



Estácio

Missão Prática Nível 1 – Mundo 3

Fernanda Canto P. da Costa - 202208379788

Campus de Ipanema

Iniciando o caminho pelo Java – 22.3 – 3º semestre

Github: 1 procedimento

<https://github.com/nandacpc/MissaoNv1-M3/tree/master>

2 procedimento

<https://github.com/nandacpc/MissaoNv1-M3/tree/main>

Objetivo da Prática

- Implementar persistência com base em JPA.
- Implementar regras de negócio na plataforma JEE, através de EJBs.
- Implementar sistema cadastral Web com base em Servlets e JSPs.
- Utilizar a biblioteca Bootstrap para melhoria do design.

1º Procedimento | Criação das Entidades e Sistema de Persistência

Arquivo CadastroPOO.java:

```
package cadastrapoo;
import model.PessoaFisica;
import model.PessoaJuridica;
import model.PessoaFisicaRepo;
import model.PessoaJuridicaRepo;
import java.io.IOException;

public class CadastroPOO {

    public static void main(String[] args) {
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();
        PessoaFisica pessoa1 = new PessoaFisica(1, "Ana", "11111111111",
25);
```

```

        PessoaFisica pessoa2 = new PessoaFisica(2, "Carlos",
"22222222222", 52);
        repo1.inserir(pessoa1);
        repo1.inserir(pessoa2);

        try {
            repo1.persistir("pessoas_fisicas.dat");
            System.out.println("Pessoas Fisicas Armazenadas.");

            PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

            repo2.recuperar("pessoas_fisicas.dat");

            System.out.println("Pessoas Fisicas Recuperadas.");
            for (PessoaFisica pessoa : repo2.obterTodos()) {
                pessoa.exibir();
            }
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }

        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();
        PessoaJuridica pessoaJuridica1 = new PessoaJuridica(3, "XPTO
Sales", "33333333333333");
        PessoaJuridica pessoaJuridica2 = new PessoaJuridica(4, "XPTO
Solutions", "44444444444444");
        repo3.inserir(pessoaJuridica1);
        repo3.inserir(pessoaJuridica2);

        try {
            repo3.persistir("pessoas_juridicas.dat");
            System.out.println("Pessoas Juridicas Armazenadas.");

            PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

            repo4.recuperar("pessoas_juridicas.dat");

            System.out.println("Pessoas Juridicas Recuperadas.");
            for (PessoaJuridica pessoaJuridica : repo4.obterTodos()) {
                pessoaJuridica.exibir();
            }
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}

```

Arquivo Pessoa.java:

```
package model;

import java.io.Serializable;

public class Pessoa implements Serializable {
    private int id;
    private String 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;
    }
    public void exibir() {
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
    }
}
```

Arquivo PessoaFisica.java:

```
package model;

import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    private String cpf;
    private int 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 void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
    }
}
```

Arquivo PessoaJuridica.java:

```
package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    private String 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 void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }
}
```

Arquivo PessoaFisicaRepo.java:

```
package model;

import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaRepo {
    private List<PessoaFisica> pessoasFisicas;

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

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

    public void alterar(PessoaFisica pessoa) {
    }

    public void excluir(int id) {
    }

    public PessoaFisica obter(int id) {
        return null;
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new
        FileOutputStream(nomeArquivo))) {
            out.writeObject(pessoasFisicas);
        }
    }

    public void recuperar(String nomeArquivo) throws IOException,
    ClassNotFoundException {
        try (ObjectInputStream in = new ObjectInputStream(new
        FileInputStream(nomeArquivo))) {
            pessoasFisicas = (List<PessoaFisica>) in.readObject();
        }
    }
}
```

