



Estácio

Informações Gerais:

- **Nome do Campus:** Nova América
- **Nome do Curso:** Desenvolvimento Full Stack
- **Nome da Disciplina:** RPG0014 - Iniciando o caminho pelo Java
- **Número da Turma:** 9001
- **Semestre:** 3º
- **Integrantes da Prática:** Camilla Rodrigues Alves Gomes

Título da Prática:

Missão Prática | Nível 1 | Mundo 3

1º Procedimento | Criação das Entidades e Sistema de Persistência

Objetivos da Prática:

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

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 Solicitados:

Pessoa.java

```
package model.entidades;
import java.io.Serializable;

public class Pessoa implements Serializable {
    protected int id;
    protected String nome;

    public void exibir(){
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
    }

    public Pessoa(){
    }

    public Pessoa(int id, String nome) {
        this.id = id;
        this.nome = nome;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("ID = ").append(id);
        sb.append(", Nome = ").append(nome);
        return sb.toString();
    }
}
```

PessoaFisica.java

```
import model.entidades.Pessoa;
import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    protected String cpf;
    protected int idade;

    @Override
    public void exibir(){
        super.exibir();
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
    }

    public PessoaFisica(){

    }

    public PessoaFisica(int id, String nome, String cpf, int idade) {
        super(id, nome);
        this.cpf = cpf;
        this.idade = idade;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public int getIdade() {
        return idade;
    }

    public void setIdade(int idade) {
        this.idade = idade;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("ID = ").append(id);
        sb.append(", Nome = ").append(nome);
        sb.append(", CPF = ").append(cpf);
        sb.append(", Idade = ").append(idade);
        return sb.toString();
    }
}
```

PessoaJuridica.java

```
package model.entidades;
```

```
import model.entidades.Pessoa;
```

```
import java.io.Serializable;
```

```
public class PessoaJuridica extends Pessoa implements Serializable {  
    protected String cnpj;
```

```
    @Override
```

```
    public void exibir(){
```

```
        super.exibir();
```

```
        System.out.println("CNPJ: " + cnpj);
```

```
    }
```

```
    public PessoaJuridica(){
```

```
    }
```

```
    public PessoaJuridica(int id, String nome, String cnpj) {
```

```
        super(id, nome);
```

```
        this.cnpj = cnpj;
```

```
    }
```

```
    public String getCnpj() {
```

```
        return cnpj;
```

```
    }
```

```
    public void setCnpj(String cnpj) {
```

```
        this.cnpj = cnpj;
```

```
    }
```

```
    @Override
```

```
    public String toString() {
```

```
        StringBuilder sb = new StringBuilder();
```

```
        sb.append("ID = ").append(id);
```

```
        sb.append(", Nome = ").append(nome);
```

```
        sb.append(", CNPJ = ").append(cnpj);
```

```
        return sb.toString();
```

```
    }
```

```
}
```

PessoaFisicaRepo.java

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

public class PessoaFisicaRepo {

    private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();

    public void inserir(PessoaFisica pessoa) {
        pessoasFisicas.add(pessoa);
    }

    public ArrayList<PessoaFisica> obterTodos() {
        return pessoasFisicas;
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new
        FileOutputStream(nomeArquivo))) {
            out.writeObject(pessoasFisicas);
            System.out.println("Dados de Pessoa Fisica Armazenados.");
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo)))
        {
            pessoasFisicas = (ArrayList<PessoaFisica>) in.readObject();
            System.out.println("Dados de Pessoa Fisica Recuperados:");
            for (PessoaFisica pessoa : pessoasFisicas) {
                pessoa.exibir();
            }
        }
    }

    public void alterar(PessoaFisica pessoa) {
        int index = pessoasFisicas.indexOf(pessoa);
        if (index != -1) {
            pessoasFisicas.set(index, pessoa);
            System.out.println("Pessoa Fisica alterada com sucesso!");
        } else {
            System.out.println("Pessoa Fisica não encontrada!");
        }
    }
}
```

