



UNIVERSIDADE  
**Estácio de Sá**

<b>Universidade</b>	Estácio de Sá
<b>Campus</b>	Polo de Cobilândia / Vila – Velha/ES
<b>Nome do Curso</b>	Desenvolvimento Full Stack
<b>Nome da Disciplina</b>	RPG0014 - Iniciando o Caminho Pelo Java
<b>Turma</b>	9001
<b>Semestre</b>	Primeiro Semestre de 2024
<b>Integrantes do Grupo</b>	Tiago de Jesus Pereira Furtado

**VILA VELHA  
2024**

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

## 1- 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.
- persistência em arquivos binários.

## 3 - Todos os códigos solicitados neste roteiro de aula:

### Códigos da Classe Pessoa

```
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
6  package model;
7  import java.io.Serializable;
8
9  /**
10   *
11   * @author TiagodeJesus
12   */
13
14  public class Pessoa implements Serializable{
15
16      private static final long serialVersionUID = 1L;
17      int id;
18      String nome;
19
20      public Pessoa(int id, String nome) {
21          this.id = id;
22          this.nome = nome;
23      }
24
25      public void exibir() {
26          System.out.println("ID:" + id);
27          System.out.println("Nome:" + nome);
28      }
29
30      public int getId() {
31          return id;
32      }
33
34      public void setId(int id) {
35          this.id = id;
36      }
37
38      public String getNome() {
39          return nome;
40      }
41
42      public void setNome(String nome) {
43          this.nome = nome;
44      }
45
46  }
```

“Descrição do Código”

Classe Pessoa, com os campos id (inteiro) e nome (texto), método exibir, para impressão dos dados, construtor padrão e completo, além de getters e setters para todos os campos.

## Códigos da Classe Pessoa Física

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5
6  package model;
7  import java.io.Serializable;
8
9  /**
10   *
11   * @author TiagodeJesus
12   */
13
14  public class PessoaFisica extends Pessoa implements Serializable{
15
16      private static final long serialVersionUID = 1L;
17      String cpf;
18      int idade;
19
20      public PessoaFisica(int id, String nome, String cpf, int idade) {
21          super(id, nome);
22          this.cpf= cpf;
23          this.idade= idade;
24      }
25
26      @Override
27      public void exibir() {
28          super.exibir();
29          System.out.println("CPF: " + cpf);
30          System.out.println("Idade: " + idade + "\n");
31      }
32
33      public String getCpf() {
34          return cpf;
35      }
36
37      public void setCpf(String cpf) {
38          this.cpf = cpf;
39      }
40
41      public int getIdade() {
42          return idade;
43      }
44
45      public void setIdade(int idade) {
46          this.idade = idade;
47      }
48  }
49
```

### “Descrição do Código”

Classe PessoaFisica, herdando de Pessoa, com o acréscimo dos campos CPF (texto) e idade (inteiro), método exibir polimórfico, construtores, getters e setters.

## Códigos da Classe Pessoa Jurídica

```
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
6  package model;
7  import java.io.Serializable;
8
9  /**
10   *
11   * @author TiagodeJesus
12   */
13
14  public class PessoaJuridica extends Pessoa implements Serializable{
15
16      private static final long serialVersionUID = 1L;
17      String cnpj;
18
19      public PessoaJuridica(int id, String nome, String cnpj) {
20          super(id, nome);
21          this.cnpj=cnpj;
22      }
23
24      @Override
25      public void exibir() {
26          super.exibir();
27          System.out.println("CNPJ: " + cnpj+"\n");
28      }
29
30      public String getcnpj() {
31          return cnpj;
32      }
33
34      public void setcnpj(String cnpj) {
35          this.cnpj = cnpj;
36      }
37  }
```

### “Descrição do Código”

Classe Pessoa Jurídica, herdando de Pessoa, com o acréscimo do campo CNPJ (texto), método exibir polimórfico, construtores, getters e setters.

No pacote model criar os gerenciadores, com as seguintes características:

### Códigos da Classe PessoaFisicaRepo