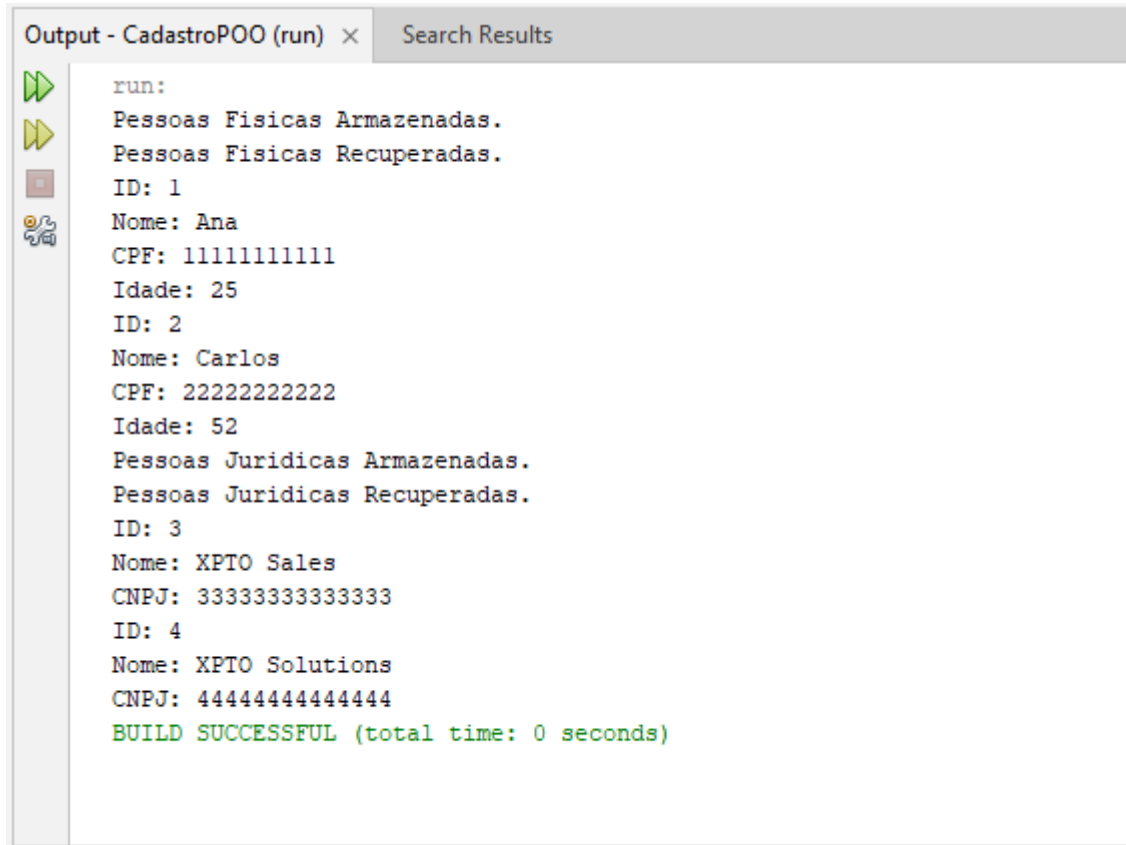
```
}  
}
```

Arquivo PessoaJuridicaRepo.java:

```
package model;  
  
import java.io.*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class PessoaJuridicaRepo {  
    private List<PessoaJuridica> pessoasJuridicas;  
  
    public PessoaJuridicaRepo() {  
        pessoasJuridicas = new ArrayList<>();  
    }  
    public void inserir(PessoaJuridica pessoa) {  
        pessoasJuridicas.add(pessoa);  
    }  
    public void alterar(PessoaJuridica pessoa) {  
    }  
    public void excluir(int id) {  
    }  
    public PessoaJuridica obter(int id) {  
        return null;  
    }  
    public List<PessoaJuridica> obterTodos() {  
        return pessoasJuridicas;  
    }  
    public void persistir(String nomeArquivo) throws IOException {  
        try (ObjectOutputStream out = new ObjectOutputStream(new  
FileOutputStream(nomeArquivo))) {  
            out.writeObject(pessoasJuridicas);  
        }  
    }  
  
    public void recuperar(String nomeArquivo) throws IOException,  
ClassNotFoundException {  
        try (ObjectInputStream in = new ObjectInputStream(new  
FileInputStream(nomeArquivo))) {  
            pessoasJuridicas = (List<PessoaJuridica>) in.readObject();  
        }  
    }  
}
```

```
}
```

