



UNIVERSIDADE ESTÁCIO DE SÁ

FULLSTACK

Nível 1: Iniciando o Caminho Pelo Java

MAIARA ACCACIO MACHADO

202204268183 | Turma 9001

RIO DE JANEIRO – RJ

2023

SUMÁRIO

SUMÁRIO	1
1. INTRODUÇÃO	2
1.1 OBJETIVOS DA PRÁTICA.....	2
2. PROCEDIMENTO 1.....	3
2.1 Classe Pessoa.....	3
2.2 Classe PessoaFisica.....	3
2.3 Classe PessoaJuridica	4
2.4 Classe PessoaFisicaRepo	5
2.5 Classe PessoaJuridicaRepo	7
2.6 Classe Principal.....	10
2.7 Saída da execução	11
2.8 Análise e Conclusão.....	12
3. PROCEDIMENTO 2.....	13
3.1 Classe Principal.....	13
3.2 Classe RepoFisicaRepo	17
3.3 Classe PessoaJuridicaRepo	19
3.4 Execução.....	22
3.4.1 Incluir Pessoa.....	22
3.4.2 Alterar pessoa	22
3.4.3 Excluir Pessoa	23
3.4.4 Buscar pelo ID.....	23
3.4.5 Exibir todos.....	23

1. INTRODUÇÃO

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

1.1 OBJETIVOS 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.
- 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.

2. PROCEDIMENTO 1

O procedimento consiste na Criação das Entidades e Sistema de Persistência

2.1 Classe Pessoa

```
package model;

import java.io.Serializable;

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

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

    public void exibir(){
        System.out.println("Id: "+this.getId());
        System.out.println("Nome: "+this.getNome());
    }

    public int getId() {
        return id;
    }

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

    public String getNome() {
        return nome;
    }

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

2.2 Classe PessoaFisica

```
package model;

import java.io.Serializable;
```

```

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

    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;
    }

    public void exibir(){
        super.exibir();
        System.out.println("CPF: "+this.getCpf());
        System.out.println("Idade: "+this.getIdade());
    }
}

```

2.3 Classe PessoaJuridica

```

package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements
Serializable{
    private String cnpj;

    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;
    }

    public void exhibir(){
        super.exibir();
        System.out.println("CNPJ: "+this.getCnpj());
    }
}

```

2.4 Classe PessoaFisicaRepo

```

package model;

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

public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoasFisicas;

    public PessoaFisicaRepo() {
        pessoasFisicas = new ArrayList<>();
    }

    //TODO: método inserir
    public void inserir(PessoaFisica pessoaFisica) {
        pessoasFisicas.add(pessoaFisica);
        System.out.println("Cadastro realizado com sucesso.");
    }

    //TODO: método alterar
    public void alterar(PessoaFisica pessoaFisica) {
        int id = pessoaFisica.getId();
        excluir(id);
        pessoasFisicas.add(pessoaFisica);
        System.out.println("Cadastro atualizado com

```

```

sucesso.");
    }

    //TODO: método excluir
    public void excluir(int id) {
        PessoaFisica pfexclusao = null;
        for (PessoaFisica pessoafisica : pessoasFisicas) {
            if (pessoafisica.getId() == id) {
                pfexclusao = pessoafisica;
                break;
            }
        }
        if (pfexclusao != null) {
            pessoasFisicas.remove(pfexclusao);
            System.out.println("Cadastro excluído com
sucesso!");
        } else {
            System.out.println("Não foi possível excluir o
cadastro. Id não encontrado.");
        }
    }

    //TODO: método obter
    public void obter(int id) {
        boolean encontrado = false;
        for (PessoaFisica pessoaFisica : pessoasFisicas
) {
            if (pessoaFisica.getId() == id) {
                pessoaFisica.exibir();
                encontrado = true;

                break;
            }
        }
        if (!encontrado) {
            System.out.println("Cadastro não encontrado. Por
favor, verifique o id.");
        }
    }