```
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
6  package model;
7  import java.io.*;
8  import java.util.ArrayList;
9  import java.util.List;
10
11  /**
12   *
13   * @author TiagodeJesus
14   */
15
16  public class PessoaFisicaRepo implements Serializable {
17
18      private static final long serialVersionUID = 1L;
19      private final List<PessoaFisica> pessoasFisicas = new ArrayList<>();
20
21      public void inserir(PessoaFisica pessoa) {
22          pessoasFisicas.add(pessoa);
23      }
24
25      public void alterar(PessoaFisica pessoa) {
26          int index = obterIndexPorId(pessoa.getId());
27          if (index != -1) {
28              pessoasFisicas.set(index, pessoa);
29          }
30      }
31
32      public void excluir(int id) {
33          PessoaFisica pessoa = obter(id);
34          if (pessoa != null) {
35              pessoasFisicas.remove(pessoa);
36          }
37      }
38
39      public PessoaFisica obter(int id) {
40          for (PessoaFisica pessoa : pessoasFisicas) {
41              if (pessoa.getId() == id) {
42                  return pessoa;
43              }
44          }
45          return null;
46      }
47
48      public List<PessoaFisica> obterTodos() {
49          return new ArrayList<>(pessoasFisicas);
50      }
51
52      public void persistir(String nomeArquivo) throws IOException {
53          try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
54              outputStream.writeObject(this);
55          }
56      }
57
58      public static PessoaFisicaRepo recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
59          try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
60              return (PessoaFisicaRepo) inputStream.readObject();
61          }
62      }
63
64      private int obterIndexPorId(int id) {
65          for (int i = 0; i < pessoasFisicas.size(); i++) {
66              if (pessoasFisicas.get(i).getId() == id) {
67                  return i;
68              }
69          }
70          return -1;
71      }
72  }
73
```

#### “Descrição do Código”

Classe PessoaFisicaRepo, contendo um ArrayList de PessoaFisica, nível de acesso privado, e métodos públicos inserir, alterar, excluir, obter e obterTodos, para gerenciamento das entidades contidas no ArrayList.

## Códigos da Classe PessoaJuridicaRepo

```
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
6  package model;
7  import java.io.*;
8  import java.util.ArrayList;
9  import java.util.List;
10
11  /**
12   *
13   * @author TiagodeJesus
14   */
15
16  public class PessoaJuridicaRepo implements Serializable {
17
18      private static final long serialVersionUID = 1L;
19      private List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
20
21      public void inserir(PessoaJuridica pessoa) {
22          pessoasJuridicas.add(pessoa);
23      }
24
25      public void alterar(PessoaJuridica pessoa) {
26          int index = obterIndexPorId(pessoa.getId());
27          if (index != -1) {
28              pessoasJuridicas.set(index, pessoa);
29          }
30      }
31
32      public void excluir(int id) {
33          PessoaJuridica pessoa = obter(id);
34          if (pessoa != null) {
35              pessoasJuridicas.remove(pessoa);
36          }
37      }
38
39      public PessoaJuridica obter(int id) {
40          for (PessoaJuridica pessoa : pessoasJuridicas) {
41              if (pessoa.getId() == id) {
42                  return pessoa;
43              }
44          }
45          return null;
46      }
47
48      public List<PessoaJuridica> obterTodos() {
49          return new ArrayList<>(pessoasJuridicas);
50      }
51
52      public void persistir(String nomeArquivo) throws IOException {
53          try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
54              outputStream.writeObject(this);
55          }
56      }
57
58      public static PessoaJuridicaRepo recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
59          try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
60              return (PessoaJuridicaRepo) inputStream.readObject();
61          }
62      }
63
64      private int obterIndexPorId(int id) {
65          for (int i = 0; i < pessoasJuridicas.size(); i++) {
66              if (pessoasJuridicas.get(i).getId() == id) {
67                  return i;
68              }
69          }
70          return -1;
71      }
72  }
```

### “Descrição do Código”

Classe PessoaJuridicaRepo, com um ArrayList de PessoaJuridica, nível de acesso privado, e métodos públicos inserir, alterar, excluir, obter e obterTodos, para gerenciamento das entidades contidas no ArrayList.

## **Alterar o método Main da classe principal para testar os repositórios:**

- Instanciar um repositório de pessoas físicas (repo1).
- Adicionar duas pessoas físicas, utilizando o construtor completo.
- Invocar o método de persistência em repo1, fornecendo um nome de
  - arquivo fixo, através do código.
- Instanciar outro repositório de pessoas físicas (repo2).
- Invocar o método de recuperação em repo2, fornecendo o mesmo
  - nome de arquivo utilizado anteriormente.
- Exibir os dados de todas as pessoas físicas recuperadas.
- Instanciar um repositório de pessoas jurídicas (repo3).
- Adicionar duas pessoas jurídicas, utilizando o construtor completo.
- Invocar o método de persistência em repo3, fornecendo um nome de
  - arquivo fixo, através do código.
- Instanciar outro repositório de pessoas jurídicas (repo4).
- Invocar o método de recuperação em repo4, fornecendo o mesmo
  - nome de arquivo utilizado anteriormente.
- Exibir os dados de todas as pessoas jurídicas recuperadas.

## Códigos da Classe Main com as devidas alterações conforme solicitado acima.