Execução:



```
run:
Pessoas Fisicas Armazenadas.
Pessoas Fisicas Recuperadas.
ID: 1
Nome: Ana
CPF: 11111111111
Idade: 25
ID: 2
Nome: Carlos
CPF: 22222222222
Idade: 52
Pessoas Juridicas Armazenadas.
Pessoas Juridicas Recuperadas.
ID: 3
Nome: XPTO Sales
CNPJ: 33333333333333
ID: 4
Nome: XPTO Solutions
CNPJ: 44444444444444
BUILD SUCCESSFUL (total time: 0 seconds)
```

Análise e Conclusão:

1. Quais as vantagens e desvantagens do uso de herança?

R: Vantagens: Reutilização de código (herdar atributos e métodos); é possível criar subclasses; polimorfismo;

Desvantagens: não suporta herança múltipla; prende a classe principal e suas subclasses; rigidez (alterar a classe principal pode afetar todas as subclasses).

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

R: Porque permite que objetos dessa classe sejam convertidos em bytes e gravados em um arquivo, sendo útil para salvar o estado de objetos (como dados de aplicativos), em disco (podem ser recuperados depois).

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

R: API stream utiliza o paradigma funcional para realizar o gerenciamento de coleções Java, permitindo criar um código mais limpo ao lidar com coleções de dados.

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

R: Padrão de desenvolvimento `Serialization` e `Deserialization` (`Serializable`, `ObjectInputStream`, `ObjectOutputStream`).

2º Procedimento | Criação do Cadastro em Modo Texto

Arquivo `CadastroPOO.java`:

```
package cadastrapoo;
import model.PessoaFisica;
import model.PessoaJuridica;
import model.PessoaFisicaRepo;
import model.PessoaJuridicaRepo;
import java.util.Scanner;
import java.util.List;
import java.io.FileOutputStream;
```

```

import java.io.ObjectOutputStream;
import java.io.FileInputStream;
import java.io.ObjectInputStream;

public class CadastroP00 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int escolha;

        PessoaFisicaRepo repoPessoasFisicas = new
PessoaFisicaRepo("pessoasFisicas.dat");
        PessoaJuridicaRepo repoPessoasJuridicas = new
PessoaJuridicaRepo("pessoasJuridicas.dat");

        do {
            System.out.println("Menu:");
            System.out.println("1. Incluir");
            System.out.println("2. Alterar");
            System.out.println("3. Excluir");
            System.out.println("4. Buscar pelo ID");
            System.out.println("5. Exibir todos");
            System.out.println("6. Salvar dados");
            System.out.println("7. Recuperar dados");
            System.out.println("0. Finalizar programa");
            System.out.print("Escolha uma opcao: ");
            escolha = scanner.nextInt();
            scanner.nextLine();

            switch (escolha) {
                case 1:
                    System.out.println("Escolha o tipo (F - Pessoa
Fisica, J - Pessoa Juridica): ");
                    String tipo = scanner.nextLine().toUpperCase();

                    if (tipo.equals("F")) {
                        System.out.print("ID: ");
                        int id = scanner.nextInt();
                        scanner.nextLine();
                        System.out.print("Nome: ");
                        String nome = scanner.nextLine();
                        System.out.print("CPF: ");
                        String cpf = scanner.nextLine();
                        System.out.print("Idade: ");
                        int idade = scanner.nextInt();
                        scanner.nextLine();
                        PessoaFisica pessoaFisica = new PessoaFisica(id,
nome, cpf, idade);
                        repoPessoasFisicas.inserir(pessoaFisica);