    //TODO: método obterTodos
    public void obterTodos() {
        System.out.println("----- Lista de Pessoas Físicas
cadastradas -----");
        for (PessoaFisica pessoaFisica : pessoasFisicas
) {
            pessoaFisica.exibir();
            System.out.println("-----");
        }
    }

```

```

    //TODO: método persistir
    public void persistir(String arquivo) throws Exception {
        try (FileOutputStream saida = new
FileOutputStream(arquivo);
            ObjectOutputStream objeto = new
ObjectOutputStream(saida)) {

            objeto.writeObject(pessoasFisicas);
            System.out.println("Dados de Pessoa Física
Armazenados.");
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    public void recuperar(String arquivo) throws Exception {
        try (FileInputStream entrada = new
FileInputStream(arquivo);
            ObjectInputStream objeto = new
ObjectInputStream(entrada)) {

            pessoasFisicas = (ArrayList<PessoaFisica>)
objeto.readObject();
            System.out.println("Dados de Pessoa Física
recuperados.");
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}

```

2.5 Classe PessoaJuridicaRepo

```

package model;

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

public class PessoaJuridicaRepo {

    private ArrayList<PessoaJuridica> pessoasJuridicas;

    public PessoaJuridicaRepo() {

```



```

        pessoasJuridicas = new ArrayList<>();
    }

    //TODO: método inserir
    public void inserir(PessoaJuridica pessoaJuridica) {
        pessoasJuridicas.add(pessoaJuridica);
        System.out.println("Cadastro realizado com sucesso.");
    }

    //TODO: método alterar
    public void alterar(PessoaJuridica pessoaJuridica) {
        int id = pessoaJuridica.getId();
        excluir(id);
        pessoasJuridicas.add(pessoaJuridica);
        System.out.println("Cadastro atualizado com
sucesso.");
    }

    //TODO: método excluir
    public void excluir(int id) {
        PessoaJuridica pjexclusao = null;
        for (PessoaJuridica pessoafisica : pessoasJuridicas) {
            if (pessoafisica.getId() == id) {
                pjexclusao = pessoafisica;
                break;
            }
        }
        if (pjexclusao != null) {
            pessoasJuridicas.remove(pjexclusao);
            System.out.println("Cadastro excluído com
sucesso!");
        } else {
            System.out.println("Não foi possível excluir o
cadastro. Id não encontrado.");
        }
    }

    //TODO: método obter
    public void obter(int id) {
        boolean encontrado = false;
        for (PessoaJuridica pessoaJuridica : pessoasJuridicas
) {
            if (pessoaJuridica.getId() == id) {
                pessoaJuridica.exibir();
                encontrado = true;

                break;
            }
        }
    }

```

```

        if (!encontrado) {
            System.out.println("Cadastro não encontrado. Por favor, verifique o id.");
        }
    }

    //TODO: método obterTodos
    public void obterTodos() {
        System.out.println("----- Lista de Pessoas jurídicas cadastradas -----");
        for (PessoaJuridica pessoaFisica : pessoasJuridicas) {
            pessoaFisica.exibir();
            System.out.println("-----");
        }
    }

    //TODO: método persistir
    public void persistir(String arquivo) throws Exception {
        try (FileOutputStream saida = new
FileOutputStream(arquivo);
            ObjectOutputStream objeto = new
ObjectOutputStream(saida)) {

            objeto.writeObject(pessoasJuridicas);
            System.out.println("Dados de Pessoa Jurídica Armazenados.");
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    public void recuperar(String arquivo) throws Exception {
        try (FileInputStream entrada = new
FileInputStream(arquivo);
            ObjectInputStream objeto = new
ObjectInputStream(entrada)) {

            pessoasJuridicas = (ArrayList<PessoaJuridica>)
objeto.readObject();
            System.out.println("Dados de Pessoa Jurídica recuperados.");