```
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   import java.io.IOException;
8
9   /**
10    *
11    * @author TiagodeJesus
12    */
13   public class Main {
14
15       public static void main(String[] args) {
16           PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
17           repo1.inserir(new PessoaFisica(1, " Julia", "33344455566", 40));
18           repo1.inserir(new PessoaFisica(2, " Mario", "66677788899", 36));
19
20           try {
21               repo1.persistir("pessoas_fisicas.dat");
22               System.out.println("Dados de Pessoa Física Armazenados.");
23           } catch (IOException e) {
24               System.out.println("Erro ao persistir dados de pessoas físicas: " + e.getMessage());
25           }
26
27           PessoaFisicaRepo repo2 = null;
28           try {
29               repo2 = PessoaFisicaRepo.recuperar("pessoas_fisicas.dat");
30           } catch (IOException | ClassNotFoundException e) {
31               System.out.println("Erro ao recuperar dados de pessoas físicas: " + e.getMessage());
32           }
33
34           if (repo2 != null) {
35               System.out.println("Dados de pessoa física recuperada.");
36               for (PessoaFisica pessoa : repo2.obterTodos()) {
37                   pessoa.exibir();
38               }
39           }
40
41           PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
42           repo3.inserir(new PessoaJuridica(3, " Empresa XPTO Sales", "11109876543210"));
43           repo3.inserir(new PessoaJuridica(4, " Empresa XPTO Solutions", "01234567891011"));
44
45           try {
46               repo3.persistir("pessoas_juridicas.dat");
47               System.out.println("Dados de Pessoa Jurídica Armazenados.");
48           } catch (IOException e) {
49               System.out.println("Erro ao persistir dados de pessoas jurídicas: " + e.getMessage());
50           }
51
52           PessoaJuridicaRepo repo4 = null;
53           try {
54               repo4 = PessoaJuridicaRepo.recuperar("pessoas_juridicas.dat");
55           } catch (IOException | ClassNotFoundException e) {
56               System.out.println("Erro ao recuperar dados de pessoas jurídicas: " + e.getMessage());
57           }
58
59           if (repo4 != null) {
60               System.out.println("Pessoas Jurídicas Recuperadas.");
61               for (PessoaJuridica pessoa : repo4.obterTodos()) {
62                   pessoa.exibir();
63               }
64           }
65       }
66   }
67
68 }
```



## 4 - Os resultados da execução dos códigos também devem ser apresentados:

```
run:

Dados de Pessoa Física Armazenados.
Dados de pessoa física recuperada.
ID:1
Nome: Julia
CPF: 33344455566
Idade: 40