```

```

        repoPessoasFisicas.persistir();
    } else if (tipo.equals("J")) {
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Nome: ");
        String nome = scanner.nextLine();
        System.out.print("CNPJ: ");
        String cnpj = scanner.nextLine();
        PessoaJuridica pessoaJuridica = new
PessoaJuridica(id, nome, cnpj);
        repoPessoasJuridicas.inserir(pessoaJuridica);
        repoPessoasJuridicas.persistir();
    } else {
        System.out.println("Opcao invalida. Use F para
Pessoa Fisica ou J para Pessoa Juridica.");
    }
    break;
case 2:
    System.out.println("Escolha o tipo (F - Pessoa
Fisica, J - Pessoa Juridica): ");
    String tipoAlterar =
scanner.nextLine().toUpperCase();

    if (tipoAlterar.equals("F")) {
        System.out.print("ID da Pessoa Fisica a ser
alterada: ");

        int idAlterar = scanner.nextInt();
        scanner.nextLine();
        PessoaFisica pessoaExistente =
repoPessoasFisicas.obter(idAlterar);

        if (pessoaExistente != null) {
            System.out.println("Dados atuais:");
            pessoaExistente.exibir();
            System.out.print("Novo nome: ");
            String novoNome = scanner.nextLine();
            System.out.print("Novo CPF: ");
            String novoCpf = scanner.nextLine();
            System.out.print("Nova idade: ");
            int novaIdade = scanner.nextInt();
            scanner.nextLine();
            pessoaExistente.setNome(novoNome);
            pessoaExistente.setCpf(novoCpf);
            pessoaExistente.setIdade(novaIdade);
            repoPessoasFisicas.persistir();
            System.out.println("Dados alterados com
sucesso.");
        }
    }
}

```

```

        } else {
            System.out.println("Pessoa Fisica nao
encontrada.");
        }
    } else if (tipoAlterar.equals("J")) {
        System.out.print("ID da Pessoa Juridica a ser
alterada: ");

        int idAlterar = scanner.nextInt();
        scanner.nextLine();

        PessoaJuridica empresaExistente =
repoPessoasJuridicas.obter(idAlterar);

        if (empresaExistente != null) {
            System.out.println("Dados atuais:");
            empresaExistente.exibir();
            System.out.print("Novo nome: ");
            String novoNome = scanner.nextLine();
            System.out.print("Novo CNPJ: ");
            String novoCnpj = scanner.nextLine();
            empresaExistente.setNome(novoNome);
            empresaExistente.setCnpj(novoCnpj);
            repoPessoasJuridicas.persistir();
            System.out.println("Dados alterados com
sucesso.");
        } else {
            System.out.println("Pessoa Juridica nao
encontrada.");
        }
    } else {
        System.out.println("Opcao invalida. Use F para
Pessoa Fisica ou J para Pessoa Juridica.");
    }
    break;

    case 3:
        System.out.println("Escolha o tipo (F - Pessoa
Fisica, J - Pessoa Juridica): ");
        String tipoExcluir =
scanner.nextLine().toUpperCase();

        if (tipoExcluir.equals("F")) {
            System.out.print("ID da Pessoa Fisica a ser
excluida: ");

            int idExcluir = scanner.nextInt();
            scanner.nextLine();
            repoPessoasFisicas.excluir(idExcluir);
            repoPessoasFisicas.persistir();

```

```

        System.out.println("Pessoa Fisica removida com
sucesso.");
    } else if (tipoExcluir.equals("J")) {
        System.out.print("ID da Pessoa Juridica a ser
excluida: ");

        int idExcluir = scanner.nextInt();
        scanner.nextLine();
        repoPessoasJuridicas.excluir(idExcluir);
        repoPessoasJuridicas.persistir();
        System.out.println("Pessoa Juridica removida com
sucesso.");
    } else {
        System.out.println("Opcao invalida. Use F para
Pessoa Fisica ou J para Pessoa Juridica.");
    }
    break;

    case 4:
        System.out.println("Escolha o tipo (F - Pessoa
Fisica, J - Pessoa Juridica): ");
        String tipoExibir = scanner.nextLine().toUpperCase();

        if (tipoExibir.equals("F")) {
            System.out.print("ID da Pessoa Fisica a ser
exibida: ");

            int idExibir = scanner.nextInt();
            scanner.nextLine();
            PessoaFisica pessoaFisica =
repoPessoasFisicas.obter(idExibir);

            if (pessoaFisica != null) {
                System.out.println("Dados da Pessoa Fisica
com ID " + idExibir + ":");
                pessoaFisica.exibir();
            } else {
                System.out.println("Pessoa Fisica nao
encontrada.");
            }
        } else if (tipoExibir.equals("J")) {
            System.out.print("ID da Pessoa Juridica a ser
exibida: ");

            int idExibir = scanner.nextInt();
            scanner.nextLine();
            PessoaJuridica pessoaJuridica =
repoPessoasJuridicas.obter(idExibir);

            if (pessoaJuridica != null) {

