



**RELATÓRIO DISCENTE DE ACOMPANHAMENTO
DESENVOLVIMENTO FULL STACK**

TURMA - 2023.1

PERÍODO – 2024.3

CAMPUS - VIRTUAL

MATRICULA - 202202923931

POLO BARREIRO - Belo Horizonte MG

ALUNO - Rian Joseph Ramos Felizardo

REPOSITÓRIO

[desenvolvimento-sistemas-mundo-3/nivel-01/missao-pratica-01 at main · rianjsp/desenvolvimento-sistemas-mundo-3 \(github.com\)](#)

RPG0014 - Iniciando o caminho pelo Java

Objetivos 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.

1º Procedimento - Criação das Entidades e Sistema de Persistência

(1º e 2º) Foi realizado a criação do projeto de nome CadastroPOO utilizando o Java With Ant, sendo realizado após este procedimento a criação do pacote Model onde foram criadas as entidades Pessoa, PessoaFisica e PessoaJuridica, utilizando elementos de herança entre as classes e polimorfismo.

3º – Criar as Entidades:

Código Classe Pessoa

```
1  [+ ...4 lines
5  package model;
6  [- import java.io.Serializable;
7
8  [+ /**...4 lines */
9  @ public class Pessoa implements Serializable {
13     // Atributos
14     private int id;
15     private String nome;
16
17     // Construtor
18     [- public Pessoa() {}
19
20     // Construtor completo
21     [- public Pessoa(int id, String nome){
22         this.id = id;
23         this.nome = nome;
24     }
25
26     // Getters e Setters
27     [- public int getId(){
28         return this.id;
29     }
30
31     [- public void setId(int id){
32         this.id = id;
33     }
34
35     [- public String getNome(){
36         return this.nome;
37     }
38
39     [- public void setNome(String nome){
40         this.nome = nome;
41     }
42
43     // Exibir separado
44     @ [- public void exibir(){
45         System.out.println("ID: " + id);
46         System.out.println("Nome: " + nome);
47     }
48 }
49
```

Código Classe PessoaFisica

```
1  [+ ...4 lines
5  package model;
6
7  [+ /**...4 lines */
11 public class PessoaFisica extends Pessoa {
12     // Atributos
13     private String cpf;
14     private int idade;
15
16     // Construtor
17     [- public PessoaFisica(){}
18
19     // Construtor completo
20     public PessoaFisica(int id, String nome,
21     [- String cpf, int idade){
22         super(id, nome);
23         this.cpf = cpf;
24         this.idade = idade;
25     }
26
27     // Getters e Setters
28     [- public String getCpf(){
29         return this.cpf;
30     }
31
32     [- public void setCpf(String cpf){
33         this.cpf = cpf;
34     }
35
36     [- public int getIdade(){
37         return idade;
38     }
39
40     [- public void setIdade(int idade){
41         this.idade = idade;
42     }
43
44     // Exibir polimorfico
45     @Override
46     [- public void exibir(){
47         super.exibir();
48         System.out.println("CPF: "+ cpf);
49         System.out.println("Idade: "+ idade);
50     }
51 }
52
```

Código classe PessoaJuridica

```
1  [+ ...4 lines
5  package model;
6
7  [+ /**...4 lines */
11 public class PessoaJuridica extends Pessoa {
12     // Atributos
13     private String cnpj;
14
15     // Construtor
16     [- public PessoaJuridica() {}
17
18     // Construtor Completo
19     public PessoaJuridica(int id, String nome,
20         [- String cnpj) {
21         |     super(id, nome);
22         |     this.cnpj = cnpj;
23         | }
24
25     // Getters e Setters
26     [- public String getCnpj() {
27     |     return cnpj;
28     | }
29
30     [- public void setCnpj(String cnpj) {
31     |     this.cnpj = cnpj;
32     | }
33
34     // Exibir polimorfico
35     @Override
36     [- public void exibir() {
37     |     super.exibir();
38     |     System.out.println("CNPJ: " + cnpj);
39     | }
40 }
41
```

4º – Criar os Gerenciadores:

Foi feita a implementação de um ArrayList privado para armazenar as entidades criadas PessoaFisicaRepo e PessoaJuridicaRepo, além de adicionar os métodos inserir, alterar, excluir, obter, e obterTodos para gerenciar as entidades no ArrayList.