```

public void excluir(int id) {
    PessoaFisica pessoa = obter(id);
    if (pessoa != null) {
        pessoasFisicas.remove(pessoa);
    }
}

public PessoaFisica obter(int id) {
    for (PessoaFisica pessoa : pessoasFisicas) {
        if (pessoa.getId() == id) {
            return pessoa;
        }
    }
    return null;
}
}

```

PessoaJuridicaRepo.java

```

package model.gerenciadores;

import model.entidades.PessoaJuridica;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

public class PessoaJuridicaRepo {

    private ArrayList<PessoaJuridica> pessoasJuridicas = new ArrayList<>();

    public void inserir(PessoaJuridica pessoa) {
        pessoasJuridicas.add(pessoa);
    }

    public ArrayList<PessoaJuridica> obterTodos() {
        return pessoasJuridicas;
    }
}

```

```

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(nomeArquivo))) {
            out.writeObject(pessoasJuridicas);
            System.out.println("Dados de Pessoa Juridica Armazenados.");
        }
    }

    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(nomeArquivo)))
        {
            pessoasJuridicas = (ArrayList<PessoaJuridica>) in.readObject();
            System.out.println("Dados de Pessoa Juridica Recuperados:");
            for (PessoaJuridica pessoa : pessoasJuridicas) {
                pessoa.exibir();
            }
        }
    }

    public void alterar(PessoaJuridica pessoa) {
        int index = pessoasJuridicas.indexOf(pessoa);
        if (index != -1) {
            pessoasJuridicas.set(index, pessoa);
            System.out.println("Pessoa Juridica alterada com sucesso!");
        } else {
            System.out.println("Pessoa Juridica não encontrada!");
        }
    }

    public void excluir(int id) {
        PessoaJuridica pessoa = obter(id);
        if (pessoa != null) {
            pessoasJuridicas.remove(pessoa);
        }
    }

    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pessoa : pessoasJuridicas) {
            if (pessoa.getId() == id) {
                return pessoa;
            }
        }
        return null;
    }
}

```

CadastroPOO.java

```
package cadastropoo;

import java.io.IOException;
import model.entidades.PessoaFisica;
import model.gerenciadores.PessoaFisicaRepo;
import model.entidades.PessoaJuridica;
import model.gerenciadores.PessoaJuridicaRepo;

public class CadastroPOO {
    public static void main(String[] args) {
        try {
            PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
            PessoaFisica pf1 = new PessoaFisica(1, "Ana", "11111111111", 25);
            PessoaFisica pf2 = new PessoaFisica(2, "Carlos", "22222222222", 52);
            repo1.inserir(pf1);
            repo1.inserir(pf2);

            repo1.persistir("pessoasFisicas.dat");