```

```

        System.out.println("Dados da Pessoa Juridica
com ID " + idExibir + ":");
        pessoaJuridica.exibir();
    } else {
        System.out.println("Pessoa Juridica nao
encontrada.");
    }
} else {
    System.out.println("Opcao invalida. Use F para
Pessoa Fisica ou J para Pessoa Juridica.");
}
break;

case 5:
    System.out.println("Escolha o tipo (F - Pessoa
Fisica, J - Pessoa Juridica): ");
    String tipoExibirTodos =
scanner.nextLine().toUpperCase();

    if (tipoExibirTodos.equals("F")) {
        List<PessoaFisica> pessoasFisicas =
repoPessoasFisicas.obterTodos();

        if (!pessoasFisicas.isEmpty()) {
            System.out.println("Pessoas Fisicas
cadastradas:");

            for (PessoaFisica pessoaFisica :
pessoasFisicas) {
                pessoaFisica.exibir();
            }
        } else {
            System.out.println("Nao ha Pessoas Fisicas
cadastradas.");
        }
    } else if (tipoExibirTodos.equals("J")) {
        List<PessoaJuridica> pessoasJuridicas =
repoPessoasJuridicas.obterTodos();

        if (!pessoasJuridicas.isEmpty()) {
            System.out.println("Pessoas Juridicas
cadastradas:");

            for (PessoaJuridica pessoaJuridica :
pessoasJuridicas) {
                pessoaJuridica.exibir();
            }
        } else {
            System.out.println("Nao ha Pessoas Juridicas
cadastradas.");
        }
    }
}

```

```

        }
    } else {
        System.out.println("Opcao invalida. Use F para
Pessoa Fisica ou J para Pessoa Juridica.");
    }
    break;

    case 6:
        System.out.println("Escolha o tipo (F - Pessoa
Fisica, J - Pessoa Juridica): ");
        String tipoSalvar = scanner.nextLine().toUpperCase();

        if (tipoSalvar.equals("F")) {
            System.out.print("Digite o nome do arquivo a ser
salvo: ");

            String prefixoFisica = scanner.nextLine();
            salvarDados(repoPessoasFisicas, prefixoFisica +
".fisica.bin");
        } else if (tipoSalvar.equals("J")) {
            System.out.print("Digite o nome do arquivo a ser
salvo: ");

            String prefixoJuridica = scanner.nextLine();
            salvarDados(repoPessoasJuridicas, prefixoJuridica
+ ".juridica.bin");
        } else {
            System.out.println("Opcao invalida. Use F para
Pessoa Fisica ou J para Pessoa Juridica.");
        }
        break;

    case 7:
        System.out.println("Escolha o tipo (F - Pessoa
Fisica, J - Pessoa Juridica): ");
        String tipoRecuperar =
scanner.nextLine().toUpperCase();

        if (tipoRecuperar.equals("F")) {
            System.out.print("Digite o nome do arquivo a ser
recuperado: ");

            String prefixoFisica = scanner.nextLine();
            recuperarDados(repoPessoasFisicas, prefixoFisica
+ ".fisica.bin");
        } else if (tipoRecuperar.equals("J")) {
            System.out.print("Digite o nome do arquivo a ser
recuperado: ");

            String prefixoJuridica = scanner.nextLine();
            recuperarDados(repoPessoasJuridicas,
prefixoJuridica + ".juridica.bin");