Código PessoaJuridicaRepo e Código PessoaFisicaRepo

```
16 public class PessoaJuridicaRepo {
17     // Definindo um array privado
18     private ArrayList<PessoaJuridica> listaPessoaJuridica = new ArrayList<>();
19
20     // Metodo inserir uma nova PessoaJuridica
21     public void inserir(PessoaJuridica pessoaJuridica) {
22         listaPessoaJuridica.add(pessoaJuridica);
23     }
24
25     // Metodo alterar uma pessoaJuridica existente
26     public void alterar(PessoaJuridica pessoaJuridica) {
27         for(int i = 0; i < listaPessoaJuridica.size(); i++)
28         {
29             if(listaPessoaJuridica.get(i).getId() == pessoaJuridica.getId()){
30                 listaPessoaJuridica.set(i, pessoaJuridica);
31                 break;
32             }
33         }
34     }
35
36     // Metodo excluir PessoaJuridica pelo id
37     public void excluir(int id){
38         listaPessoaJuridica.removeIf(p -> p.getId() == id);
39     }
40
41     // Metodo obter pessoaJuridica pelo id
42     public PessoaJuridica obter(int id){
43         for(PessoaJuridica p : listaPessoaJuridica)
44         {
45             if(p.getId() == id){
46                 return p;
47             }
48         }
49         return null;
50     }
51
52     // Metodo obter todas pessoasJuridicas
53     public ArrayList<PessoaJuridica> obterTodos() {
54         return listaPessoaJuridica;
55     }
56
57     // Metodo persistencia de dados
58     public void persistir(String nomeArquivo) throws IOException {
59         try (ObjectOutputStream oos = new ObjectOutputStream
60             (new FileOutputStream(nomeArquivo))) {
61             oos.writeObject(listaPessoaJuridica);
62         }
63     }
64
65     // Metodo para recuperar os dados do arquivo
66     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
67         try (ObjectInputStream ois = new ObjectInputStream
68             (new FileInputStream(nomeArquivo))) {
69             listaPessoaJuridica = (ArrayList<PessoaJuridica>) ois.readObject();
70         }
71     }
72 }
```

```
15 public class PessoaFisicaRepo {
16     // Definindo um array privado
17     private ArrayList<PessoaFisica> listaPessoaFisica = new ArrayList<>();
18
19     // Metodo inserir nova pessoa
20     public void inserir(PessoaFisica pessoaFisica) {
21         listaPessoaFisica.add(pessoaFisica);
22     }
23
24     // Metodo alterar uma pessoasFisica existente
25     public void alterar(PessoaFisica pessoaFisica) {
26         for(int i = 0; i < listaPessoaFisica.size(); i++)
27         {
28             if(listaPessoaFisica.get(i).getId() == pessoaFisica.getId()){
29                 listaPessoaFisica.set(i, pessoaFisica);
30                 break;
31             }
32         }
33     }
34
35     // Metodo excluir pessoasFisica por id
36     public void excluir(int id){
37         listaPessoaFisica.removeIf(p -> p.getId() == id);
38     }
39
40     // Metodo obter pessoasFisica pelo id
41     public PessoaFisica obter(int id){
42         for(PessoaFisica p : listaPessoaFisica)
43         {
44             if(p.getId() == id){
45                 return p;
46             }
47         }
48         return null;
49     }
50
51     // Metodo obter todas as pessoasFisica
52     public ArrayList<PessoaFisica> obterTodos() {
53         return listaPessoaFisica;
54     }
55
56     // Metodo para persistencia de dados em arquivos
57     public void persistir(String nomeArquivo) throws IOException {
58         try (ObjectOutputStream oos = new ObjectOutputStream
59             (new FileOutputStream(nomeArquivo))) {
60             oos.writeObject(listaPessoaFisica);
61         }
62     }
63
64     // Metodo para recuperar os dados
65     @SuppressWarnings("unchecked")
66     public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
67         try (ObjectInputStream ois = new ObjectInputStream
68             (new FileInputStream(nomeArquivo))) {
69             listaPessoaFisica = (ArrayList<PessoaFisica>) ois.readObject();
70         }
71     }
72 }
```

5º – Alterar o método Main da classe principal para testar os repositórios:

Foi realizado a instanciação dos Repositórios de PessoaFisica (Repo_01, Repo_02) onde o repo_01 e feita a adição das pessoas com o construtor completo e é feita a chamada do método para realizar a persistência, e o repo_02 e usado para recuperar os dados pelo arquivo 'pessoas_fisicas.dat'. Após a recuperação do arquivo o programa exibe as informações das pessoas físicas.