ID:2
Nome: Mario
CPF: 66677788899
Idade: 36

Dados de Pessoa Jurídica Armazenados.
Pessoas Jurídicas Recuperadas.
ID:3
Nome: Empresa XPTO Sales
CNPJ: 11109876543210

ID:4
Nome: Empresa XPTO Solutions
CNPJ: 01234567891011

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

## 5 – Análise e Conclusão:

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

#### ✓ Vantagens

- **Reutilização de código:** Com a herança, você pode criar uma nova classe que é baseada em uma classe existente. Isso permite a reutilização do código já escrito na classe base.
- **Extensibilidade:** A herança permite estender o comportamento de uma classe existente, adicionando novos métodos e propriedades à subclasse.
- **Polimorfismo:** Classes derivadas podem ser tratadas como a classe base em determinadas situações, o que facilita o polimorfismo e a flexibilidade do código.

#### ✓ Desvantagens

- **Acoplamento forte:** Muita herança pode levar a um acoplamento forte entre classes, o que torna o código mais difícil de entender e manter.
- **Herança múltipla não suportada:** Algumas linguagens, como Java, não suportam a herança múltipla de classes. Isso pode limitar a flexibilidade do design.
- **Complexidade:** Uma hierarquia de classes muito profunda pode levar à complexidade do código e dificultar a compreensão.

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

Ao implementar a interface **Serializable**, você está indicando que a classe pode ser serializada e desserializada, ou seja, pode ser transformada em bytes e depois recriada a partir desses bytes. Isso é útil em diversas situações, como quando você precisa enviar objetos entre diferentes processos Java ou quando precisa armazenar objetos em um cache distribuído. Segue abaixo alguns exemplos:

- **Persistência de Objetos:** A interface **Serializable** em Java é usada para persistir objetos em um fluxo de bytes. Quando uma classe implementa a interface **Serializable**, os objetos dessa classe podem ser salvos em arquivos binários.

- **Comunicação de Rede:** A serialização é importante quando se trata de enviar objetos pela rede. Uma vez que os objetos são serializados, podem ser transferidos pela rede e recriados em outra máquina.

- **Armazenamento de Objetos:** A serialização permite que os objetos sejam armazenados em arquivos, para que possam ser recuperados posteriormente.

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

A API Stream em Java adota o paradigma funcional para operações de processamento de dados. Utilizando expressões lambda, a API Stream permite operações poderosas e concisas em coleções, contribuindo para um código mais limpo e legível. Expressões Lambda permitem a definição de funções anônimas concisamente. Operações de filtragem e mapeamento, como os métodos filter e map aplicam operações funcionalmente em elementos da coleção.

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

Ao trabalhar com Java, o padrão de desenvolvimento comumente adotado para persistência de dados em arquivos é o padrão DAO (Data Access Object). O DAO abstrai e encapsula todos os acessos aos dados, permitindo uma separação clara entre a lógica de negócios e as operações de acesso aos dados. Isso facilita a manutenção e a escalabilidade do código.

## **2 - Procedimento | Criação do Cadastro em Modo Texto**

### **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.**
- **persistência em arquivos binários.**

## Todos os códigos solicitados neste roteiro de aula:

```
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  import java.io.IOException;
8  import java.util.Scanner;
9
10 /**
11  *
12  * @author TiagodeJesus
13  */
14 public class Mainz {
15
16     private static Scanner scanner = new Scanner(System.in);
17     private static PessoaFisicaRepo repoPessoaFisica = new PessoaFisicaRepo();
18     private static PessoaJuridicaRepo repoPessoaJuridica = new PessoaJuridicaRepo();
19
20     public static void main(String[] args) {
21         int opcao;
22         do {
23             System.out.println("=====");
24             System.out.println("Opções:");
25             System.out.println("1. Incluir Pessoa");
26             System.out.println("2. Alterar Pessoa");
27             System.out.println("3. Excluir Pessoa");
28             System.out.println("4. Exibir Por ID");
29             System.out.println("5. Exibir Todos");
30             System.out.println("6. Salvar Dados");
31             System.out.println("7. Recuperar Dados");
32             System.out.println("0. Finalizar Execução");
33             System.out.println("=====");
34
35             opcao = lerInteiro("Digite a opção desejada: ");
36
37             switch (opcao) {
38                 case 1:
39                     incluir();
40                     break;
41                 case 2:
42                     alterar();
43                     break;
44                 case 3:
45                     excluir();
46                     break;
47                 case 4:
48                     exibirPorId();
49                     break;
50                 case 5:
51                     exibirTodos();
52                     break;
53                 case 6:
54                     salvarDados();
55                     break;
56                 case 7:
57                     recuperarDados();
58                     break;
59                 case 0:
60                     System.out.println("Finalizando o programa.");
61                     break;
62                 default:
63                     System.out.println("Opção inválida. Tente novamente.");
64             }
65         } while (opcao != 0);
66     }
67 }
```