```

```

        } else {
            System.out.println("Opcao invalida. Use F para
Pessoa Fisica ou J para Pessoa Juridica.");
        }
        break;
    }

    } while (escolha != 0);

    scanner.close();
}

private static void salvarDados(PessoaFisicaRepo repo, String
nomeArquivo) {
    try {
        FileOutputStream fileOutputStream = new
FileOutputStream(nomeArquivo);
        ObjectOutputStream objectOutputStream = new
ObjectOutputStream(fileOutputStream);
        objectOutputStream.writeObject(repo);
        objectOutputStream.close();
        fileOutputStream.close();
        System.out.println("Dados salvos com sucesso em: " +
nomeArquivo);
    } catch (Exception e) {
        System.out.println("Erro ao salvar os dados. " +
e.getMessage());
    }
}

private static void salvarDados(PessoaJuridicaRepo repo, String
nomeArquivo) {
    try {
        FileOutputStream fileOutputStream = new
FileOutputStream(nomeArquivo);
        ObjectOutputStream objectOutputStream = new
ObjectOutputStream(fileOutputStream);
        objectOutputStream.writeObject(repo);
        objectOutputStream.close();
        fileOutputStream.close();
        System.out.println("Dados salvos com sucesso em: " +
nomeArquivo);
    } catch (Exception e) {
        System.out.println("Erro ao salvar os dados. " +
e.getMessage());
    }
}

private static void recuperarDados(PessoaFisicaRepo repo, String
nomeArquivo) {

```



```

        try {
            FileInputStream fileInputStream = new
FileInputStream(nomeArquivo);
            ObjectInputStream objectInputStream = new
ObjectInputStream(fileInputStream);
            PessoaFisicaRepo novoRepo = (PessoaFisicaRepo)
objectInputStream.readObject();
            objectInputStream.close();
            fileInputStream.close();
            repo.substituir(novoRepo);

            System.out.println("Dados recuperados com sucesso do arquivo:
" + nomeArquivo);
        } catch (Exception e) {
            System.out.println("Erro ao recuperar os dados. " +
e.getMessage());
        }
    }

    private static void recuperarDados(PessoaJuridicaRepo repo,
String nomeArquivo) {
        try {
            FileInputStream fileInputStream = new
FileInputStream(nomeArquivo);
            ObjectInputStream objectInputStream = new
ObjectInputStream(fileInputStream);
            PessoaJuridicaRepo novoRepo = (PessoaJuridicaRepo)
objectInputStream.readObject();
            objectInputStream.close();
            fileInputStream.close();
            repo.substituir(novoRepo);
            System.out.println("Dados recuperados com sucesso do arquivo:
" + nomeArquivo);
        } catch (Exception e) {
            System.out.println("Erro ao recuperar os dados. " +
e.getMessage());
        }
    }
}

```

Arquivo Pessoa.java:

```

package model;
import java.io.Serializable;

public class Pessoa implements Serializable {
    private int id;
    private String 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;
}
public void exibir() {
    System.out.println("ID: " + id);
    System.out.println("Nome: " + nome);
}
}

```

Arquivo PessoaFisica.java:

```

package model;
import java.io.Serializable;

public class PessoaFisica extends Pessoa implements Serializable {
    private String cpf;
    private int 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 void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
    }
}

```

Arquivo PessoaJuridica.java:

```

package model;
import java.io.Serializable;

public class PessoaJuridica extends Pessoa implements Serializable {
    private String 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 void exibir() {
        super.exibir();
        System.out.println("Cnpj: " + cnpj);
    }
}

```

Arquivo PessoaFisicaRepo.java:

```
package model;
import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaRepo implements Serializable {
    private List<PessoaFisica> pessoasFisicas = new ArrayList<>();
    private String arquivo;

    public PessoaFisicaRepo(String arquivo) {
        this.arquivo = arquivo;
        recuperar();
    }

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

    public void alterar(PessoaFisica pessoa) throws
    IllegalArgumentException {
        int id = pessoa.getId();
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            PessoaFisica existente = pessoasFisicas.get(i);
            if (existente.getId() == id) {
                pessoasFisicas.set(i,pessoa);
                persistir();
                return;
            }
        }
        throw new IllegalArgumentException("Pessoa nao encontrada" + id);
    }

    public void excluir(int id) throws IllegalArgumentException {
        PessoaFisica pessoaExcluir = null;
        for (PessoaFisica pessoa : pessoasFisicas) {
            if (pessoa.getId() == id) {
                pessoaExcluir = pessoa;
                break;
            }
        }
        if (pessoaExcluir != null) {
            pessoasFisicas.remove(pessoaExcluir);
        }
    }
}
```

```

        persistir();
    } else{
        throw new IllegalArgumentException ("Pessoa nao encontrada" +
id);
    }
}
}
public PessoaFisica obter(int id) throws IllegalArgumentException {
    for (PessoaFisica pessoa : pessoasFisicas) {
        if (pessoa.getId() == id) {
            return pessoa;
        }
    }
    throw new IllegalArgumentException ("Pessoa nao encontrada" +
id);
}
public List<PessoaFisica> obterTodos() {
    return pessoasFisicas;
}
public void persistir() {
    try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(arquivo))) {
        outputStream.writeObject(pessoasFisicas);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
public void recuperar() {
    try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(arquivo))) {
        List<PessoaFisica> dados = (List<PessoaFisica>)
inputStream.readObject();
        pessoasFisicas.clear();
        pessoasFisicas.addAll(dados);
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    }
}
public void substituir(PessoaFisicaRepo novoRepo) {
    this.pessoasFisicas.clear();
    this.pessoasFisicas.addAll(novoRepo.pessoasFisicas);
}
}