Os repositórios de pessoas jurídicas também foram testados, onde no repo_03 duas pessoas jurídicas foram adicionadas também utilizando o construtor completo e feito a chamada do método de persistência para salvar os dados no arquivo 'pessoas_juridicas.dat', e o repo_04 foi o repositório utilizado para recuperar os dados do arquivo 'pessoas_juridicas.dat'.

Foi realizado o tratamento de exceções onde mensagens de erro serão exibidas no console caso ocorram.

Código CadastroPOO

```
1  ...4 lines
2  package cadastropoo;
3
4  //Imports
5  import model.PessoaFisica;
6  import model.PessoaFisicaRepo;
7  import model.PessoaJuridica;
8  import model.PessoaJuridicaRepo;
9  import java.io.IOException;
10 import java.util.ArrayList;
11
12 /**...4 lines */
13 public class CadastroPOO {
14
15     /**
16      * @param args the command line arguments
17      */
18
19     // Metodo Main
20     public static void main(String[] args) {
21         // Repositorio de pessoas fisicas repo_01
22         PessoaFisicaRepo repo_01 = new PessoaFisicaRepo();
23
24         // Add 4 pessoas fisicas utilizando o construtor completo
25         repo_01.inserir(new PessoaFisica(1, "Joao Silva", "123.456.789.10", 38));
26         repo_01.inserir(new PessoaFisica(2, "Maria Alves", "453.336.459.20", 45));
27         repo_01.inserir(new PessoaFisica(3, "Marta Rosa", "673.357.789.10", 26));
28         repo_01.inserir(new PessoaFisica(4, "Caio Artur", "231.456.689.23", 24));
29
30         // Persistindo os dados do repo_01 em um arqv
31         String arquivoFisicas = "pessoas_fisicas.dat";
32         try {
33             repo_01.persistir(arquivoFisicas);
34             System.out.println("Dados de Pessoa fisica armazenados.");
35         } catch (IOException e) {
36             System.out.println("Aconteceu um erro ao persistir dados de pessoas fisicas: " + e.getMessage());
37         }
38
39         //Repositorio de pessoas fisicas repo_02
40         PessoaFisicaRepo repo_02 = new PessoaFisicaRepo();
41         try {
42             repo_02.recuperar(arquivoFisicas);
43             System.out.println("Dados de Pessoa fisicas Recuperados.");
44         } catch (IOException | ClassNotFoundException e) {
45             System.out.println("Aconteceu um erro ao recuperar dados de pessoas fisicas: " + e.getMessage());
46         }
47
48         // Exibindo os dados
49         ArrayList<PessoaFisica> pessoasFisicas = repo_02.obterTodos();
50         for(PessoaFisica pessoa : pessoasFisicas) {
51             pessoa.exibir();
52         }
53
54         // Repositorio de pessoas juridicas repo_03
55         PessoaJuridicaRepo repo_03 = new PessoaJuridicaRepo();
56         // Adicionando 2 pessoas utilizando o construtor completo
57         repo_03.inserir(new PessoaJuridica(1, "Empresa Primeira", "12.345.678/0001-99"));
58         repo_03.inserir(new PessoaJuridica(2, "Empresa Segunda", "12.345.678/0002-88"));
59
60         // Persistindo os dados do repo_03 em um arqv
61         String arquivoJuridicas = "pessoas_juridicas.dat";
62         try {
63             repo_03.persistir(arquivoJuridicas);
64             System.out.println("Dados de pessoas juridicas armazenados.");
65         } catch (IOException e) {
66             System.out.println("Aconteceu um erro ao armazenar dados de pessoas juridicas: " + e.getMessage());
67         }
68
69         // Repositorio de pessoas juridicas repo_04
70         PessoaJuridicaRepo repo_04 = new PessoaJuridicaRepo();
71         try {
72             repo_04.recuperar(arquivoJuridicas);
73             System.out.println("Dados de pessoas juridicas recuperados com sucesso.");
74         } catch (IOException | ClassNotFoundException e) {
75             System.out.println("Aconteceu um erro ao recuperar dados de pessoas juridicas: " + e.getMessage());
76         }
77
78         // Exibindo os dados
79         ArrayList<PessoaJuridica> pessoasJuridicas = repo_04.obterTodos();
80         for (PessoaJuridica pessoa : pessoasJuridicas) {
81             pessoa.exibir();
82         }
83     }
84 }
85
86
87
88
89
90
91
92
93
94
```