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}

```

2.6 Classe Principal

```
package model;

import java.util.List;

public class Principal {
    public static void main(String[] args) throws Exception {
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

        PessoaFisica jonas = new PessoaFisica(1, "Jonas",
"121212115811", 51);
        PessoaFisica maria = new PessoaFisica(2, "Maria",
"121212112112", 25);

        repo1.inserir(maria);
        repo1.inserir(jonas);

        repo1.obterTodos();
        repo1.persistir("pessoasFisicas");
        repo2.recuperar("pessoasFisicas");
        repo2.obterTodos();

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

        PessoaJuridica gasbras = new PessoaJuridica(10,
"Gasbras", "4567891211464846");
        PessoaJuridica arno = new PessoaJuridica(11, "Arno",
"4578942311464846");
        repo3.inserir(gasbras);
        repo3.inserir(arno);

        repo3.persistir("pessoasJuridicas");

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

        repo4.recuperar("pessoasJuridicas");
        repo4.obterTodos();

    }
}
```

2.7 Saída da execução

```

Cadastro realizado com sucesso.
Cadastro realizado com sucesso.
----- Lista de Pessoas Físicas cadastradas -----
Id: 2
Nome: Maria
CPF: 121212112112
Idade: 25
-----
Id: 1
Nome: Jonas
CPF: 121212115811
Idade: 51
-----
Dados de Pessoa Física Armazenados.
Dados de Pessoa Física recuperados.
----- Lista de Pessoas Físicas cadastradas -----
Id: 2
Nome: Maria
CPF: 121212112112
Idade: 25
-----
Id: 1
Nome: Jonas
CPF: 121212115811
Idade: 51
-----
Cadastro realizado com sucesso.
Cadastro realizado com sucesso.
Dados de Pessoa Jurídica Armazenados.
Dados de Pessoa Jurídica recuperados.
----- Lista de Pessoas jurídicas cadastradas -----
Id: 10
Nome: Gasbras
CNPJ: 4567891211464846
-----
Id: 11
Nome: Arno
CNPJ: 4578942311464846
-----

```

2.8 Análise e Conclusão

Quais as vantagens e desvantagens do uso de herança?

Como vantagem temos a possibilidade de reutilização de códigos, evitando duplicidade de código e desorganização, o que facilita a manutenção. Além disso, a herança oferece flexibilidade por permitir o uso de novos atributos e métodos nas classes filhas sem interferência na estrutura da classe Pai. Outra vantagem é o polimorfismo que é possibilitado pela herança.

Como desvantagens temos o acoplamento entre a classe Pai e seus filhos, o que pode gerar impactos indesejados quando uma alteração é feita na classe Pai. Outra desvantagem é a alta complexidade que pode ser ocasionada por uma grande quantidade de heranças.

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

A interface Serializable indica que uma classe pode ser convertida em uma sequência de bytes (serialização) para ser salva no arquivo e recuperada posteriormente para seu formato original no processo de deserialização.

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

Funções de ordem superior, que são funções que podem receber outras funções como argumento ou retorná-las. A API stream também utiliza operações comuns do paradigma funcional como map (transforma cada elemento da coleção em um novo valor de acordo com a definição da função), filter (filtra os elementos de acordo com a condição estabelecida), reduce (reduz a coleção a um único valor aplicando uma função acumuladora) e a lazy evaluation que adia as operações até que o resultado seja necessário.

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

O padrão mais comum é o DAO (Data Access Object) que separa a lógica de acesso a dados da lógica de negócios da aplicação.

3. PROCEDIMENTO 2

O procedimento consiste na Criação do cadastro em modo texto.

3.1 Classe Principal

```
package model;

import java.util.Scanner;

public class Principal {
    public static void main(String[] args) throws Exception {