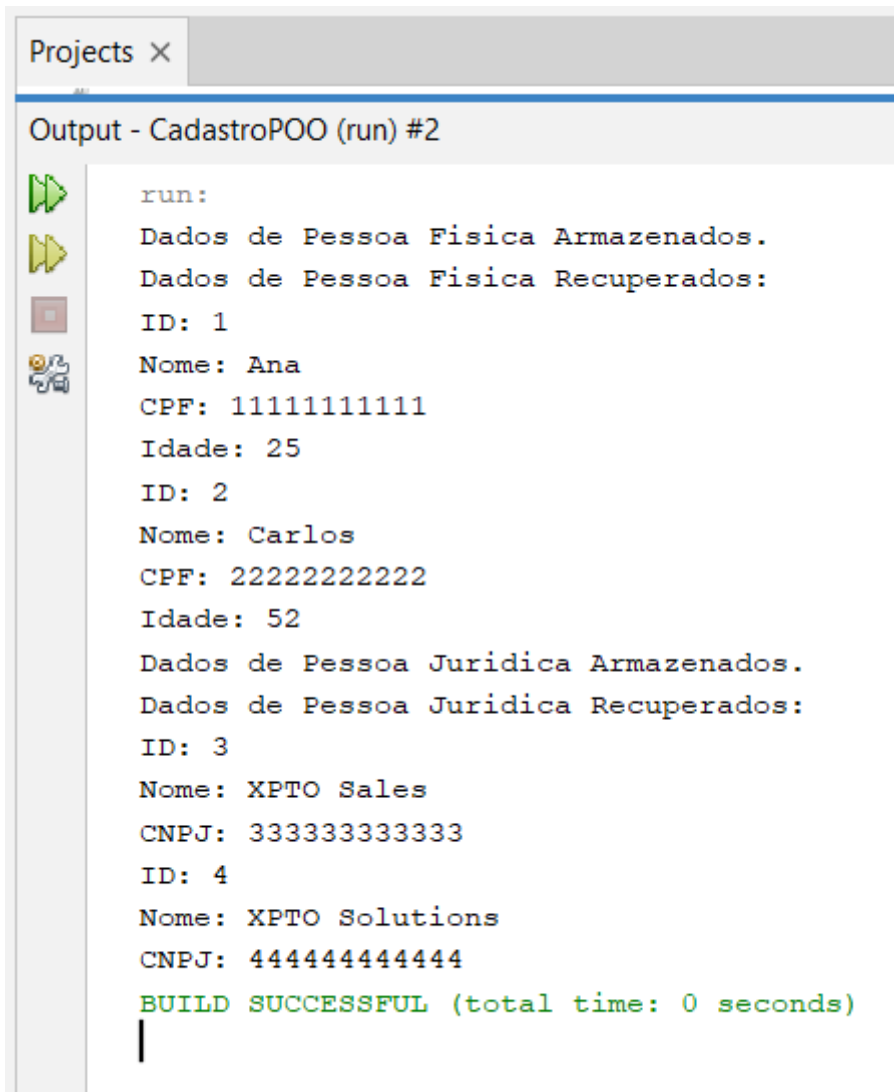
            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
            repo2.recuperar("pessoasFisicas.dat");

            PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
            PessoaJuridica pj1 = new PessoaJuridica(3, "XPTO Sales", "3333333333333");
            PessoaJuridica pj2 = new PessoaJuridica(4, "XPTO Solutions", "4444444444444");
            repo3.inserir(pj1);
            repo3.inserir(pj2);

            repo3.persistir("pessoasJuridicas.dat");

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
            repo4.recuperar("pessoasJuridicas.dat");
        } catch (IOException e) {
            System.err.println("Erro ao acessar o arquivo: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.err.println("Erro ao carregar os dados: " + e.getMessage());
        }
    }
}
```


Resultado da Execução:



```
run:
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados:
ID: 1
Nome: Ana
CPF: 11111111111
Idade: 25
ID: 2
Nome: Carlos
CPF: 22222222222
Idade: 52
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados:
ID: 3
Nome: XPTO Sales
CNPJ: 3333333333333
ID: 4
Nome: XPTO Solutions
CNPJ: 4444444444444
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

Análise e Conclusão:

1. **Quais as vantagens e desvantagens do uso de herança?**
 - **Vantagens:**
 - Reutilização de código.
 - Facilita a manutenção e extensão do sistema.
 - **Desvantagens:**
 - Pode levar a uma hierarquia complexa e difícil de gerenciar.
 - A dependência entre classes pode aumentar.

2. **Por que a interface `Serializable` é necessária ao efetuar persistência em arquivos binários?**

A interface `Serializable` é necessária para permitir que objetos Java sejam convertidos em uma sequência de bytes, o que é essencial para a persistência em arquivos binários. Isso possibilita a gravação e leitura dos objetos de forma eficiente.

3. **Como o paradigma funcional é utilizado pela API Stream no Java?**

A API Stream do Java utiliza conceitos funcionais como operações de alta ordem, imutabilidade e expressões lambda para realizar operações em coleções de maneira mais concisa e expressiva.

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

O padrão de desenvolvimento adotado na persistência de dados em arquivos em Java é a serialização de objetos. Isso significa que os objetos são convertidos em um formato binário que pode ser armazenado em um arquivo e posteriormente recuperado e desserializado para recriar os objetos originais. As classes comumente usadas para serialização de objetos em Java são `ObjectOutputStream` e `ObjectInputStream`.