Análise e Conclusão

- Quais as vantagens do uso de herança?

A herança permite que uma classe (subclasse) herde atributos e métodos de outra classe (superclasse). Isso promove a reutilização de código, evitando duplicações e facilitando a manutenção.

Herança ajuda a estruturar o código de maneira hierárquica, o que torna mais fácil entender e gerenciar grandes sistemas.

Com a herança, as subclasses podem ser tratadas como instâncias de suas superclasses, permitindo o uso de polimorfismo, onde um método pode agir de maneira diferente dependendo da classe que o invoca.

- Quais as desvantagens do uso de herança?

A herança pode aumentar o acoplamento entre classes, tornando o sistema menos flexível e mais difícil de modificar.

Mudanças na superclasse podem afetar todas as subclasses, o que pode introduzir erros e aumentar o custo de manutenção.

O uso excessivo de herança pode levar a um design de código complexo e difícil de entender. Isso é frequentemente referido como "herança em árvore" que pode se tornar difícil de navegar.

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

A interface Serializable é necessária porque ela marca os objetos como "aptos" para serem convertidos em uma sequência de bytes. Esse processo, chamado de **serialização**, permite que o objeto seja salvo em um arquivo ou enviado pela rede. Ao ler o arquivo novamente, a **desserialização** recria o objeto original a partir dos bytes.

Sem a implementação de Serializable, o Java não saberia como transformar o objeto em bytes, impossibilitando a gravação correta no arquivo.

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

A API Stream do Java implementa conceitos do paradigma funcional, permitindo operações sobre coleções de dados de maneira mais declarativa. Isso é feito através de uma série de métodos que podem ser encadeados, como map, filter e reduce, possibilitando a transformação e a agregação de dados de forma clara e concisa. Por exemplo, usando streams, um desenvolvedor pode filtrar, transformar e coletar dados de uma lista em poucas linhas de código, tornando a leitura e a manutenção mais fáceis. Essa abordagem também permite aproveitar melhor os processadores multi-core, pois, operações em streams podem ser executadas em paralelo.

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

No Java, o padrão mais comum para persistência de dados em arquivos é o Data Access Object (DAO). Esse padrão separa a lógica de acesso a dados da lógica de negócios, organizando o código. As classes DAO lidam com operações de leitura e escrita, permitindo que você mude a forma como os dados são armazenados (como trocar arquivos por um banco de dados) sem afetar o restante da aplicação. Além disso, para salvar objetos em arquivos, usa-se a interface `Serializable`, que facilita a conversão de objetos em bytes.

Output do primeiro procedimento

```
Output - CadastroPOO (run)

run:
Dados de Pessoa fisica armazenados.
Dados de Pessoa fisicas Recuperados.
ID: 1
Nome: Joao Silva
CPF: 123.456.789.10
Idade: 38
ID: 2
Nome: Maria Alves
CPF: 453.336.459.20
Idade: 45
ID: 3
Nome: Marta Rosa
CPF: 673.357.789.10
Idade: 26
ID: 4
Nome: Caio Artur
CPF: 231.456.689.23
Idade: 24
Dados de pessoas juridicas armazenados.
Dados de pessoas juridicas recuperados com sucesso.
ID: 1
Nome: Empresa Primeira
CNPJ: 12.345.678/0001-99
ID: 2
Nome: Empresa Segunda
CNPJ: 12.345.678/0002-88
BUILD SUCCESSFUL (total time: 0 seconds)
|
```