        // PROCEDIMENTO 2
        Scanner leitura = new Scanner(System.in);
        String menu = "===== " +
            "\n1 - Incluir Pessoa" +
            "\n2 - Alterar Pessoa" +
            "\n3 - Excluir Pessoa" +
            "\n4 - Buscar pelo ID" +
            "\n5 - Exibir todos" +
            "\n6 - Persistir dados" +
            "\n7 - Recuperar dados" +
            "\n0 - Finalizar programa" +
            "\n===== \n";

        boolean ligado = true;

        PessoaJuridicaRepo repoJuridica = new
PessoaJuridicaRepo();
        PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();

        do {
            System.out.println(menu);

            String opcao = leitura.nextLine();

            switch (opcao) {
```

```

        case "1":
            System.out.println("F - Pessoa Física | J
- Pessoa Jurídica");
            String tipoPessoa =
leitura.nextLine().toUpperCase();

            System.out.println("Digite o id da
pessoa:");

            int id =
Integer.parseInt(leitura.nextLine());

            if (tipoPessoa.equals("F")) {

                repoFisica.incluir(id);
                System.out.println("Cadastro realizado
com sucesso!");

            } else if (tipoPessoa.equals("J")) {
                repoJuridica.incluir(id);
                System.out.println("Cadastro realizado
com sucesso!");
            }

            break;

        case "2":
            System.out.println("F - Pessoa Física | J
- Pessoa Jurídica");
            String tipoPessoaAlterar =
leitura.nextLine().toUpperCase();

            System.out.println("Digite o id da
pessoa:");

            int idAlterar =
Integer.parseInt(leitura.nextLine());

            if (tipoPessoaAlterar.equals("F")) {
                repoFisica.alterar(idAlterar);
                System.out.println("Cadastro
atualizado com sucesso!");
            } else if (tipoPessoaAlterar.equals("J"))
{
                repoJuridica.alterar(idAlterar);
                System.out.println("Cadastro
atualizado com sucesso!");
            }

            break;

        case "3":

```

```

        System.out.println("F - Pessoa Física | J
- Pessoa Jurídica");
        String tipoPessoaExcluir =
leitura.nextLine().toUpperCase();

        System.out.println("Digite o id da
pessoa:");

        int idExcluir =
Integer.parseInt(leitura.nextLine());

        if (tipoPessoaExcluir.equals("F")) {
            repoFisica.excluir(idExcluir);
            System.out.println("Cadastro excluído
com sucesso");

        } else if (tipoPessoaExcluir.equals("J"))
{
            repoJuridica.excluir(idExcluir);
            System.out.println("Cadastro excluído
com sucesso");

        }

        break;

    case "4":
        System.out.println("F - Pessoa Física | J
- Pessoa Jurídica");
        String tipoPessoaObter =
leitura.nextLine().toUpperCase();

        System.out.println("Digite o id da
pessoa:");

        int idObter =
Integer.parseInt(leitura.nextLine());

        if (tipoPessoaObter.equals("F")) {
            repoFisica.obter(idObter);

        } else if (tipoPessoaObter.equals("J")) {
            repoJuridica.obter(idObter);

        }

        break;

    case "5":
        System.out.println("F - Pessoa Física | J
- Pessoa Jurídica");
        String tipoPessoaObterTodos =
leitura.nextLine().toUpperCase();

        if (tipoPessoaObterTodos.equals("F")) {

```



```

        repoFisica.obterTodos();

        } else if
(tipoPessoaObterTodos.equals("J")) {
            repoJuridica.obterTodos();
        }

        break;
    case "6":
        System.out.println("F - Pessoa Física | J
- Pessoa Jurídica");
        String tipoPessoaPersistir =
leitura.nextLine().toUpperCase();

        if (tipoPessoaPersistir.equals("F")) {
repoFisica.persistir("pessoas_fisicas");