### “Descrição do Código”

Apresentar as opções do programa para o usuário, sendo 1 para incluir, 2 para alterar, 3 para excluir, 4 para exibir pelo id, 5 para exibir todos, 6 para salvar dados, 7 para recuperar dados e 0 para finalizar a execução.

## Incluindo Pessoa Física e Pessoa Jurídica

```
79 private static void incluirPessoaFisica() {
80     int id = lerInteiro("Digite o ID: ");
81     String nome = lerString("Digite o nome: ");
82     String cpf = lerString("Digite o CPF: ");
83     int idade = lerInteiro("Digite a idade: ");
84     repoPessoaFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
85 }
86
87 private static void incluirPessoaJuridica() {
88     int id = lerInteiro("Digite o ID: ");
89     String nome = lerString("Digite o nome: ");
90     String cnpj = lerString("Digite o CNPJ: ");
91     repoPessoaJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
92 }
```

## Alterando Pessoa Física e Pessoa Jurídica

```
94 private static void alterar() {
95     String tipo = lerString("Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): ");
96     int id = lerInteiro("Digite o ID: ");
97     if (tipo.equalsIgnoreCase("F")) {
98         PessoaFisica pessoa = repoPessoaFisica.obter(id);
99         if (pessoa != null) {
100             System.out.println("Dados atuais da pessoa física:");
101             pessoa.exibir();
102             System.out.println("Digite os novos dados:");
103             incluirPessoaFisica();
104         } else {
105             System.out.println("Pessoa física não encontrada.");
106         }
107     } else if (tipo.equalsIgnoreCase("J")) {
108         PessoaJuridica pessoa = repoPessoaJuridica.obter(id);
109         if (pessoa != null) {
110             System.out.println("Dados atuais da pessoa jurídica:");
111             pessoa.exibir();
112             System.out.println("Digite os novos dados:");
113             incluirPessoaJuridica();
114         } else {
115             System.out.println("Pessoa jurídica não encontrada.");
116         }
117     } else {
118         System.out.println("Tipo inválido.");
119     }
120 }
```

“Descrição dos Códigos acima”

Selecionada a opção incluir, escolher o tipo (Física ou Jurídica), receber os dados a partir do teclado e adicionar no repositório correto.

Selecionada a opção alterar, escolher o tipo (Física ou Jurídica), receber o id a partir do teclado, apresentar os dados atuais, solicitar os novos dados e alterar no repositório correto.

## Excluir e Exibir o tipo de Pessoa Física e Jurídica

```
122 private static void excluir() {
123     String tipo = LerString("Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): ");
124     int id = LerInteiro("Digite o ID: ");
125     if (tipo.equalsIgnoreCase("F")) {
126         repoPessoaFisica.excluir(id);
127     } else if (tipo.equalsIgnoreCase("J")) {
128         repoPessoaJuridica.excluir(id);
129     } else {
130         System.out.println("Tipo inválido.");
131     }
132 }
133
134 private static void exibirPorId() {
135     String tipo = LerString("Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): ");
136     int id = LerInteiro("Digite o ID: ");
137     if (tipo.equalsIgnoreCase("F")) {
138         PessoaFisica pessoa = repoPessoaFisica.obter(id);
139         if (pessoa != null) {
140             pessoa.exibir();
141         } else {
142             System.out.println("Pessoa física não encontrada.");
143         }
144     } else if (tipo.equalsIgnoreCase("J")) {
145         PessoaJuridica pessoa = repoPessoaJuridica.obter(id);
146         if (pessoa != null) {
147             pessoa.exibir();
148         } else {
149             System.out.println("Pessoa jurídica não encontrada.");
150         }
151     } else {
152         System.out.println("Tipo inválido.");
153     }
154 }
```