2º Procedimento - Criação do Cadastro em Modo Texto

Código Completo classe CadastroPOO

```
CadastroPOO.java | PessoaFisica.java | PessoaJuridica.java | PessoaFisicaRepo.java | PessoaJuridicaRepo.java
Source History
10 import java.util.InputMismatchException;
11 import java.util.Scanner;
12
13 public class CadastroPOO {
14
15     public static void main(String[] args) {
16         Scanner scanner = new Scanner(System.in);
17         PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
18         PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();
19
20         int opcao;
21         do {
22             System.out.println("=====");
23             System.out.println("Escolha uma opcao:");
24             System.out.println("1 - Incluir Pessoa");
25             System.out.println("2 - Alterar Pessoa");
26             System.out.println("3 - Excluir Pessoa");
27             System.out.println("4 - Buscar pelo ID");
28             System.out.println("5 - Exibir todos");
29             System.out.println("6 - Persistir dados");
30             System.out.println("7 - Recuperar dados");
31             System.out.println("0 - Finalizar programa");
32             System.out.println("=====");
33
34             while (true) {
35                 try {
36                     opcao = scanner.nextInt();
37                     scanner.nextLine();
38                     break;
39                 } catch (InputMismatchException e) {
40                     System.out.println("Insira um numero valido.");
41                     scanner.nextLine();
42                 }
43             }
44
45             switch (opcao) {
46                 case 1: // Incluir
47                     System.out.println("F - Fisica | J - Juridica:");
48                     String tipoInclusao = scanner.nextLine().toUpperCase();
49                     if (tipoInclusao.equals("F")) {
50                         System.out.print("Digite o ID da pessoa: ");
51                         int id = scanner.nextInt();
52                         scanner.nextLine();
53
54                         if (repoFisica.obter(id) != null) {
55                             System.out.println("ID ja existe. Escolha um ID unico.");
56                             break;
57                         }
58
59                         System.out.print("Digite o nome da pessoa: ");
60                         String nome = scanner.nextLine();
61                         System.out.print("Digite o CPF da pessoa: ");
62                         String cpf = scanner.nextLine();
63                         System.out.print("Digite a idade da pessoa: ");
64                         int idade = scanner.nextInt();
65
66                         repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
67                         System.out.println("Pessoa Fisica cadastrada com sucesso!");
68                     } else if (tipoInclusao.equals("J")) {
69                         System.out.print("Digite o ID da pessoa: ");
70                         int id = scanner.nextInt();
71                         scanner.nextLine();
72
73                         if (repoJuridica.obter(id) != null) {
74                             System.out.println("ID ja existe. Escolha um ID unico.");
75                             break;
76                         }
77
78                         System.out.print("Digite o nome da pessoa: ");
79                         String nome = scanner.nextLine();
80                         System.out.print("Digite o CNPJ da pessoa: ");
81                         String cnpj = scanner.nextLine();
82                         repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
83                         System.out.println("Pessoa Juridica cadastrada com sucesso!");
84                     } else {
85                         System.out.println("Tipo invalido. Escolha 'F' ou 'J'.");
86                         break;
87                     }
88
89                     case 2: // Alterar
90                         System.out.println("F - Fisica | J - Juridica:");
91                         String tipoAlteracao = scanner.nextLine().toUpperCase();
92                         System.out.print("Digite o ID: ");
93                         int idAlterar = scanner.nextInt();
94                         scanner.nextLine();
95                         if (tipoAlteracao.equals("F")) {
96                             PessoaFisica pessoa = repoFisica.obter(idAlterar);
97                             if (pessoa != null) {
98                                 System.out.println("Dados atuais: " + pessoa);
99                                 System.out.print("Digite o novo nome: ");
100                                 String novoNome = scanner.nextLine();
101                                 System.out.print("Digite o novo CPF: ");
102                                 String novoCpf = scanner.nextLine();
103                                 System.out.print("Digite a nova idade: ");
104                                 int novaIdade = scanner.nextInt();
105                                 pessoa.setNome(novoNome);
106                                 pessoa.setCpf(novoCpf);
107                                 pessoa.setIdade(novaIdade);
108                                 repoFisica.alterar(pessoa);
109                                 System.out.println("Pessoa Fisica alterada com sucesso!");
110                             } else {
111                                 System.out.println("Pessoa nao encontrada.");
112                             }
113                         } else if (tipoAlteracao.equals("J")) {
114                             PessoaJuridica pessoa = repoJuridica.obter(idAlterar);
115                             if (pessoa != null) {
116                                 System.out.println("Dados atuais: " + pessoa);
117                                 System.out.print("Digite o novo nome: ");
118                                 String novoNome = scanner.nextLine();