        } else if
(tipoPessoaPersistir.equals("J")) {
repoJuridica.persistir("pessoas_juridicas");
        }break;

    case "7":
        System.out.println("F - Pessoa Física | J
- Pessoa Jurídica");
        String tipoPessoaRecuperar =
leitura.nextLine().toUpperCase();

        if (tipoPessoaRecuperar.equals("F")) {
repoFisica.recuperar("pessoas_fisicas");

        } else if
(tipoPessoaRecuperar.equals("J")) {
repoJuridica.recuperar("pessoas_juridicas");
        }
        break;

    case "0":
        ligado = false;
        break;

    default:
        System.out.println("Insira um opção
válida.");
}

```

```

        } while (ligado);
        leitura.close();
    }
}

```

3.2 Classe RepoFisicaRepo

```

package model;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.Scanner;

public class PessoaFisicaRepo {
    private ArrayList<PessoaFisica> pessoasFisicas;

    public PessoaFisicaRepo() {
        pessoasFisicas = new ArrayList<>();
    }

    Scanner leitura = new Scanner(System.in);
    public void incluir(int id) {

        System.out.println("Digite o nome da pessoa: ");
        String nome = leitura.nextLine();

        System.out.println("Digite o cpf da pessoa: ");
        String cpf = leitura.nextLine();

        System.out.println("Digite a idade da pessoa: ");
        int idade = Integer.parseInt(leitura.nextLine());

        PessoaFisica novaPessoa = new PessoaFisica(id, nome,
        cpf, idade);
        pessoasFisicas.add(novaPessoa);
    }

    public void alterar(int id) {

```

```

        obter(id);
        excluir(id);
        incluir(id);
    }

    public void excluir(int id) {
        PessoaFisica pfexclusao = null;
        for (PessoaFisica pessoafisica : pessoasFisicas) {
            if (pessoafisica.getId() == id) {
                pfexclusao = pessoafisica;
                break;
            }
        }
        if (pfexclusao != null) {
            pessoasFisicas.remove(pfexclusao);
        } else {
            System.out.println("Não foi possível excluir o cadastro. Id não encontrado.");
        }
    }

    public void obter(int id) {
        boolean encontrado = false;
        for (PessoaFisica pessoaFisica : pessoasFisicas) {
            if (pessoaFisica.getId() == id) {
                pessoaFisica.exibir();
                encontrado = true;

                break;
            }
        }
        if (!encontrado) {
            System.out.println("Cadastro não encontrado. Por favor, verifique o id.");
        }
    }

    public void obterTodos() {
        System.out.println("----- Lista de Pessoas Físicas cadastradas -----");
        for (PessoaFisica pessoaFisica : pessoasFisicas) {
            pessoaFisica.exibir();
            System.out.println("-----");
        }
    }

```

```

        public void persistir(String prefixo) throws Exception {
            try (FileOutputStream saida = new
FileOutputStream(prefixo+".fisica.bin");
                ObjectOutputStream objeto = new
ObjectOutputStream(saida)) {

                objeto.writeObject(pessoasFisicas);
                System.out.println("Dados de Pessoa Física
Armazenados.");
            } catch (Exception e) {
                System.out.println("erro "+ e);
            }
        }

        public void recuperar(String prefixo) throws Exception {
            try (FileInputStream entrada = new
FileInputStream(prefixo+".fisica.bin");
                ObjectInputStream objeto = new
ObjectInputStream(entrada)) {

                pessoasFisicas = (ArrayList<PessoaFisica>)
objeto.readObject();
                System.out.println("Dados de Pessoa Física
recuperados.");
            } catch (Exception e) {
                System.out.println("Erro "+ e);
            }
        }
    }
}

```

3.3 Classe PessoaJuridicaRepo

```

package model;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.Scanner;

public class PessoaJuridicaRepo {

    private ArrayList<PessoaJuridica> pessoasJuridicas;

```

```

public PessoaJuridicaRepo() {
    pessoasJuridicas = new ArrayList<>();
}
Scanner leitura = new Scanner(System.in);

public void incluir(int id){

    System.out.println("Digite o nome da pessoa: ");
    String nome = leitura.nextLine();