“Descrição do Código”

Selecionada a opção excluir, escolher o tipo (Física ou Jurídica), receber o id a partir do teclado e remover do repositório correto.

Selecionada a opção exibir, escolher o tipo (Física ou Jurídica), receber o id a partir do teclado e apresentar os dados atuais para a entidade.

## Exibir e Salvar Pessoa Física e Jurídica

```
156 private static void exibirTodos() {
157     String tipo = LerString("Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): ");
158     if (tipo.equalsIgnoreCase("F")) {
159         for (PessoaFisica pessoa : repoPessoaFisica.obterTodos()) {
160             pessoa.exibir();
161             System.out.println();
162         }
163     } else if (tipo.equalsIgnoreCase("J")) {
164         for (PessoaJuridica pessoa : repoPessoaJuridica.obterTodos()) {
165             pessoa.exibir();
166             System.out.println();
167         }
168     } else {
169         System.out.println("Tipo inválido.");
170     }
171 }
172
173 private static void salvarDados() {
174     String prefixo = LerString("Digite o prefixo dos arquivos: ");
175     try {
176         repoPessoaFisica.persistir(prefixo + ".fisica.bin");
177         repoPessoaJuridica.persistir(prefixo + ".juridica.bin");
178         System.out.println("Dados salvos com sucesso.");
179     } catch (IOException e) {
180         System.out.println("Erro ao salvar dados: " + e.getMessage());
181     }
182 }
183 }
```

“Descrição do Código”

Selecionada a opção ExibirTodos, escolher o tipo (Física ou Jurídica) e apresentar os dados de todas as entidades do repositório correto.

Selecionada a opção salvar, solicitar o prefixo dos arquivos e persistir os dados nos arquivos [prefixo].fisica.bin e [prefixo].juridica.bin.