119                                 System.out.print("Digite o novo CNPJ: ");
120                                 String novoCnpj = scanner.nextLine();
121                                 pessoa.setNome(novoNome);
122                                 pessoa.setCnpj(novoCnpj);
123                                 repoJuridica.alterar(pessoa);
124                                 System.out.println("Pessoa Juridica alterada com sucesso!");
125                             } else {
126                                 System.out.println("Pessoa nao encontrada.");
127                             }
128                         }
129                     }
130                     break;
131                 case 3: // Excluir
132                     System.out.println("F - Fisica | J - Juridica:");
133                     String tipoExclusao = scanner.nextLine().toUpperCase();
134                     System.out.print("Digite o ID: ");
135                     int idExcluir = scanner.nextInt();
136                     scanner.nextLine();
137                     if (tipoExclusao.equals("F")) {
138                         repoFisica.excluir(idExcluir);
139                         System.out.println("Pessoa Fisica excluida.");
140                     } else if (tipoExclusao.equals("J")) {
141                         repoJuridica.excluir(idExcluir);
142                         System.out.println("Pessoa Juridica excluida.");
143                     } else {
144                         System.out.println("Tipo invalido. Escolha 'F' ou 'J'.");
145                         break;
146                     }
147                     break;
148                 case 4: // Exibir pelo ID
149                     System.out.println("Escolha o tipo (F - Fisica | J - Juridica):");
150                     String tipoExibir = scanner.nextLine().toUpperCase();
151                     System.out.print("Digite o ID: ");
152                     int idExibir = scanner.nextInt();
153                     scanner.nextLine();
154                     if (tipoExibir.equals("F")) {
155                         PessoaFisica pessoa = repoFisica.obter(idExibir);
156                         System.out.println(pessoa != null ? pessoa : "Pessoa nao encontrada.");
157                     } else if (tipoExibir.equals("J")) {
158                         PessoaJuridica pessoa = repoJuridica.obter(idExibir);
159                         System.out.println(pessoa != null ? pessoa : "Pessoa nao encontrada.");
160                     } else {
161                         System.out.println("Tipo invalido. Escolha 'F' ou 'J'.");
162                         break;
163                     }
164                     break;
165                 case 5: // Exibir todos
166                     System.out.println("Escolha o tipo (F - Fisica | J - Juridica):");
167                     String tipoExibirTodos = scanner.nextLine().toUpperCase();
168                     if (tipoExibirTodos.equals("F")) {
169                         ArrayList<PessoaFisica> pessoasFisicas = repoFisica.obterTodos();
170                         if (pessoasFisicas.isEmpty()) {
171                             System.out.println("Nenhuma pessoa fisica cadastrada.");
172                         } else {
173                             for (PessoaFisica pessoa : pessoasFisicas) {
174                                 System.out.println(pessoa);
175                             }
176                         }
177                     } else if (tipoExibirTodos.equals("J")) {
178                         ArrayList<PessoaJuridica> pessoasJuridicas = repoJuridica.obterTodos();
179                         if (pessoasJuridicas.isEmpty()) {
180                             System.out.println("Nenhuma pessoa juridica cadastrada.");
181                         } else {
182                             for (PessoaJuridica pessoa : pessoasJuridicas) {
183                                 System.out.println(pessoa);
184                             }
185                         }
186                     } else {
187                         System.out.println("Tipo invalido. Escolha 'F' ou 'J'.");
188                         break;
189                     }
190                     break;
191                 case 6: // Salvar dados
192                     System.out.print("Digite o prefixo dos arquivos: ");
193                     String prefixoSalvar = scanner.nextLine();
194                     try {
195                         repoFisica.persistir(prefixoSalvar + ".fisica.bin");
196                         repoJuridica.persistir(prefixoSalvar + ".juridica.bin");
197                         System.out.println("Dados salvos com sucesso.");
198                     } catch (IOException e) {
199                         System.out.println("Erro ao salvar dados: " + e.getMessage());
200                     }
201                     break;
202                 case 7: // Recuperar dados
203                     System.out.print("Digite o prefixo dos arquivos: ");
204                     String prefixoRecuperar = scanner.nextLine();
205                     try {
206                         repoFisica.recuperar(prefixoRecuperar + ".fisica.bin");
207                         repoJuridica.recuperar(prefixoRecuperar + ".juridica.bin");
208                         System.out.println("Dados recuperados com sucesso.");
209                     } catch (IOException | ClassNotFoundException e) {
210                         System.out.println("Erro ao recuperar dados: " + e.getMessage());
211                     }
212                     break;
213                 case 0: // Sair
214                     System.out.println("Sistema finalizado.");
215                     break;
216                 default:
217                     System.out.println("Opcao invalida. Tente novamente.");
218             }
219         } while (opcao != 0);
220         scanner.close();
221     }
222 }
```