    System.out.println("Digite o cnpj da pessoa: ");
    String cnpj = leitura.nextLine();

    PessoaJuridica novaPessoa = new PessoaJuridica(id,
nome, cnpj);
    pessoasJuridicas.add(novaPessoa);

}

public void alterar(int id) {

    obter(id);
    excluir(id);
    incluir(id);
}

//TODO: método excluir
public void excluir(int id) {

    PessoaJuridica pjexclusao = null;
    for (PessoaJuridica pessoafisica : pessoasJuridicas) {
        if (pessoafisica.getId() == id) {
            pjexclusao = pessoafisica;
            break;
        }
    }
    if (pjexclusao != null) {
        pessoasJuridicas.remove(pjexclusao);
        System.out.println("Cadastro excluído com
sucesso!");
    } else {
        System.out.println("Não foi possível excluir o
cadastro. Id não encontrado.");
    }
}

//TODO: método obter
public void obter(int id) {

    boolean encontrado = false;

```

```

        for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
            if (pessoaJuridica.getId() == id) {
                pessoaJuridica.exibir();
                encontrado = true;

                break;
            }
        }
        if (!encontrado) {
            System.out.println("Cadastro não encontrado. Por favor, verifique o id.");
        }
    }

    //TODO: método obterTodos
    public void obterTodos() {
        System.out.println("----- Lista de Pessoas jurídicas cadastradas -----");
        for (PessoaJuridica pessoaFisica : pessoasJuridicas) {
            pessoaFisica.exibir();
            System.out.println("-----");
        }
    }

    //TODO: método persistir
    public void persistir(String prefixo) throws Exception {
        try (FileOutputStream saida = new
FileOutputStream(prefixo+".juridica.bin");
            ObjectOutputStream objeto = new
ObjectOutputStream(saida)) {

            objeto.writeObject(pessoasJuridicas);
            System.out.println("Dados de Pessoa Jurídica Armazenados.");
        } catch (Exception e) {
            System.out.println("Erro " +e);
        }
    }

    public void recuperar(String prefixo) throws Exception {
        try (FileInputStream entrada = new
FileInputStream(prefixo+".juridica.bin");
            ObjectInputStream objeto = new
ObjectInputStream(entrada)) {

            pessoasJuridicas = (ArrayList<PessoaJuridica>)
objeto.readObject();
            System.out.println("Dados de Pessoa Jurídica

```

```
recuperados.");

        } catch (Exception e) {
            System.out.println("Erro "+e);
        }
    }
}
```

3.4 Execução

3.4.1 Incluir Pessoa

<pre>===== 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 ===== F - Pessoa Física J - Pessoa Jurídica F Digite o id da pessoa: 1 Digite o nome da pessoa: Marieta Machado Digite o cpf da pessoa: 123456789 Digite a idade da pessoa: 86 Cadastro realizado com sucesso! =====</pre>	<pre>===== 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 ===== F - Pessoa Física J - Pessoa Jurídica F Digite o id da pessoa: 10 Digite o nome da pessoa: Brastemp Digite o cnpj da pessoa: 456123789123 Cadastro realizado com sucesso! =====</pre>
---	--

3.4.2 Alterar pessoa

<pre>===== F - Pessoa Física J - Pessoa Jurídica F Digite o id da pessoa: 1 Id: 1 Nome: Marieta Machado CPF: 123456789 Idade: 86 Digite o nome da pessoa: Maria Machado Digite o cpf da pessoa: 123456789 Digite a idade da pessoa: 87 Cadastro atualizado com sucesso! =====</pre>	<pre>2 F - Pessoa Física J - Pessoa Jurídica F Digite o id da pessoa: 10 Id: 10 Nome: Brastemp CNPJ: 456123789123 Cadastro excluído com sucesso! Digite o nome da pessoa: Brastemp-New Digite o cnpj da pessoa: 456123789123 Cadastro atualizado com sucesso! =====</pre>
---	---

3.4.3 Excluir Pessoa

