inserir(pessoa3);
31     repositorio.alterar(pessoa,pessoa1);
32     repositorio.excluir(id: 12);
33     repositorio.obter(id: 3);
```

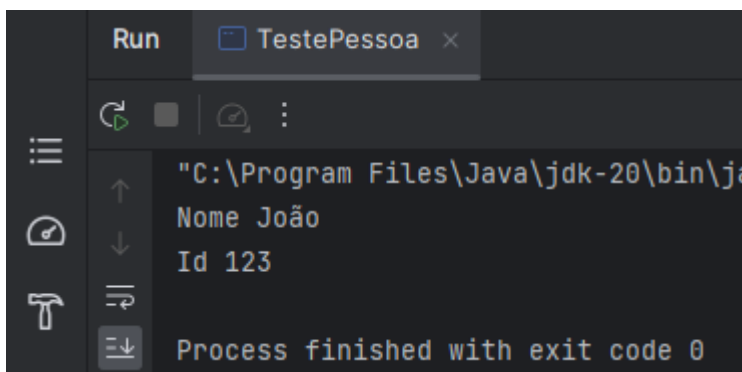
```

35         try {
36             System.out.println("Salvo com sucesso");
37             repositorio.persistir( nomeArquivo: "pessoasJuridicas.dat");
38         } catch (IOException e) {
39             e.printStackTrace();
40         }
41
42         try {
43             repositorio.recuperar( nomeArquivo: "pessoasJuridicas.dat");
44             System.out.println("Pessoas Jurídicas recuperadas:");
45             repositorio.obterTodos().forEach(PessoaJuridica::exibir);
46         } catch (IOException | ClassNotFoundException e) {
47             throw new RuntimeException(e);
48         }
49     }
50 }
51 }

```

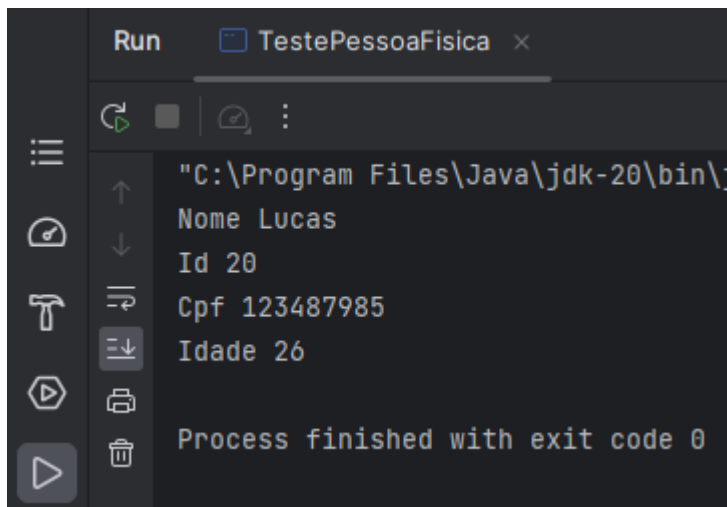
## RESULTADOS NO CONSOLE

### Run - Teste Pessoa Física:





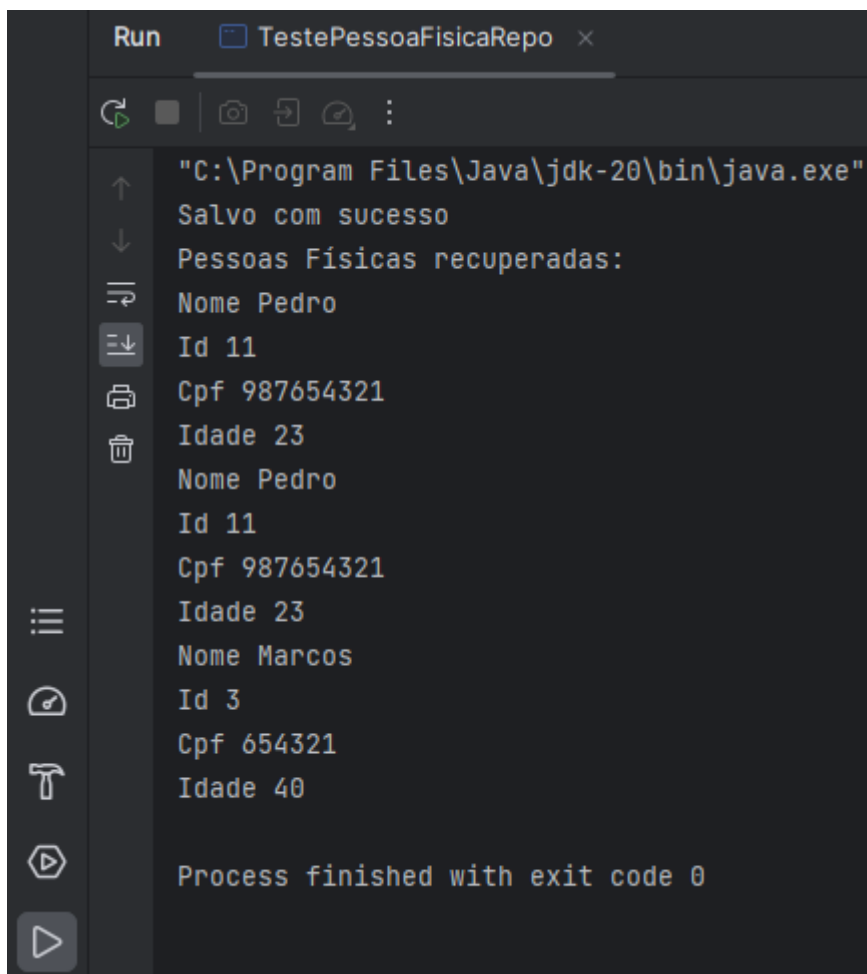
## Run - Teste Pessoa Física:



The screenshot shows the Run console for a Java application named 'TestePessoaFisica'. The console output displays the execution path and the details of a physical person object created during the test.

```
"C:\Program Files\Java\jdk-20\bin\java.exe"  
Nome Lucas  
Id 20  
Cpf 123487985  
Idade 26  
  
Process finished with exit code 0
```

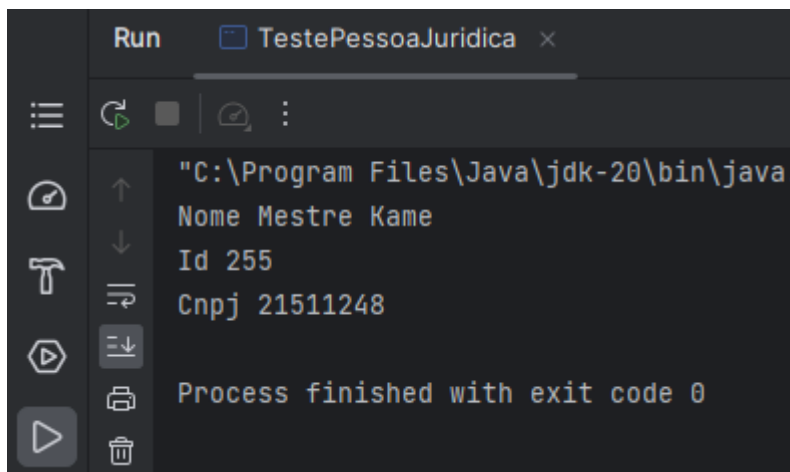
## Run - Teste Pessoa Física Repositório:



The screenshot shows the Run console for a Java application named 'TestePessoaFisicaRepo'. The console output displays the execution path, a success message, and a list of retrieved physical persons from a repository.

```
"C:\Program Files\Java\jdk-20\bin\java.exe"  
Salvo com sucesso  
Pessoas Físicas recuperadas:  
Nome Pedro  
Id 11  
Cpf 987654321  
Idade 23  
Nome Pedro  
Id 11  
Cpf 987654321  
Idade 23  
Nome Marcos  
Id 3  
Cpf 654321  
Idade 40  
  