A estrutura do código é organizada e modular, utilizando classes de repositório para gerenciar dados, o que torna a manutenção e a reutilização mais simples. A interação com o usuário é simples, mas eficaz, através de um menu que oferece opções bem claras. No geral, o sistema é eficiente para o cadastro e manipulação de informações de pessoas e está de acordo com o enunciado do trabalho.

Foi realizado alterações nas classes de repositório para termos um resultado final no output mais agradável.

Output esperado operante

Output - CadastroPOO (run)

```
run:
=====
Escolha uma opcao:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir todos
6 - Persistir dados
7 - Recuperar dados
0 - Finalizar programa
=====

1
F - Fisica | J - Juridica):
F
Digite o ID da pessoa: 90
Digite o nome da pessoa: Rian Joseph
Digite o CPF da pessoa: 123456789
Digite a idade da pessoa: 23
Pessoa Fisica cadastrada com sucesso!
=====

Escolha uma opcao:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir todos
6 - Persistir dados
7 - Recuperar dados
0 - Finalizar programa
=====

2
F - Fisica | J - Juridica):
F
Digite o ID: 90
Dados atuais: PessoaFisica{id=90, nome='Rian Joseph', cpf='123456789', idade=23}
Digite o novo nome: Joseph Rian
Digite o novo CPF: 12354345123
Digite a nova idade: 24
Pessoa Fisica alterada com sucesso!
=====

Escolha uma opcao:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo ID
5 - Exibir todos
6 - Persistir dados
7 - Recuperar dados
0 - Finalizar programa
=====
```

Output



Analise e Conclusão

- O que são elementos estáticos e por que o método main é estático?

Elementos estáticos são aqueles que pertencem à classe em si, e não a instâncias específicas dela. Isso significa que você pode usá-los sem precisar criar um objeto da classe. O método main, por exemplo, é estático porque precisa ser chamado pelo sistema para iniciar o programa antes de qualquer objeto ser criado. Isso permite que o Java execute o código diretamente da classe, facilitando o começo do programa. Em outras palavras, o main é como a porta de entrada da aplicação e precisa ser acessível sem depender de instâncias.

- Para que serve a classe Scanner?

A classe Scanner em Java é super útil para ler dados que o usuário insere, de forma simples e eficiente. Com ela, dá para capturar diferentes tipos de dados, como números inteiros, textos e decimais, e isso pode ser feito a partir de várias fontes, como o teclado ou arquivos.

- Como o uso de classes de repositório impactou a organização do código?

Usar classes de repositório deixou o código bem mais organizado e fácil de entender. Aqui estão alguns pontos principais sobre isso:

Separação de responsabilidades: Cada classe de repositório se encarrega de um tipo específico de dado, o que ajuda a saber onde cada parte do código está.

Reutilização: Posso usar as mesmas classes de repositório em diferentes partes do projeto, evitando a repetição de código.

Manutenção: Se eu precisar mudar algo sobre como os dados são armazenados ou recuperados, posso fazer isso em um único lugar, em vez de ter que alterar várias partes do código.

Resumindo, tudo isso torna o código mais limpo, fácil de trabalhar e mais agradável de lidar.

Código no GitHub: [desenvolvimento-sistemas-mundo-3/nivel-01/missao-pratica-01 at main · rianjsp/desenvolvimento-sistemas-mundo-3 \(github.com\)](https://github.com/rianjsp/desenvolvimento-sistemas-mundo-3/nivel-01/missao-pratica-01)