Repositório GIT:

- O projeto foi armazenado em um repositório no GIT. O endereço do repositório é: <https://github.com/Camilla-Alves/CadastroPOO>

Título da Prática:

Missão Prática | Nível 1 | Mundo 3

2º Procedimento | Criação do Cadastro em Modo Texto

Objetivos da prática:

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

- 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.
- 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 Solicitados:

Obs: Todas as outras classes necessárias já foram incluídas nos códigos solicitados do 1º Procedimento. Aqui incluirei apenas as modificações que foram pedidas no 2º Procedimento para o arquivo principal CadastroPOO, que criei com o nome de CadastroPOOParte2, para que ambos estejam no GitHub.

CadastroPOOParte2.java

```
package cadastrapoo;

import java.io.IOException;
import java.util.List;
import java.util.Scanner;
import model.entidades.PessoaFisica;
import model.entidades.PessoaJuridica;
import model.gerenciadores.PessoaFisicaRepo;
import model.gerenciadores.PessoaJuridicaRepo;

public class CadastroPOOParte2 {
    public static void main(String[] args) {
        PessoaFisicaRepo repoPF = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoPJ = new PessoaJuridicaRepo();

        Scanner sc = new Scanner(System.in);
        boolean executando = true;

        while (executando) {
            System.out.println("=====");
            System.out.println("1 - Incluir Pessoa");
            System.out.println("2 - Alterar Pessoa");
```

```

System.out.println("3 - Excluir Pessoa");
System.out.println("4 - Buscar pelo Id");
System.out.println("5 - Exibir Todos");
System.out.println("6 - Persistir Dados");
System.out.println("7 - Recuperar Dados");
System.out.println("0 - Finalizar Programa");
System.out.println("=====");
int opcao = sc.nextInt();

switch (opcao) {
    case 0:
        System.out.println("Finalizando programa...");
        executando = false;
        break;
    case 1:
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        char tipoPessoaInserir = sc.next().toUpperCase().charAt(0);
        if (tipoPessoaInserir == 'F') {
            System.out.println("Digite o ID da Pessoa: ");
            int id = sc.nextInt();
            sc.nextLine();
            System.out.println("Insira os dados...");
            System.out.println("Nome: ");
            String nome = sc.nextLine();
            System.out.println("CPF: ");
            String cpf = sc.nextLine();
            System.out.println("Idade: ");
            int idade = sc.nextInt();
            repoPF.inserir(new PessoaFisica(id, nome, cpf, idade));

        } else if (tipoPessoaInserir == 'J') {
            System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
            System.out.println("Digite o ID da Pessoa: ");
            int id = sc.nextInt();
            sc.nextLine();
            System.out.println("Insira os dados...");
            System.out.println("Nome: ");
            String nome = sc.nextLine();
            System.out.println("CNPJ: ");
            String cnpj = sc.nextLine();
            repoPJ.inserir(new PessoaJuridica(id, nome, cnpj));
        } else {
            System.out.println("Erro: Escolha Invalida!");
        }
        break;
    case 2:
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        char tipoPessoaAlterar = sc.next().toUpperCase().charAt(0);
        if (tipoPessoaAlterar == 'F') {
            System.out.println("Digite o ID da Pessoa: ");
            int id = sc.nextInt();
            sc.nextLine();