```

3
F - Pessoa Física | J - Pessoa Jurídica
f
Digite o id da pessoa:
1
Cadastro excluído com sucesso
=====

=====
3
F - Pessoa Física | J - Pessoa Jurídica
j
Digite o id da pessoa:
10
Cadastro excluído com sucesso!

```

3.4.4 Buscar pelo ID

```

4
F - Pessoa Física | J - Pessoa Jurídica
f
Digite o id da pessoa:
3
Id: 3
Nome: Dilma Machado
CPF: 456789123
Idade: 71
=====

-----
4
F - Pessoa Física | J - Pessoa Jurídica
j
Digite o id da pessoa:
12
Id: 12
Nome: Arno Ultimate
CNPJ: 951841357456
=====

```

3.4.5 Exibir todos

```

5
F - Pessoa Física | J - Pessoa Jurídica
f
----- Lista de Pessoas Físicas cadastradas -----
Id: 1
Nome: João Machado
CPF: 123456789
Idade: 68
-----
Id: 2
Nome: Marieta Machado
CPF: 789456132
Idade: 89
-----
Id: 3
Nome: Dilma Machado
CPF: 456789123
Idade: 71
-----

5
F - Pessoa Física | J - Pessoa Jurídica
j
----- Lista de Pessoas jurídicas cadastradas -----
Id: 10
Nome: Brastemp Global
CNPJ: 789456123456
-----
Id: 11
Nome: Sti
CNPJ: 789159753123
-----
Id: 12
Nome: Arno Ultimate
CNPJ: 951841357456
-----

```


3.4.6 Persistir dados

```
=====
6
F - Pessoa Física | J - Pessoa Juridica
f
Dados de Pessoa Física Armazenados.
=====
```

```
=====
6
F - Pessoa Física | J - Pessoa Juridica
j
Dados de Pessoa Juridica Armazenados.
=====
```

```
> out
> src
> untitled
  .gitignore
  CadastroPOO.iml
  pessoas.fisica.bin
  pessoas.juridica.bin
  External Libraries
```

3.4.7 Recuperar dados

```
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:
=====
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
=====
f
F - Pessoa Física | J - Pessoa Juridica
f
Dados de Pessoa Física recuperados.
=====
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
=====
f
F - Pessoa Física | J - Pessoa Juridica
f
----- Lista de Pessoas Físicas cadastradas -----
Id: 1
Nome: Maria
CPF: 123456789
Idade: 25
-----
Id: 2
Nome: Joao
CPF: 7894561336
Idade: 56
-----

"C:\Program Files\Java\jdk-17\bin\java.exe" "-jav
=====
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
=====
j
F - Pessoa Física | J - Pessoa Juridica
j
Dados de Pessoa Juridica recuperados.
=====
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
=====
j
F - Pessoa Física | J - Pessoa Juridica
j
----- Lista de Pessoas juridicas cadastradas -----
Id: 10
Nome: Brastemp Global
CNPJ: 789456123159
-----
Id: 11
Nome: Sony
CNPJ: 852741951489
-----
```

3.4.8 Finalizar a execução

```

=====
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
Process finished with exit code 0

```

3.5 Análise e conclusão

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

Elementos estáticos são aqueles que pertencem a própria classe. São compartilhados por todas as instâncias da classe e existem independentemente de um objeto.

A classe main adota esse modificador para que seja chamada diretamente sem necessidade de instanciação quando o programa é executado.

Para que serve a classe Scanner?

Ela permite a leitura de dados a partir de diversos tipos de entrada, como console e arquivos. Por meio dessa classe é possível interagir com o usuário no console.

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

A divisão do código em classes de repositório contribuiu para a organização e, consequentemente, facilitou a manutenção e descoberta de erros.