```

Arquivo PessoaJuridicaRepo.java:

```
package model;
import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaRepo implements Serializable {
    private List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
    private String arquivo;

    public PessoaJuridicaRepo(String arquivo) {
        this.arquivo = arquivo;

        recuperar();
    }

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

    public void alterar(PessoaJuridica empresa) throws
    IllegalArgumentException {
        int id = empresa.getId();
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            PessoaJuridica existente = pessoasJuridicas.get(i);
            if (existente.getId() == id) {
                pessoasJuridicas.set(i, empresa);
                persistir();
                return;
            }
        }
    }

    public void excluir(int id) throws IllegalArgumentException {
        PessoaJuridica empresaExcluir = null;
        for (PessoaJuridica empresa : pessoasJuridicas) {
            if (empresa.getId() == id) {
                empresaExcluir = empresa;
                break;
            }
        }
        if (empresaExcluir != null) {
            pessoasJuridicas.remove(empresaExcluir);
        }
    }
}
```

```

        persistir();
    }
}

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

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

public void persistir() {
    try (ObjectOutputStream outputStream = new ObjectOutputStream(new
FileOutputStream(arquivo))) {
        outputStream.writeObject(pessoasJuridicas);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void recuperar() {
    try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(arquivo))) {
        List<PessoaJuridica> dados = (List<PessoaJuridica>)
inputStream.readObject();
        pessoasJuridicas.clear();
        pessoasJuridicas.addAll(dados);
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    }
}

public void substituir(PessoaJuridicaRepo novoRepo) {
    this.pessoasJuridicas.clear();
    this.pessoasJuridicas.addAll(novoRepo.pessoasJuridicas);
}
}

```

Execução:

```
Output - CadastroPOO (run) x Search Results

run:
Menu:
1. Incluir
2. Alterar
3. Excluir
4. Buscar pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar programa
Escolha uma opcao: 5
Escolha o tipo (F - Pessoa Fisica, J - Pessoa Juridica):
F
Pessoas Fisicas cadastradas:
ID: 1
Nome: Anita
CPF: 12345678911
Idade: 27
ID: 2
Nome: Carlos
CPF: 22222222222
Idade: 52
Menu:
1. Incluir
2. Alterar
3. Excluir
4. Buscar pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar programa
Escolha uma opcao: 4
Escolha o tipo (F - Pessoa Fisica, J - Pessoa Juridica):
J
ID da Pessoa Juridica a ser exibida: 22
Pessoa Juridica nao encontrada.
Menu:
1. Incluir
2. Alterar
3. Excluir
4. Buscar pelo ID
5. Exibir todos
6. Salvar dados
7. Recuperar dados
0. Finalizar programa
Escolha uma opcao:
```


Análise e Conclusão:

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

R: Elementos estáticos em Java são coisas que pertencem à classe em vez de a objetos individuais. O método main é estático para ser o ponto de entrada do programa sem criar uma instância da classe.

2.Para que serve a classe Scanner?

R: A classe Scanner é usada para pegar informações do usuário ou de arquivos em Java, facilitando a entrada e saída de dados.

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

R: Classes de repositório são usadas para organizar o acesso a dados em um sistema, tornando o código mais organizado e facilitando a manutenção. Elas ajudam a esconder os detalhes de como os dados são armazenados e recuperados.