```

```

PessoaFisica pessoa = repoPF.obter(id);
if (pessoa != null) {
    System.out.println("Dados atuais:");
    System.out.println("Nome: " + pessoa.getNome());
    System.out.println("CPF: " + pessoa.getCpf());
    System.out.println("Idade: " + pessoa.getIdade());

    System.out.println("Insira os novos dados:");
    System.out.println("Nome: ");
    String novoNome = sc.nextLine();
    System.out.println("CPF: ");
    String novoCpf = sc.nextLine();
    System.out.println("Idade: ");
    int novaldade = sc.nextInt();

    pessoa.setNome(novoNome);
    pessoa.setCpf(novoCpf);
    pessoa.setIdade(novaldade);

    repoPF.alterar(pessoa);
}
} else if (tipoPessoaAlterar == 'J') {
    System.out.println("Digite o ID da Pessoa: ");
    int id = sc.nextInt();
    sc.nextLine();
    PessoaJuridica pessoa = repoPJ.obter(id);
    if (pessoa != null) {
        System.out.println("Dados atuais:");
        System.out.println("Nome: " + pessoa.getNome());
        System.out.println("CNPJ: " + pessoa.getCnpj());

        System.out.println("Insira os novos dados:");
        System.out.println("Nome: ");
        String novoNome = sc.nextLine();
        System.out.println("CNPJ: ");
        String novoCnpj = sc.nextLine();

        pessoa.setNome(novoNome);
        pessoa.setCnpj(novoCnpj);

        repoPJ.alterar(pessoa);
    }
} else {
    System.out.println("Erro: Escolha Invalida!");
}
break;

case 3:
    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
    char tipoPessoaExclur = sc.next().toUpperCase().charAt(0);

```

```

switch (tipoPessoaExcluir) {
    case 'F':
        System.out.println("Digite o ID da Pessoa: ");
        int id = sc.nextInt();
        PessoaFisica pessoaFisica = repoPF.obter(id);
        if (pessoaFisica != null) {
            repoPF.excluir(id);
            System.out.println("Pessoa fisica excluida com sucesso!");
        } else {
            System.out.println("Pessoa fisica nao encontrada!");
        }
        break;
    case 'J':
        System.out.println("Digite o ID da Pessoa: ");
        id = sc.nextInt();
        PessoaJuridica pessoaJuridica = repoPJ.obter(id);
        if (pessoaJuridica != null) {
            repoPJ.excluir(id);
            System.out.println("Pessoa juridica excluida com sucesso!");
        } else {
            System.out.println("Pessoa juridica nao encontrada!");
        }
        break;
    default:
        System.out.println("Erro: Escolha Invalida!");
}
break;

```

case 4:

```

System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
char tipoPessoaObter = sc.next().toUpperCase().charAt(0);

```

```

switch (tipoPessoaObter) {
    case 'F':
        System.out.println("Digite o ID da pessoa fisica: ");
        int id = sc.nextInt();
        PessoaFisica pessoaFisica = repoPF.obter(id);
        if (pessoaFisica == null) {
            System.out.println("Pessoa fisica nao encontrada!");
        } else {
            System.out.println("Dados da pessoa fisica:");
            System.out.println(pessoaFisica);
        }
        break;
    case 'J':
        System.out.println("Digite o ID da pessoa juridica: ");
        id = sc.nextInt();
        PessoaJuridica pessoaJuridica = repoPJ.obter(id);
        if (pessoaJuridica == null) {
            System.out.println("Pessoa juridica nao encontrada!");
        } else {
            System.out.println("Dados da pessoa juridica:");

```

```

        System.out.println(pessoaJuridica);
    }
    break;
default:
    System.out.println("Erro: Escolha invalida!");
}
break;

```

case 5:

```

System.out.println("F - Pessoas Fisicas | J - Pessoas Juridicas");
char tipoPessoaObterTodos = sc.next().toUpperCase().charAt(0);

if (tipoPessoaObterTodos == 'F') {
    System.out.println("Pessoas Fisicas:");
    List<PessoaFisica> pessoasFisicas = repoPF.obterTodos();

    if (pessoasFisicas.isEmpty()) {
        System.out.println("Nao existem pessoas fisicas cadastradas no sistema.");
    } else {
        for (PessoaFisica pf : pessoasFisicas) {
            System.out.println(pf.toString());
        }
    }
} else if (tipoPessoaObterTodos == 'J') {
    System.out.println("Pessoas Juridicas:");
    List<PessoaJuridica> pessoasJuridicas = repoPJ.obterTodos();

    if (pessoasJuridicas.isEmpty()) {
        System.out.println("Nao existem pessoas juridicas cadastradas no sistema.");
    } else {
        for (PessoaJuridica pj : pessoasJuridicas) {
            System.out.println(pj.toString());
        }
    }
} else {
    System.out.println("Erro: Escolha invalida!");
}
break;