## Código utilizado na Recuperação dos dados

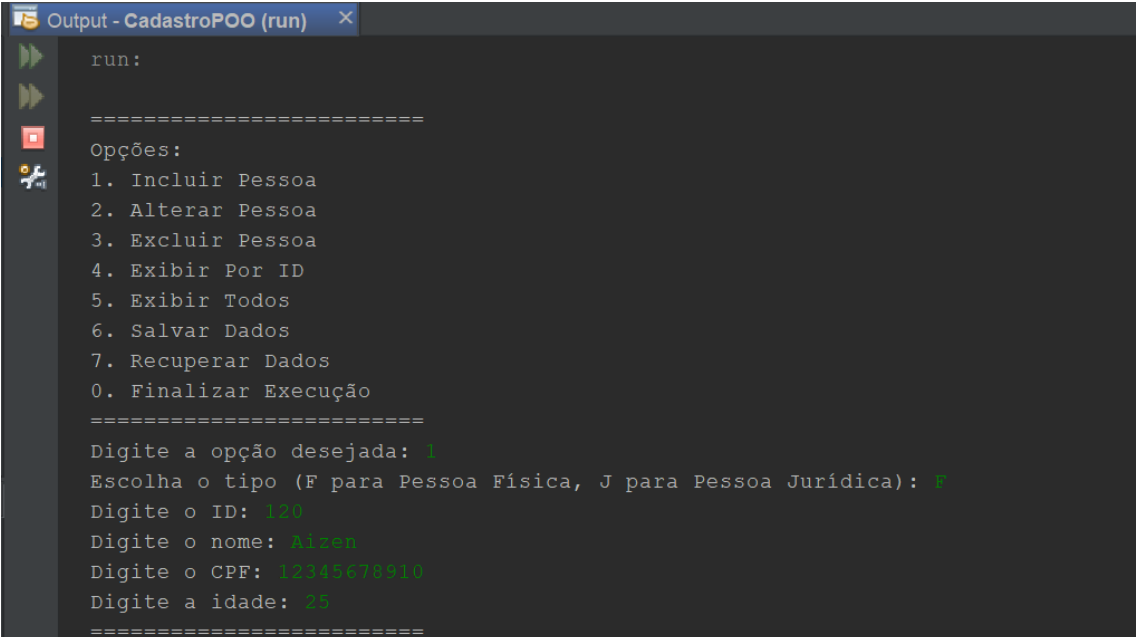
```
184 private static void recuperarDados() {
185     String prefixo = LerString("Digite o prefixo dos arquivos: ");
186     try {
187         repoPessoaFisica = PessoaFisicaRepo.recuperar(prefixo + ".fisica.bin");
188         repoPessoaJuridica = PessoaJuridicaRepo.recuperar(prefixo + ".juridica.bin");
189         System.out.println("Dados recuperados com sucesso.");
190     } catch (IOException | ClassNotFoundException e) {
191         System.out.println("Erro ao recuperar dados: " + e.getMessage());
192     }
193 }
194
195 private static int lerInteiro(String mensagem) {
196     System.out.print(mensagem);
197     return scanner.nextInt();
198 }
199
200 private static String lerString(String mensagem) {
201     System.out.print(mensagem);
202     return scanner.next();
203 }
204 }
205
```

### “Descrição do Código”

Selecionada a opção recuperar, solicitar o prefixo dos arquivos e obter os dados a partir dos arquivos [prefixo].fisica.bin e [prefixo].juridica.bin. Nas opções salvar e recuperar devem ser tratadas as exceções.

**Os resultados da execução dos códigos também devem ser apresentados:**

## Resultado do Código Incluir Pessoa



```
Output - CadastroPOO (run)
run:
=====
Opções:
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Exibir Por ID
5. Exibir Todos
6. Salvar Dados
7. Recuperar Dados
0. Finalizar Execução
=====
Digite a opção desejada: 1
Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): F
Digite o ID: 120
Digite o nome: Alzen
Digite o CPF: 12345678910
Digite a idade: 25
=====
```

## Resultado do Código Exibir Pessoa ID

```
Opções:
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Exibir Por ID
5. Exibir Todos
6. Salvar Dados
7. Recuperar Dados
0. Finalizar Execução
=====
Digite a opção desejada: 4
Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): F
Digite o ID: 120
ID:120
Nome:Aizen
CPF: 12345678910
Idade: 25
```

## Resultado do Código Alterar Pessoa

```
=====
Opções:
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Exibir Por ID
5. Exibir Todos
6. Salvar Dados
7. Recuperar Dados
0. Finalizar Execução
=====
Digite a opção desejada: 2
Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): F
Digite o ID: 120
Dados atuais da pessoa física:
ID:120
Nome:Aizen
CPF: 12345678910
Idade: 25

Digite os novos dados:
Digite o ID: 121
Digite o nome: Aizen
Digite o CPF: 12345678910
Digite a idade: 25
=====
```

## Resultado do Código Exibir Todos

```
=====
Opções:
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Exibir Por ID
5. Exibir Todos
6. Salvar Dados
7. Recuperar Dados
0. Finalizar Execução
=====
Digite a opção desejada: 5
Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): F
ID:120
Nome:Aizen
CPF: 12345678910
Idade: 25

ID:121
Nome:Aizen
CPF: 12345679910
Idade: 25
```

## Resultado do Código Salvar Todos

```
=====
Opções:
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Exibir Por ID
5. Exibir Todos
6. Salvar Dados
7. Recuperar Dados
0. Finalizar Execução
=====
Digite a opção desejada: 6
Digite o prefixo dos arquivos: aizen
Dados salvos com sucesso.
=====
```

- build
- nbproject
- src
- test
- aizen.fisica.bin
- aizen.juridica.bin
- build
- manifest.mf
- peessoas\_fisicas.dat
- peessoas\_juridicas.dat

Arquivos criando  
atraves da opção salvar  
todos.