Process finished with exit code 0
```

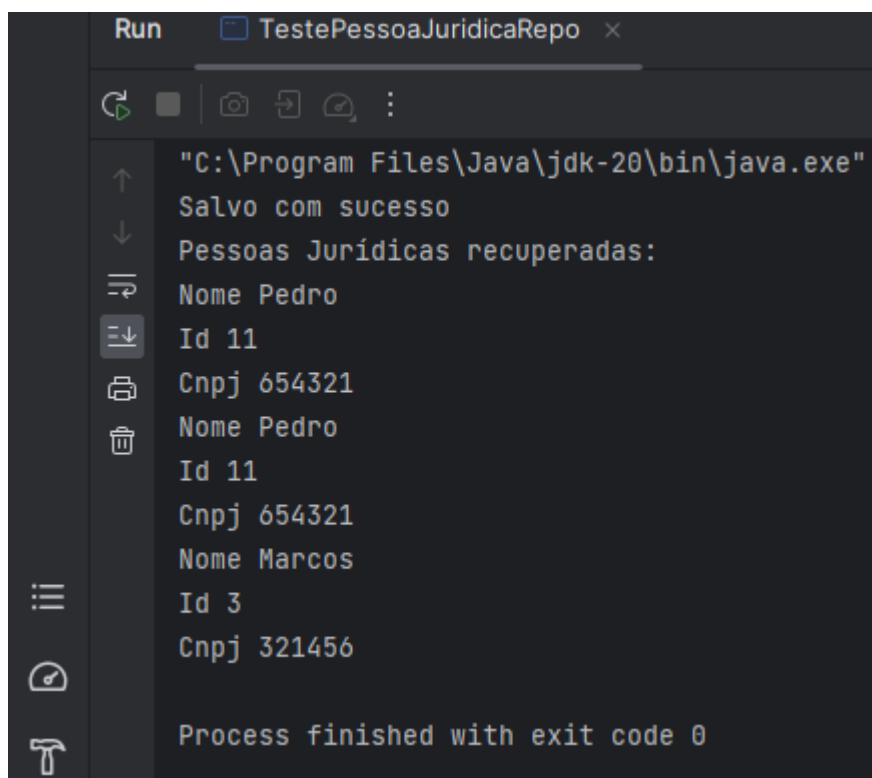
## Run - Teste Pessoa Jurídica:



The screenshot shows the Run console of an IDE. The title bar indicates the running application is 'TestePessoaJuridica'. The console output shows the execution of a Java program. The first line is the Java command path: `"C:\Program Files\Java\jdk-20\bin\java`. The subsequent lines are the program's output: `Nome Mestre Kame`, `Id 255`, and `Cnpj 21511248`. The final line indicates the process completed successfully: `Process finished with exit code 0`. The left sidebar of the IDE contains various icons for debugging and development tools.

```
"C:\Program Files\Java\jdk-20\bin\java
Nome Mestre Kame
Id 255
Cnpj 21511248
Process finished with exit code 0
```

## Run - Teste Pessoa Jurídica Repositório:



The screenshot shows the Run console of an IDE for the application 'TestePessoaJuridicaRepo'. The console output shows the execution of a Java program. The first line is the Java command path: `"C:\Program Files\Java\jdk-20\bin\java.exe"`. The subsequent lines are the program's output: `Salvo com sucesso`, `Pessoas Jurídicas recuperadas:`, `Nome Pedro`, `Id 11`, `Cnpj 654321`, `Nome Pedro`, `Id 11`, `Cnpj 654321`, `Nome Marcos`, `Id 3`, and `Cnpj 321456`. The final line indicates the process completed successfully: `Process finished with exit code 0`. The left sidebar of the IDE contains various icons for debugging and development tools.

```
"C:\Program Files\Java\jdk-20\bin\java.exe"
Salvo com sucesso
Pessoas Jurídicas recuperadas:
Nome Pedro
Id 11
Cnpj 654321
Nome Pedro
Id 11
Cnpj 654321
Nome Marcos
Id 3
Cnpj 321456
Process finished with exit code 0
```

## RESULTADOS DO CÓDIGO MAIN

Seleção:

```
© Main.java x
1 package Model;
2
3 import java.io.IOException;
4 import java.util.List;
5 import java.util.Scanner;
6
7 public class Main {
8     public static void main(String[] args) {
9         Scanner scanner = new Scanner(System.in);
10        PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
11        PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();
12
13        while (true) {
14            System.out.println("Escolha uma opção:\n");
15            System.out.println("1 - Incluir");
16            System.out.println("2 - Alterar");
17            System.out.println("3 - Excluir");
18            System.out.println("4 - Exibir pelo ID");
19            System.out.println("5 - Exibir todos");
20            System.out.println("6 - Salvar dados");
21            System.out.println("7 - Recuperar dados");
22            System.out.println("0 - Sair\n");
23
24            int opcao = scanner.nextInt();
```

## Opção 1

```
© Main.java x
26  switch (opcao) {
27      case 1:
28          System.out.println("Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica:");
29          int tipo = scanner.nextInt();
30          if (tipo == 1) {
31              System.out.println("Digite o ID: ");
32              int id = scanner.nextInt();
33              System.out.println("Digite o Nome: ");
34              String nome = scanner.next();
35              System.out.println("Digite o CPF: ");
36              String cpf = scanner.next();
37              System.out.println("Digite a Idade: ");
38              int idade = scanner.nextInt();
39
40              PessoaFisica pessoaFisica = new PessoaFisica(nome, id, cpf, idade);
41              pessoaFisicaRepo.inserir(pessoaFisica);
42              System.out.println("Pessoa Física cadastrada com sucesso!\n");
43
44          } else if (tipo == 2) {
45              System.out.println("Digite o ID:");
46              int id = scanner.nextInt();
47              System.out.println("Digite o nome da empresa:");
48              String nomeEmpresa = scanner.next();
49              System.out.println("Digite o CNPJ:");
50              String cnpj = scanner.next();
51
52              PessoaJuridica pessoaJuridica = new PessoaJuridica(nomeEmpresa, id, cnpj);
53              pessoaJuridicaRepo.inserir(pessoaJuridica);
54              System.out.println("Pessoa Jurídica cadastrada com sucesso!\n");
55          } else {
56              System.out.println("Tipo inválido.\n");
57          }
58          break;
```

## Opção 2

```
© Main.java x
case 2:
60 System.out.println("Escolha o tipo (1 - Pessoa Fisica, 2 - Pessoa Juridica):");
61 int alterar = scanner.nextInt();
62 if (alterar == 1) {
63     System.out.println("Digite o ID da Pessoa Fisica que deseja alterar:");
64     int idPessoaFisicaAlterar = scanner.nextInt();
65
66     PessoaFisica pessoaFisicaExistente = pessoaFisicaRepo.obter(idPessoaFisicaAlterar);
67
68     if (pessoaFisicaExistente != null) {
69         System.out.println("Dados atuais da Pessoa Fisica:");
70         pessoaFisicaExistente.exibir();
71         System.out.println("Digite o novo nome:");
72         String novoNomePF = scanner.next();
73         System.out.println("Digite o novo CPF:");
74         String novoCpf = scanner.next();
75         System.out.println("Digite a nova idade:");
76         int novaIdade = scanner.nextInt();
77
78         PessoaFisica pessoaFisicaAtualizada = new PessoaFisica(novoNomePF, idPessoaFisicaAlterar, novoCpf, novaIdade);
79         pessoaFisicaRepo.alterar(pessoaFisicaExistente, pessoaFisicaAtualizada);
80
81         System.out.println("Pessoa Fisica alterada com sucesso!\n");
82     } else {
83         System.out.println("Pessoa Fisica com o id"+ idPessoaFisicaAlterar + " não encontrada.");
84     }
85 }
```

```
© Main.java x
86 } else if (alterar == 2) {
87     System.out.println("Digite o ID da Pessoa Juridica que deseja alterar:");
88     int idPessoaJuridicaAlterar = scanner.nextInt();
89
90     PessoaJuridica pessoaJuridicaExistente = pessoaJuridicaRepo.obter(idPessoaJuridicaAlterar);
91
92     if (pessoaJuridicaExistente != null) {
93         System.out.println("Dados atuais da Pessoa Juridica: ");
94         pessoaJuridicaExistente.exibir();
95         System.out.println("Digite o novo nome da Empresa: ");
96         String novoNomePJ = scanner.next();
97         System.out.println("Digite o novo CNPJ: ");
98         String novoCNPJ = scanner.next();
99
100         PessoaJuridica pessoaJuridicaAtualizada = new PessoaJuridica(novoNomePJ, idPessoaJuridicaAlterar, novoCNPJ);
101         pessoaJuridicaRepo.alterar(pessoaJuridicaExistente, pessoaJuridicaAtualizada);
102
103         System.out.println("Pessoa Juridica alterada com sucesso!\n");
104
105     } else {
106         System.out.println("Pessoa Fisica com o id"+ idPessoaJuridicaAlterar + " não encontrada.\n");
107     }
108
109 } else {
110     System.out.println("Tipo inválido.\n");
111 }
112 break;
```

## Opção 3

```
© Main.java x
113 case 3:
114     System.out.println("Escolha (1 - Pessoa Física, 2 - Pessoa Jurídica):");
115     int tipoExcluir = scanner.nextInt();
116     if (tipoExcluir == 1) {
117         System.out.println("Digite o ID da Pessoa Física que deseja excluir:");
118         int idPessoaFisicaExcluir = scanner.nextInt();
119
120         PessoaFisica pessoaFisicaExistente = pessoaFisicaRepo.obter(idPessoaFisicaExcluir);
121
122         if (pessoaFisicaExistente != null) {
123             System.out.println("Dados da Pessoa Física a ser excluída:");
124             pessoaFisicaExistente.exibir();
125
126             System.out.println("Tem certeza de que deseja excluir esta Pessoa Física? (S/N)");
127             String confirmacaoPessoaFisica = scanner.next();
128
129             if (confirmacaoPessoaFisica.equalsIgnoreCase("S")) {
130                 pessoaFisicaRepo.excluir(idPessoaFisicaExcluir);
131                 System.out.println("Pessoa Física excluída com sucesso!\n");
132             } else {
133                 System.out.println("Exclusão cancelada.");
134             }
135             } else {
136                 System.out.println("Pessoa Física com o ID especificado não encontrada.\n");
137             }
138         }
139     }
```