```

case 6:

```

System.out.println("Salvar Dados");
sc.nextLine();
System.out.print("Informe o prefixo dos arquivos: ");
String prefixo = sc.nextLine();

try {
    repoPF.persistir(prefixo + ".fisica.bin");
    repoPJ.persistir(prefixo + ".juridica.bin");
} catch (IOException e) {
    System.err.println("Erro ao salvar os dados: " + e.getMessage());
}
break;

```

```

        case 7:
            System.out.println("Recuperar Dados");
            sc.nextLine();
            System.out.print("Informe o prefixo dos arquivos: ");
            String prefixoRec = sc.nextLine();

            try {
                repoPF.recuperar(prefixoRec + ".fisica.bin");
                repoPJ.recuperar(prefixoRec + ".juridica.bin");
            } catch (IOException | ClassNotFoundException e) {
                System.err.println("Erro ao recuperar os dados: " + e.getMessage());
            }
            break;

        default:
            System.out.println("Erro: Escolha invalida!");
    }
}

sc.close();
}
}

```

Resultados da Execução:

Na primeira imagem, foram testados os métodos **Incluir** e **Alterar**.

Na segunda imagem, foram testados os métodos **Buscar pelo Id**, **Excluir** e **Finalizar Programa**.

Na terceira imagem, foi testado o método **Exibir Todos**.

Na quarta imagem, foram testados os métodos **Persistir Dados** e **Recuperar Dados**.

Output - **CadastroPOO (run) #23**

```

4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
1
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o ID da Pessoa:
190
Insira os dados...
Nome:
Camilla
CPF:
10000000
Idade:
24
=====
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 - Pessoa Fisica | J - Pessoa Juridica
f
Digite o ID da Pessoa:
190
Dados atuais:
Nome: Camilla
CPF: 10000000
Idade: 24
Insira os novos dados:
Nome:
Camilla
CPF:
100000
Idade:
25
Pessoa Fisica alterada com sucesso!
=====
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
=====
|
```

Primeira imagem

```

=====
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
=====

4
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o ID da pessoa fisica:
190
Dados da pessoa fisica:
ID = 190, Nome = Camilla, CPF = 100000, Idade = 25
=====

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
=====

3
F - Pessoa Fisica | J - Pessoa Juridica
f
Digite o ID da Pessoa:
190
Pessoa fisica excluida com sucesso!
=====

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
=====

0
Finalizando programa...
BUILD SUCCESSFUL (total time: 2 minutes 49 seconds)


```

Segunda imagem

```

=====
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
=====
5
F - Pessoas Fisicas | J - Pessoas Juridicas
f
Pessoas Fisicas:
ID = 100, Nome = Alice, CPF = 10000000, Idade = 40
ID = 200, Nome = Bianca, CPF = 20000000, Idade = 25
=====
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
=====
5
F - Pessoas Fisicas | J - Pessoas Juridicas
j
Pessoas Juridicas:
ID = 300, Nome = Carlos, CNPJ = 30000000
ID = 400, Nome = Daniel, CNPJ = 40000000
=====
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 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
6
Salvar Dados
Informe o prefixo dos arquivos: teste
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Juridica Armazenados.
=====
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
=====
7
Recuperar Dados
Informe o prefixo dos arquivos: teste
Dados de Pessoa Fisica Recuperados
ID: 100
Nome: Alice
CPF: 10000000
Idade: 40
ID: 200
Nome: Bianca
CPF: 20000000
Idade: 25
Dados de Pessoa Juridica Recuperados:
ID: 300
Nome: Carlos
CNPJ: 30000000
ID: 400
Nome: Daniel
CNPJ: 40000000
=====
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
=====
.
```

Quarta imagem

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 permite a leitura de diferentes tipos de dados diretamente do teclado, facilitando a interação com o usuário.

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

O uso de classes de repositório simplifica a organização do código, separando as operações de acesso aos dados das operações de negócios. Isso torna o código mais limpo e fácil de entender.

Repositório GIT:

- O projeto foi armazenado em um repositório no GIT. O endereço do repositório é: <https://github.com/Camilla-Alves/CadastroPOO>