Pasta de arquivos	
Pasta de arquivos	
Pasta de arquivos	
Pasta de arquivos	
Arquivo BIN	1 KB
Arquivo BIN	1 KB
Documento XML	4 KB
Arquivo MF	1 KB
Arquivo DAT	1 KB
Arquivo DAT	1 KB

## Resultado do Código Excluir Pessoa

```
Opções:
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Exibir Por ID
5. Exibir Todos
6. Salvar Dados
7. Recuperar Dados
0. Finalizar Execução
=====
Digite a opção desejada: 3
Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): F
Digite o ID: 120
=====
Opções:
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Exibir Por ID
5. Exibir Todos
6. Salvar Dados
7. Recuperar Dados
0. Finalizar Execução
=====
Digite a opção desejada: 3
Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): F
ID:121
Nome:Aizen
CPF: 12345679910
Idade: 25
```

## Resultado do Código Recuperar Dados

```
Opções:
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Exibir Por ID
5. Exibir Todos
6. Salvar Dados
7. Recuperar Dados
0. Finalizar Execução
=====
Digite a opção desejada: 7
Digite o prefixo dos arquivos: aizen7
Erro ao recuperar dados: aizen2.fisica.bin (O sistema não pode encontrar o arquivo especificado)
=====
Opções:
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Exibir Por ID
5. Exibir Todos
6. Salvar Dados
7. Recuperar Dados
0. Finalizar Execução
=====
Digite a opção desejada: 7
Digite o prefixo dos arquivos: aizen
Dados recuperados com sucesso.
=====
Opções:
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Exibir Por ID
5. Exibir Todos
6. Salvar Dados
7. Recuperar Dados
0. Finalizar Execução
=====
Digite a opção desejada: 0
Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica): F
ID:120
Nome:Aizen
CPF: 12345678910
Idade: 25

ID:121
Nome:Aizen
CPF: 12345679910
Idade: 25
```

## Resultado do Código Finalizar Execução

```
=====
Opções:
1. Incluir Pessoa
2. Alterar Pessoa
3. Excluir Pessoa
4. Exibir Por ID
5. Exibir Todos
6. Salvar Dados
7. Recuperar Dados
0. Finalizar Execução
=====
Digite a opção desejada: 0
Finalizando o programa.
BUILD SUCCESSFUL (total time: 25 minutes 9 seconds)
```

## **5 - Análise e Conclusão:**

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

Elementos estáticos em Java são aqueles que pertencem à classe em si, em vez de pertencerem a instâncias específicas da classe.

Isso significa que eles podem ser acessados sem a necessidade de criar um objeto da classe. Exemplos de elementos estáticos são métodos e variáveis estáticas.

O método main em Java é declarado como estático para que possa ser invocado sem a necessidade de instanciar a classe. Isso permite que o programa seja executado sem a criação de um objeto da classe principal, facilitando o início da execução do programa.

### **❖ Para que serve a classe Scanner?**

A classe Scanner em Java é usada para ler dados de entrada, como texto digitado pelo usuário no teclado ou informações de arquivos. Ela é frequentemente utilizada para permitir a interação do usuário com programas, facilitando a leitura e análise de dados formatados. Em resumo, o Scanner é uma ferramenta importante para a entrada de dados interativa em programas Java.

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

O uso de classes de repositório em um projeto Java contribui para a organização eficiente do código, promovendo a separação nítida da lógica de acesso a dados. Isso resulta em uma melhor reutilização de código, abstração da camada de armazenamento, maior testabilidade, clareza e manutenção do código, além da capacidade de organizar hierarquicamente as operações de dados para refletir a estrutura de dados do projeto, simplificando o desenvolvimento e a escalabilidade do software.