```
© Main.java x
139
140
141     else if (tipoExcluir == 2) {
142         System.out.println("Digite o ID da Pessoa Jurídica que deseja excluir:");
143         int idPessoaJuridicaExcluir = scanner.nextInt();
144
145         PessoaJuridica pessoaJuridicaExistente = pessoaJuridicaRepo.obter(idPessoaJuridicaExcluir);
146
147         if (pessoaJuridicaExistente != null) {
148             System.out.println("Dados da Pessoa Jurídica a ser excluída:");
149             pessoaJuridicaExistente.exibir();
150
151             System.out.println("Tem certeza de que deseja excluir esta Pessoa Jurídica? (S/N)");
152             String confirmacaoPessoaJuridica = scanner.next();
153
154             if (confirmacaoPessoaJuridica.equalsIgnoreCase("S")) {
155                 pessoaJuridicaRepo.excluir(idPessoaJuridicaExcluir);
156                 System.out.println("Pessoa Jurídica excluída com sucesso!\n");
157             } else {
158                 System.out.println("Exclusão cancelada.\n");
159             }
160             } else {
161                 System.out.println("Pessoa Jurídica com o ID especificado não encontrada.\n");
162             }
163         } else {
164             System.out.println("Tipo inválido.\n");
165         }
166     }
167     break;
```

## Opção 4

```
© Main.java x
165 case 4:
166     System.out.println("Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):");
167     int tipoObter = scanner.nextInt();
168     if (tipoObter == 1) {
169         System.out.println("Digite o ID da Pessoa Física que deseja obter:");
170         int idObterPessoaFisica = scanner.nextInt();
171
172         PessoaFisica pessoaFisicaExistente = pessoaFisicaRepo.obter(idObterPessoaFisica);
173
174         if (pessoaFisicaExistente != null) {
175             System.out.println("Dados da Pessoa Física obtida:");
176             pessoaFisicaExistente.exibir();
177         } else {
178             System.out.println("Pessoa Física com o ID especificado não encontrada.");
179         }
180     } else if (tipoObter == 2) {
181         System.out.println("Digite o ID da Pessoa Jurídica que deseja obter:");
182         int idObterPessoaJuridica = scanner.nextInt();
183
184         PessoaJuridica pessoaJuridicaExiste = pessoaJuridicaRepo.obter(idObterPessoaJuridica);
185
186         if (pessoaJuridicaExiste != null) {
187             System.out.println("Dados da Pessoa Jurídica obtida:");
188             pessoaJuridicaExiste.exibir();
189         }
190     } else {
191         System.out.println("Tipo inválido.\n");
192     }
```

## Opção 5

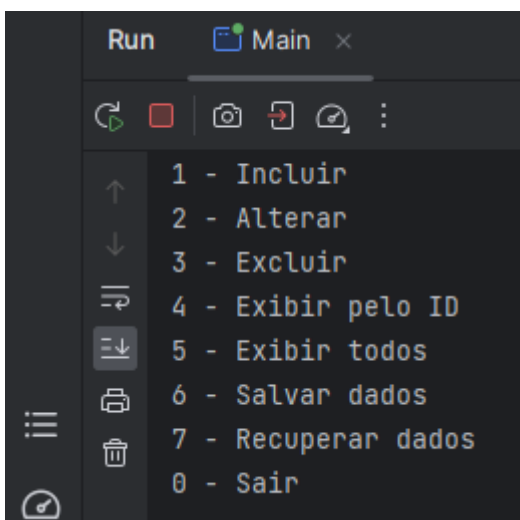
```
© Main.java ×
194      case 5:
195          System.out.println("Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):");
196          int tipoExibirTodos = scanner.nextInt();
197          if (tipoExibirTodos == 1) {
198              List<PessoaFisica> pessoasFisicas = pessoaFisicaRepo.obterTodos();
199              if (!pessoasFisicas.isEmpty()) {
200                  System.out.println("Lista de todas as Pessoas Físicas:");
201                  for (PessoaFisica pessoaFisica : pessoasFisicas) {
202                      pessoaFisica.exibir();
203                      System.out.println();
204                  }
205              } else {
206                  System.out.println("Nenhuma Pessoa Física encontrada.\n");
207              }
208          } else if (tipoExibirTodos == 2) {
209              List<PessoaJuridica> pessoaJuridicas = pessoaJuridicaRepo.obterTodos();
210              if (!pessoaJuridicas.isEmpty()) {
211                  System.out.println("Lista de todas as Pessoas Jurídicas:");
212                  for (PessoaJuridica pessoaJuridica : pessoaJuridicas) {
213                      pessoaJuridica.exibir();
214                      System.out.println();
215                  }
216              } else {
217                  System.out.println("Nenhuma Pessoa Jurídica encontrada.\n");
218              }
219          } else {
220              System.out.println("Tipo inválido.\n");
221          }
222          break;
```



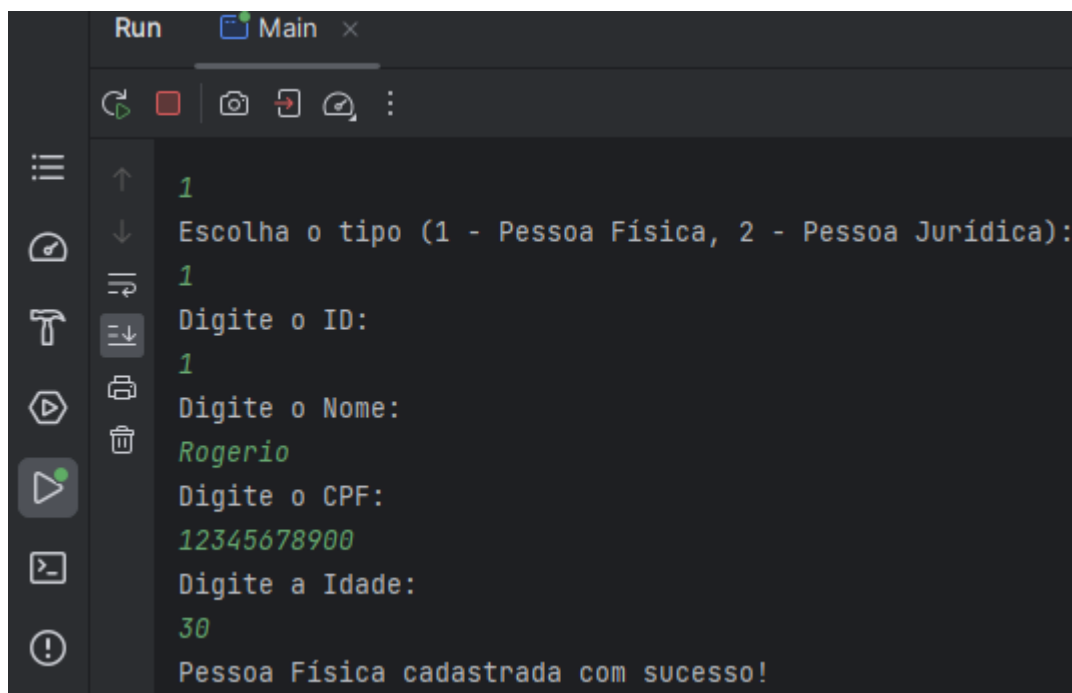
## Opção 6 e 0

```
© Main.java x
223     case 6:
224         System.out.println("Digite o prefixo dos arquivos: ");
225         String prefixoSalvar = scanner.next();
226         try {
227             pessoaFisicaRepo.persistir( nomeArquivo: prefixoSalvar + ".fisica.bin");
228             pessoaJuridicaRepo.persistir( nomeArquivo: prefixoSalvar + ".juridica.bin");
229             System.out.println("Dados salvos com sucesso.\n");
230         } catch (IOException e) {
231             System.out.println("Erro ao salvar dados:" + e.getMessage());
232         }
233         break;
234     case 7:
235         System.out.println("Digite o prefixo dos arquivos: ");
236         String prefixoRecuperar = scanner.next();
237         try {
238             pessoaFisicaRepo.recuperar( nomeArquivo: prefixoRecuperar + ".fisica.bin");
239             pessoaJuridicaRepo.recuperar( nomeArquivo: prefixoRecuperar + ".juridica.bin");
240             System.out.println("Dados recuperados com sucesso.\n");
241         } catch (IOException | ClassNotFoundException e) {
242             System.out.println("Erro ao recuperar dados: " + e.getMessage());
243         }
244         break;
245     case 0:
246         scanner.close();
247         System.exit( status: 0);
248         break;
249     default:
250         System.out.println("Opção inválida. Tente novamente.\n");
251         break;
252     }
253 }
254 }
255 }
```

## TESTES DO CÓDIGO MAIN

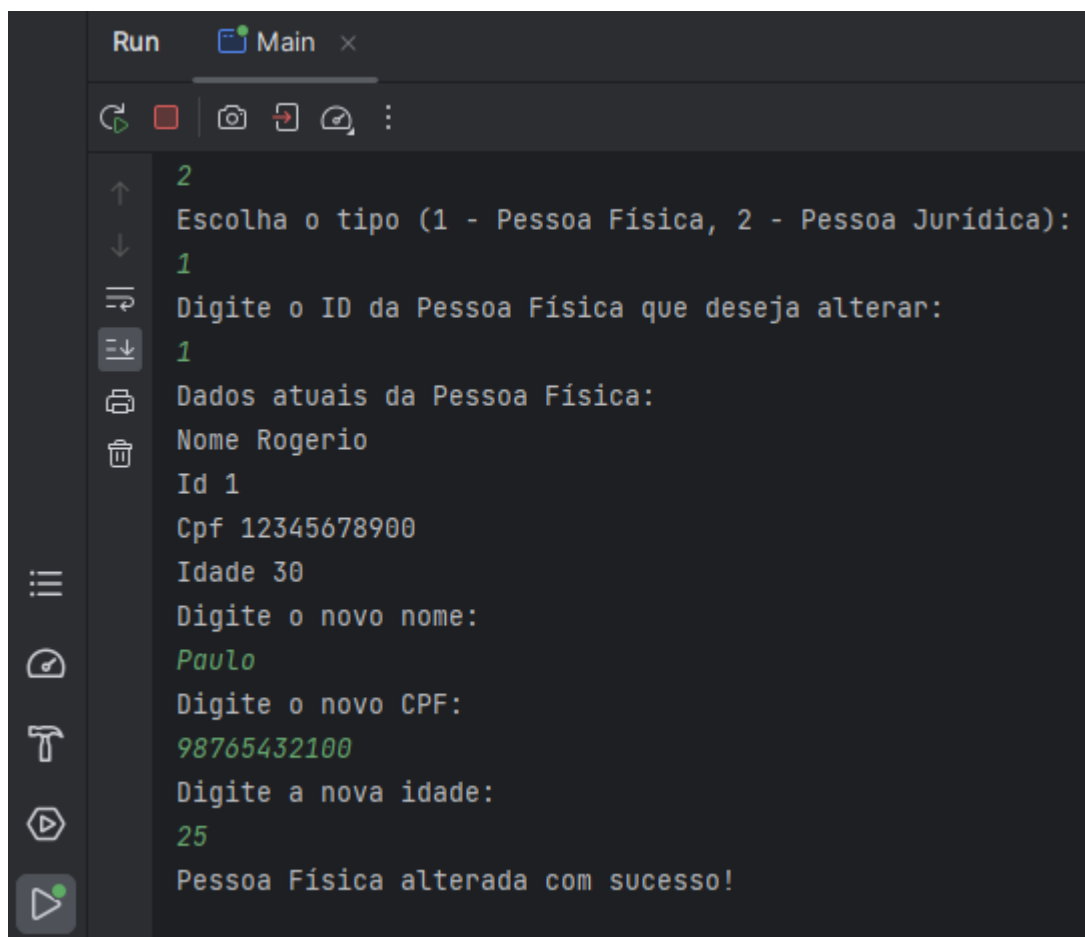


## Incluir Pessoa Física:



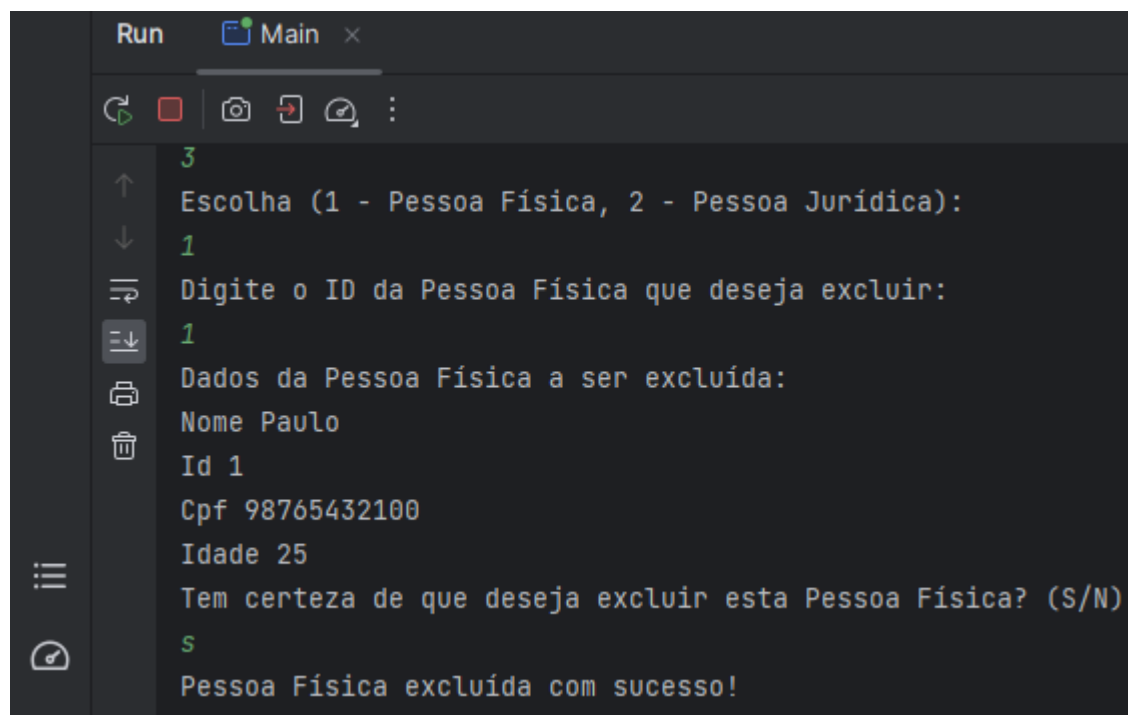
```
Run Main x
1
Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Digite o ID:
1
Digite o Nome:
Rogerio
Digite o CPF:
12345678900
Digite a Idade:
30
Pessoa Física cadastrada com sucesso!
```

## Alterar Pessoa Física:



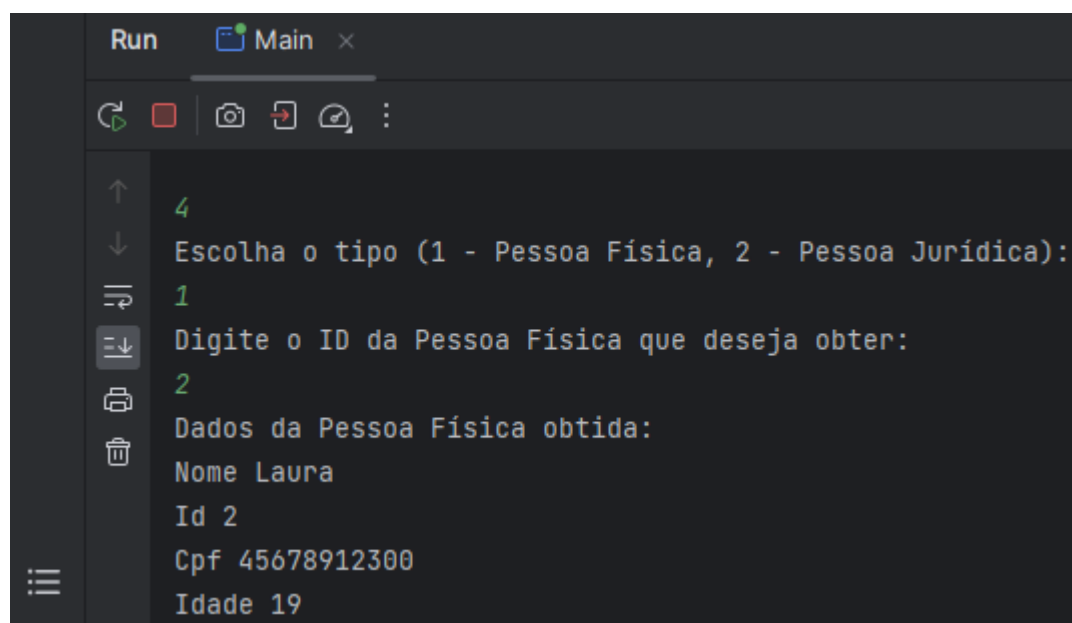
```
Run Main x
2
Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Digite o ID da Pessoa Física que deseja alterar:
1
Dados atuais da Pessoa Física:
Nome Rogerio
Id 1
Cpf 12345678900
Idade 30
Digite o novo nome:
Paulo
Digite o novo CPF:
98765432100
Digite a nova idade:
25
Pessoa Física alterada com sucesso!
```

## Excluir Pessoa Física:



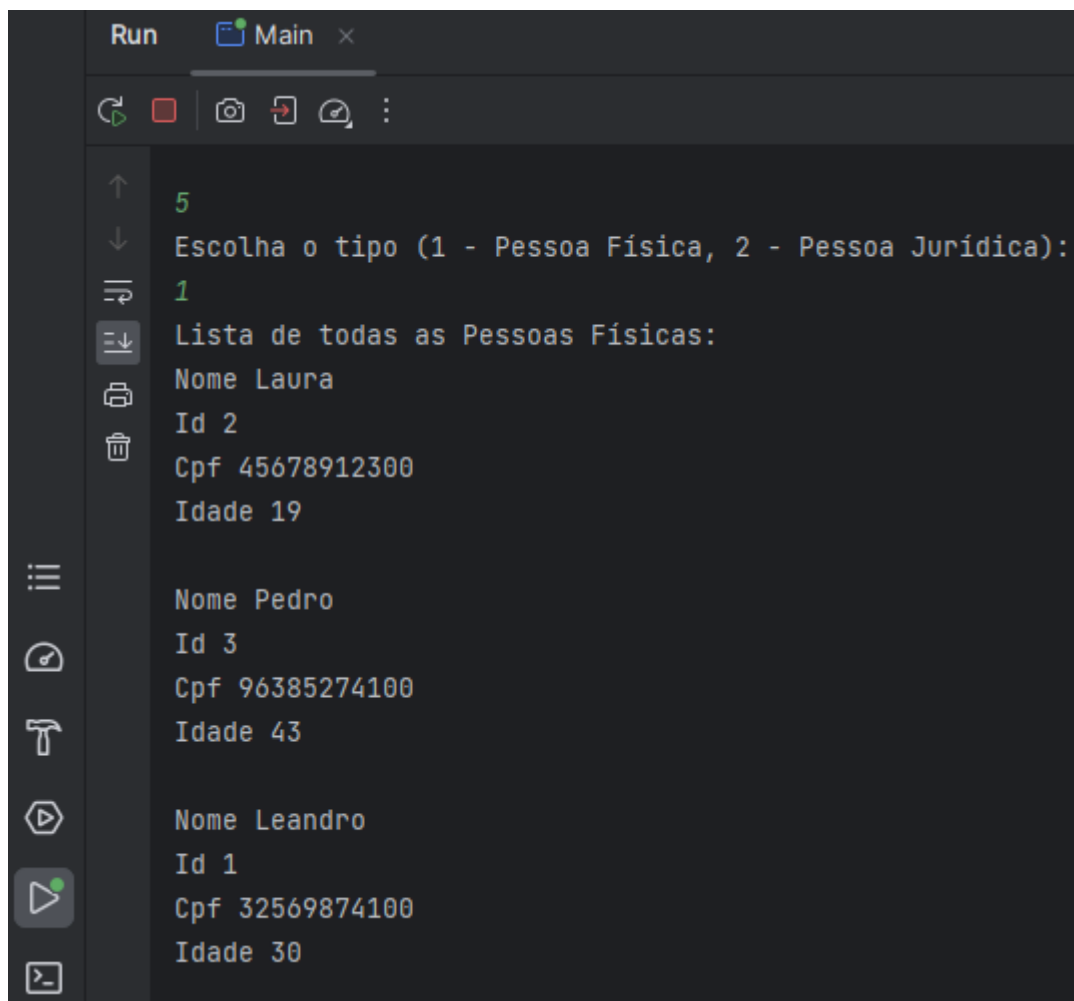
```
Run Main x
3
Escolha (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Digite o ID da Pessoa Física que deseja excluir:
1
Dados da Pessoa Física a ser excluída:
Nome Paulo
Id 1
Cpf 98765432100
Idade 25
Tem certeza de que deseja excluir esta Pessoa Física? (S/N)
S
Pessoa Física excluída com sucesso!
```

## Exibir pelo ID Pessoa Física:



```
Run Main x
4
Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Digite o ID da Pessoa Física que deseja obter:
2
Dados da Pessoa Física obtida:
Nome Laura
Id 2
Cpf 45678912300
Idade 19
```

## Exibir todos Pessoa Física:



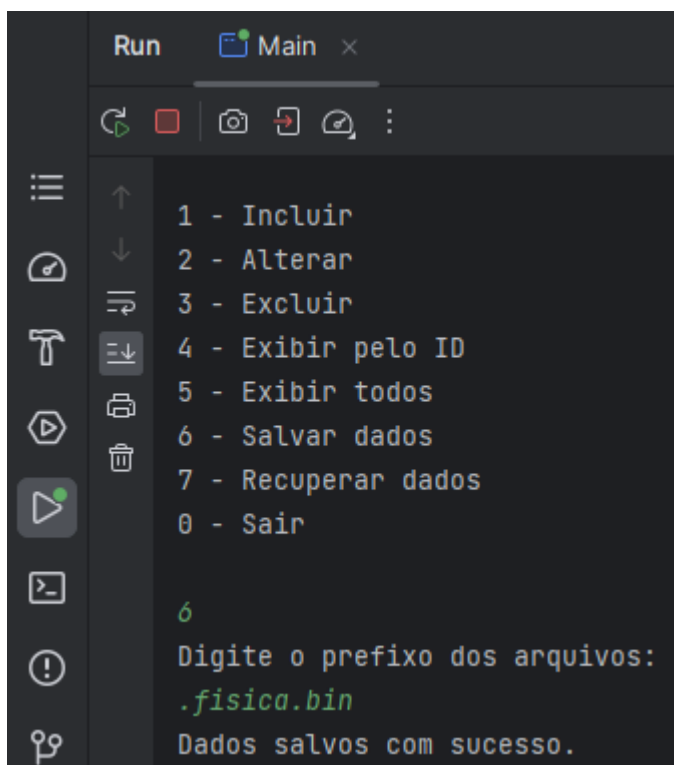
The screenshot shows a terminal window with a dark background. At the top, there's a tab labeled 'Run' and 'Main'. Below the tab, there's a toolbar with icons for running, stopping, and other actions. The main area of the terminal displays the following text:

```
5
Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Lista de todas as Pessoas Físicas:
Nome Laura
Id 2
Cpf 45678912300
Idade 19

Nome Pedro
Id 3
Cpf 96385274100
Idade 43

Nome Leandro
Id 1
Cpf 32569874100
Idade 30
```

## Salvar dados Pessoa Física:



The screenshot shows a terminal window with a dark background. At the top, there's a tab labeled 'Run' and 'Main'. Below the tab, there's a toolbar with icons for running, stopping, and other actions. The main area of the terminal displays the following text:

```
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair

6
Digite o prefixo dos arquivos:
.fisica.bin
Dados salvos com sucesso.
```

## Recuperar dados Pessoa Física:

```
Run Main x
7
Digite o prefixo dos arquivos:
.fisica.bin
Dados recuperados com sucesso.

Escolha uma opção:

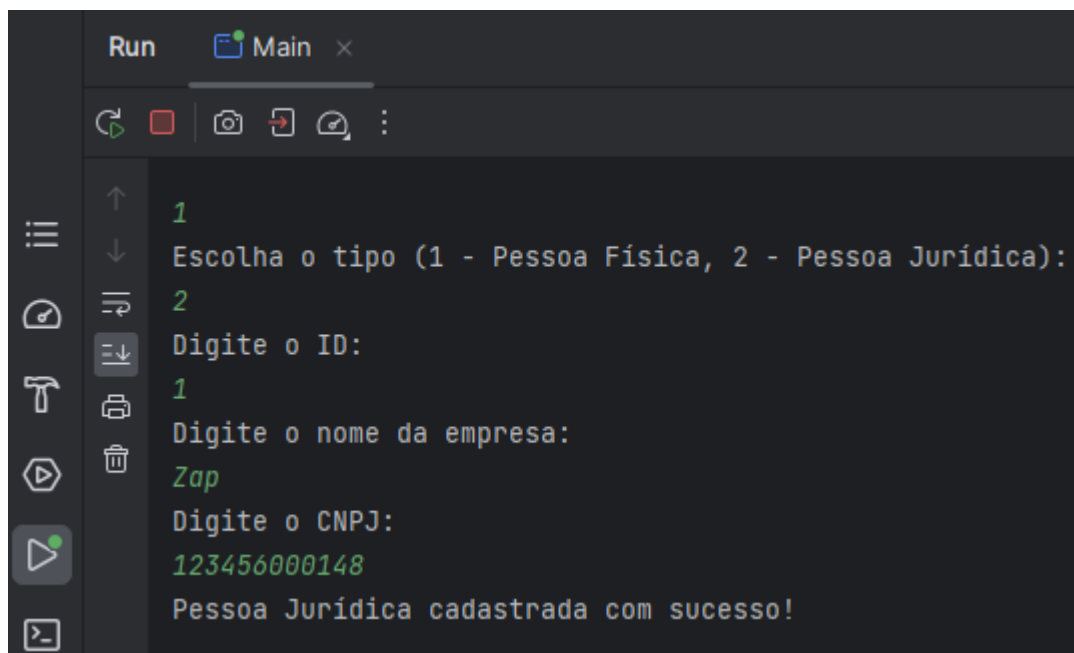
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair

5
Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
1
Lista de todas as Pessoas Físicas:
Nome Laura
Id 2
Cpf 45678912300
Idade 19

Nome Pedro
Id 3
Cpf 96385274100
Idade 43

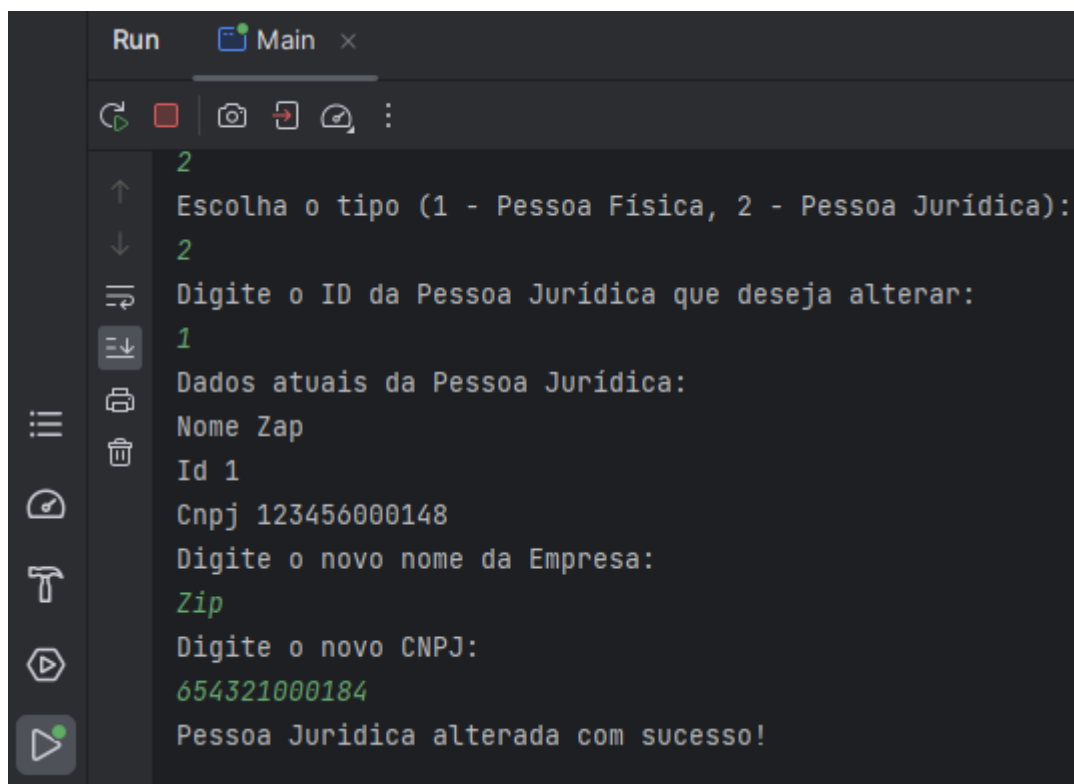
Nome Leandro
Id 1
Cpf 32569874100
Idade 30
```

## Incluir Pessoa Jurídica:



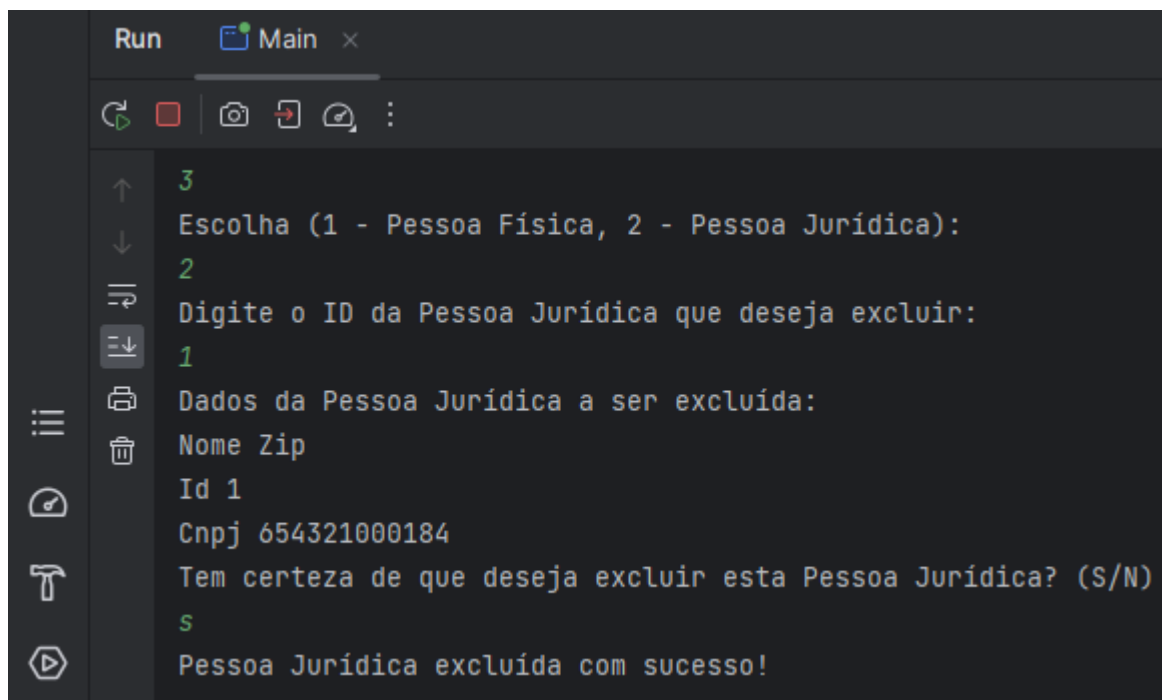
```
Run Main x
1
Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
2
Digite o ID:
1
Digite o nome da empresa:
Zap
Digite o CNPJ:
123456000148
Pessoa Jurídica cadastrada com sucesso!
```

## Alterar Pessoa Jurídica:



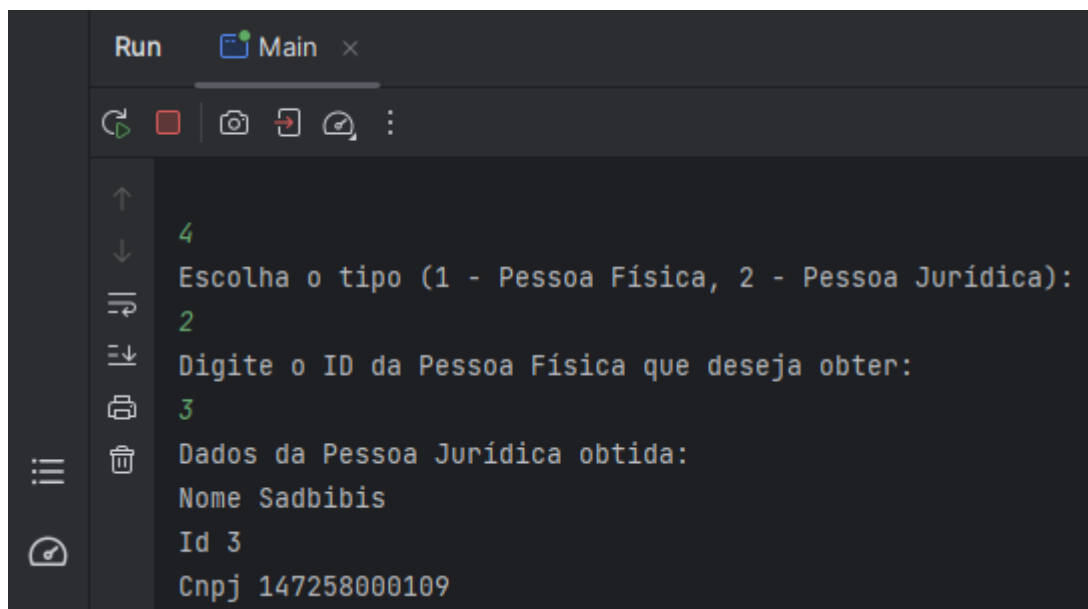
```
Run Main x
2
Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
2
Digite o ID da Pessoa Jurídica que deseja alterar:
1
Dados atuais da Pessoa Jurídica:
Nome Zap
Id 1
Cnpj 123456000148
Digite o novo nome da Empresa:
Zip
Digite o novo CNPJ:
654321000184
Pessoa Juridica alterada com sucesso!
```

## Excluir Pessoa Jurídica:



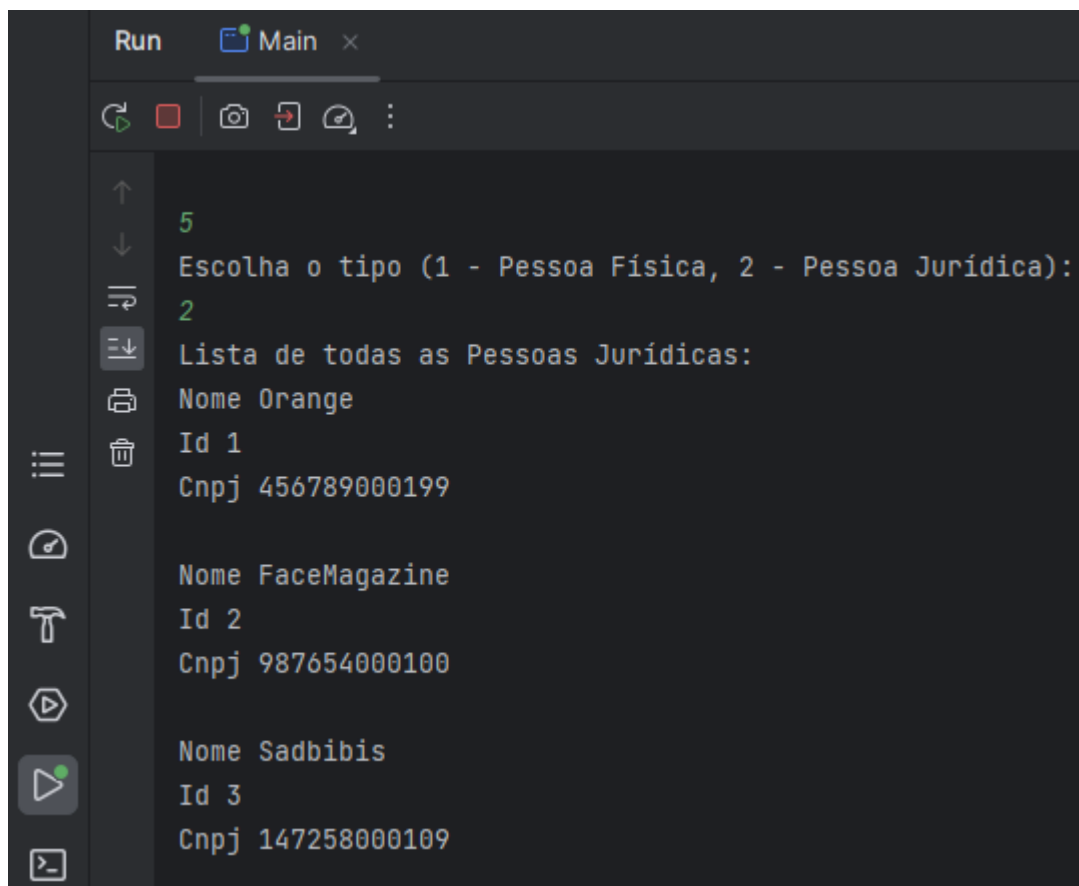
```
Run Main x
3
Escolha (1 - Pessoa Física, 2 - Pessoa Jurídica):
2
Digite o ID da Pessoa Jurídica que deseja excluir:
1
Dados da Pessoa Jurídica a ser excluída:
Nome Zip
Id 1
Cnpj 654321000184
Tem certeza de que deseja excluir esta Pessoa Jurídica? (S/N)
s
Pessoa Jurídica excluída com sucesso!
```

## Exibir pelo ID Pessoa Jurídica:



```
Run Main x
4
Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
2
Digite o ID da Pessoa Física que deseja obter:
3
Dados da Pessoa Jurídica obtida:
Nome Sadbibis
Id 3
Cnpj 147258000109
```

## Exibir todos Pessoa Jurídica:



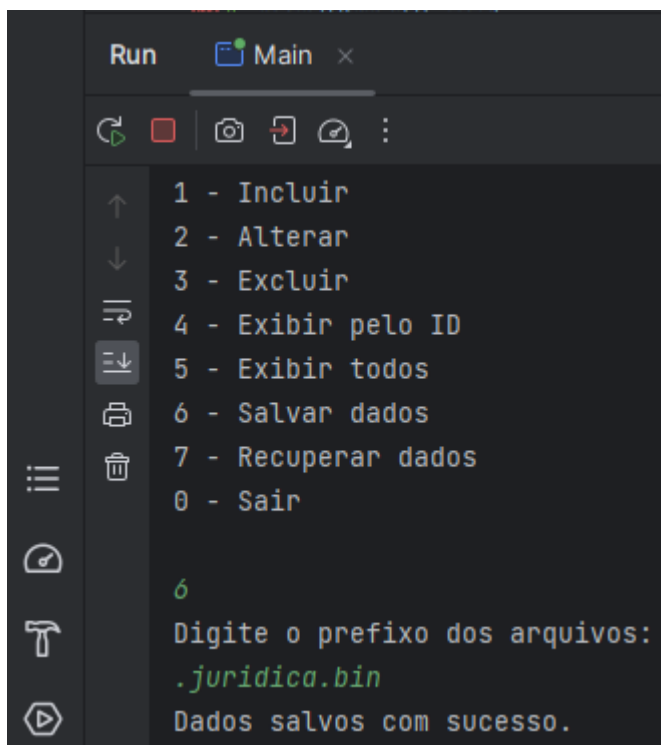
The screenshot shows a terminal window with a dark background. The title bar at the top says "Run" and "Main x". Below the title bar is a toolbar with icons for running, stopping, and other actions. The main area of the terminal displays the following text:

```
5
Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
2
Lista de todas as Pessoas Jurídicas:
Nome Orange
Id 1
Cnpj 456789000199

Nome FaceMagazine
Id 2
Cnpj 987654000100

Nome Sadbibis
Id 3
Cnpj 147258000109
```

## Salvar dados Pessoa Jurídica:



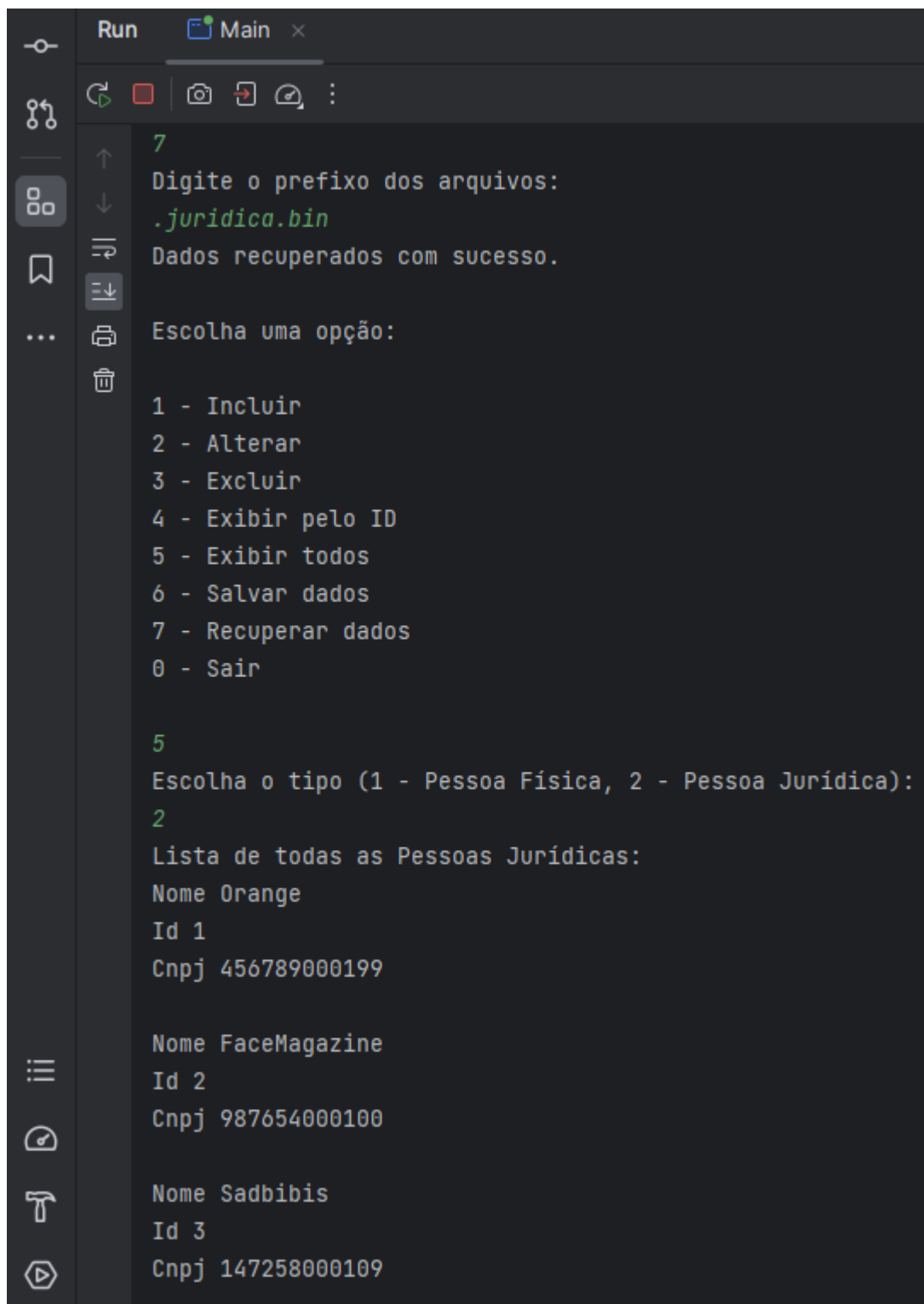
The screenshot shows a terminal window with a dark background. The title bar at the top says "Run" and "Main x". Below the title bar is a toolbar with icons for running, stopping, and other actions. The main area of the terminal displays the following text:

```
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair

6
Digite o prefixo dos arquivos:
.juridica.bin
Dados salvos com sucesso.
```



## Recuperar dados Pessoa Jurídica:



```
Run Main x
7
Digite o prefixo dos arquivos:
.juridica.bin
Dados recuperados com sucesso.

Escolha uma opção:

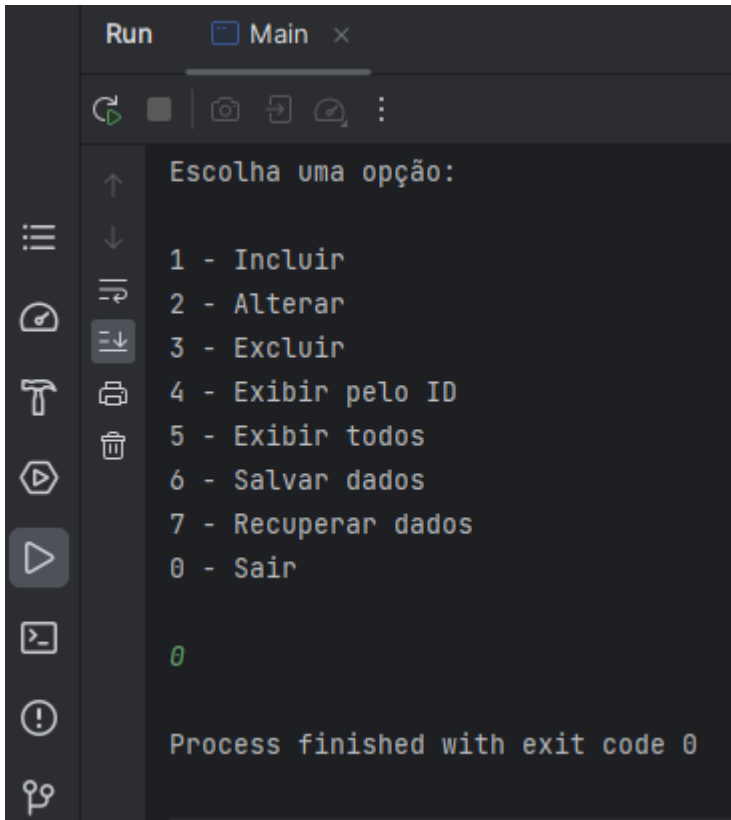
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair

5
Escolha o tipo (1 - Pessoa Física, 2 - Pessoa Jurídica):
2
Lista de todas as Pessoas Jurídicas:
Nome Orange
Id 1
Cnpj 456789000199

Nome FaceMagazine
Id 2
Cnpj 987654000100

Nome Sadbibis
Id 3
Cnpj 147258000109
```

Sair:



```
Run Main x
Escolha uma opção:
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir todos
6 - Salvar dados
7 - Recuperar dados
0 - Sair

0

Process finished with exit code 0
```

## Quais as vantagens e desvantagens do uso de herança?

**Reutilização de Código:** A herança permite que as classes filhas herdem os atributos e métodos da classe pai, o que promove a reutilização de código.

**Polimorfismo:** As classes derivadas podem ser tratadas como instâncias da classe base, permitindo a criação de código mais flexível e genérico.

**Organização Hierárquica:** A herança pode ser usada para criar uma estrutura hierárquica que reflete as relações do mundo real, facilitando o design orientado a objetos.

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

**Acoplamento:** Herança pode levar a um alto acoplamento entre classes pai e filhas, tornando as mudanças em uma classe pai potencialmente impactantes em todas as classes filhas.

**Herança Múltipla Complexa:** Em linguagens que suportam herança múltipla, pode surgir complexidade quando uma classe herda de várias classes, levando a ambiguidades e dificuldades de manutenção.

**Fragilidade da Hierarquia:** Alterações na hierarquia de herança podem afetar inesperadamente o comportamento das classes derivadas, resultando em problemas de manutenção.

**Interface Serializable na Persistência em Arquivos Binários:** A interface Serializable em Java é necessária ao efetuar persistência em arquivos binários porque ela indica ao mecanismo de serialização que uma classe pode ser convertida em uma sequência de bytes, que pode ser gravada em um arquivo e posteriormente reconstruída. Sem a implementação da interface Serializable, uma classe não pode ser serializada e, portanto, não pode ser armazenada em arquivos binários.

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

### **Paradigma Funcional na API Stream do Java:**

A API Stream do Java utiliza o paradigma funcional para executar operações em coleções de dados de forma concisa e expressiva. Ela introduz conceitos como funções de alta ordem, lambda expressions e operações como map, filter, reduce e forEach, que permitem a manipulação de coleções de maneira funcional e declarativa. Isso torna o código mais legível e facilita a execução de operações complexas em coleções de dados.

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

### **Paradigma Funcional na API Stream do Java:**

A API Stream do Java utiliza o paradigma funcional para executar operações em coleções de dados de forma concisa e expressiva. Ela introduz conceitos como funções de alta ordem, lambda expressions e operações como map, filter,

reduce e forEach, que permitem a manipulação de coleções de maneira funcional e declarativa. Isso torna o código mais legível e facilita a execução de operações complexas em coleções de dados.

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

Quando se trabalha com Java, um padrão comum de desenvolvimento na persistência de dados em arquivos é a utilização de classes de entrada e saída (I/O) do Java, como FileInputStream, FileOutputStream, ObjectInputStream e ObjectOutputStream. Essas classes fornecem métodos para ler e escrever dados em arquivos de forma eficiente. Para a persistência em arquivos binários, a implementação da interface Serializable é frequentemente usada para permitir a serialização e desserialização de objetos em arquivos binários. Além disso, práticas como tratamento de exceções e fechamento adequado de recursos são adotadas para garantir a integridade dos dados e a segurança durante a operação de leitura e escrita em arquivos.

## **O que são elementos estáticos e qual o motivo para o método main adotar esse modificador?**

### **Elementos Estáticos:**

Elementos estáticos em Java, como variáveis e métodos, pertencem à classe em vez de uma instância específica dessa classe. São acessíveis diretamente a partir da classe, sem a necessidade de criar uma instância da classe. O modificador static é usado para definir elementos estáticos. Os elementos estáticos são compartilhados entre todas as instâncias da classe e podem ser acessados usando o nome da classe em vez do objeto. Isso é útil para criar variáveis ou métodos que são comuns a todas as instâncias da classe.

### **Método Main com Modificador Estático:**

O método main em Java é o ponto de entrada para a execução de um programa. Ele é declarado como estático (public static void main(String[] args)) para que possa ser chamado sem a necessidade de criar uma instância da classe que contém o método main. Isso permite que o sistema operacional ou a máquina virtual Java inicie o programa sem a necessidade de criar um objeto. O método

main precisa ser estático porque é chamado antes que qualquer instância da classe seja criada.

## Para que serve a classe Scanner?

A classe Scanner em Java é usada para ler dados do usuário ou de outras fontes de entrada, como arquivos. Ela fornece métodos para ler dados de vários tipos, como inteiros, números de ponto flutuante, strings, entre outros. A classe Scanner é amplamente utilizada para obter entrada do teclado durante a interação com o usuário em programas Java, tornando a leitura de dados mais simples e flexível.

## Como o uso de classes de repositório impactou na organização do código?

A introdução das classes de repositório (por exemplo, PessoaFisicaRepo e PessoaJuridicaRepo) teve um impacto significativo na organização do código. Elas ajudaram a:

**Separar Responsabilidades:** As classes de repositório ficam encarregadas de gerenciar o armazenamento e recuperação de objetos, separando essa responsabilidade do restante do código.

**Melhorar a Manutenção:** As operações de persistência (salvar e recuperar) são encapsuladas nas classes de repositório, tornando as mudanças futuras nessas operações mais fáceis de serem realizadas sem afetar o restante do código.

**Promover Reutilização:** As classes de repositório podem ser reutilizadas em diferentes partes do código, permitindo que várias partes do programa acessem os mesmos dados de forma consistente.

**Facilitar a Escalabilidade:** À medida que o sistema cresce, a organização em classes de repositório ajuda a manter a estrutura do código organizada e escalável.

Em resumo, o uso de classes de repositório melhora a modularidade e a manutenção do código, além de promover boas práticas de programação ao separar as preocupações relacionadas ao armazenamento e recuperação de dados.

**Repositório GIT:** <https://github.com/LucasHSS904/CadastroPOO>

**Data de Elaboração:** 06/09/2023