



Nível 1: Iniciando o caminho pelo Java (RPG0014)

GILSON MIRANDA NETO 202204437562

Campus Pintangueiras

Iniciando o caminho pelo Java – 2023.2 – 3º semestre do curso

Objetivo da Prática

- Utilizar herança e polimorfismo na definição de entidades.
- Utilizar persistência de objetos em arquivos binários.
- Implementar uma interface cadastral em modo texto.
- Utilizar o controle de exceções da plataforma Java.
- 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

CÓDIGOS SOLICITADOS NO ROTEIRO DE AULA:

Arquivo: CadastroPOO.java

```
package cadastropoo;

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

public class CadastroPOO {

    public static void main(String[] args) {
        // Teste com PessoaFisicaRepo

        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

        PessoaFisica pessoaFisical = new PessoaFisica(1, "Ana",
"111.111.111-11", 25);
        PessoaFisica pessoaFisica2 = new PessoaFisica(2, "Carlos",
"222.222.222-22", 52);

        repo1.inserir(pessoaFisical);
        repo1.inserir(pessoaFisica2);

        try {
            repo1.persistir("pessoas_fisicas.dat");
        } catch (IOException e) {
            System.out.println("erro");
            e.printStackTrace(System.out);
        }

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();
        try {
            repo2.recuperar("pessoas_fisicas.dat");
        } catch (IOException e) {
            e.printStackTrace(System.out);
        }

        System.out.println("Pessoas Físicas Recuperadas:");
        for (PessoaFisica pessoa : repo2.obterTodos()) {
            pessoa.exibir();
        }

        // Teste com PessoaJuridicaRepo
        PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

        PessoaJuridica pessoaJuridical = new PessoaJuridica(3, "XPTO
Sales", "33.333.333/3333-33");
```

```

        PessoaJuridica pessoaJuridica2 = new PessoaJuridica(4, "XPTO
Solutions", "44.444.444/4444-44");

        System.out.println("\n");

        repo3.inserir(pessoaJuridica1);
        repo3.inserir(pessoaJuridica2);

        try {
            repo3.persistir("pessoas_juridicas.dat");
        } catch (IOException e) {
            e.printStackTrace(System.out);
        }

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        try {
            repo4.recuperar("pessoas_juridicas.dat");
        } catch (IOException e) {
            e.printStackTrace(System.out);
        }

        System.out.println("Pessoas Jurídicas Recuperadas:");
        for (PessoaJuridica pessoa : repo4.obterTodos()) {
            pessoa.exibir();
        }
    }
}

```

Arquivo: Pessoas.java

```

package model;

import java.io.Serializable;

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

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

    // Getter para o campo 'id'
    public int getId() {
        return id;
    }

    // Setter para o campo 'id'
    public void setId(int id) {
        this.id = id;
    }

    // Getter para o campo 'nome'
    public String getNome() {
        return nome;
    }
}

```

```

    }

    // Setter para o campo 'nome'
    public void setNome(String nome) {
        this.nome = nome;
    }

    // Método para exibir os dados da pessoa
    public void exibir() {
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
    }
}

```

Arquivo: PessoasFisica.java

```

package model;

// Classe PessoaFisica que herda de Pessoa e implementa Serializable
import java.io.Serializable;
import model.Pessoa;

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;
    }

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

```

Arquivo: PessoasJuridica.java

```
package model;

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

// Classe PessoaJuridica que herda de Pessoa e implementa 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;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }
}
```

Arquivo: PessoaFisicaRepo.java

```
package model.gerenciadores;

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

public class PessoaFisicaRepo {
    private final List<PessoaFisica> listaPessoasFisicas;

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

    public void inserir(PessoaFisica pessoaFisica) {
        listaPessoasFisicas.add(pessoaFisica);
    }

    public void alterar(PessoaFisica pessoaFisica) {
        int index = encontrarIndicePorId(pessoaFisica.getId());
        if (index != -1) {
            listaPessoasFisicas.set(index, pessoaFisica);
        }
    }
}
```

```

public void excluir(int id) {
    int index = encontrarIndicePorId(id);
    if (index != -1) {
        listaPessoasFisicas.remove(index);
    }
}

public PessoaFisica obter(int id) {
    int index = encontrarIndicePorId(id);
    if (index != -1) {
        return listaPessoasFisicas.get(index);
    }
    return null;
}

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

public void persistir(String nomeArquivo) throws IOException {
    try (ObjectOutputStream outputStream = new
ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
        outputStream.writeObject(listaPessoasFisicas);
        System.out.println("Dados de Pessoa Física Armazenados no
arquivo: " + nomeArquivo);
    } catch (IOException e) {
        throw new IOException("Erro ao persistir os dados", e);
    }
}

public void recuperar(String nomeArquivo) throws IOException {
    try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
        List<PessoaFisica> pessoasFisicasRecuperadas =
(List<PessoaFisica>) inputStream.readObject();
        listaPessoasFisicas.clear();
        listaPessoasFisicas.addAll(pessoasFisicasRecuperadas);
        System.out.println("Dados de Pessoa Física Recuperados do
arquivo: " + nomeArquivo);
    } catch (IOException | ClassNotFoundException e) {
        throw new IOException("Erro ao recuperar os dados", e);
    }
}

private int encontrarIndicePorId(int id) {
    for (int i = 0; i < listaPessoasFisicas.size(); i++) {
        if (listaPessoasFisicas.get(i).getId() == id) {
            return i;
        }
    }
    return -1; // Retorna -1 se não encontrar o ID
}
}

```

Arquivo: PessoaJuridicaRepo.java

```

package model.gerenciadores;

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

public class PessoaJuridicaRepo {
    private final List<PessoaJuridica> listaPessoasJuridicas;

    public PessoaJuridicaRepo() {
        listaPessoasJuridicas = new ArrayList<>();
    }

    public void inserir(PessoaJuridica pessoaJuridica) {
        listaPessoasJuridicas.add(pessoaJuridica);
    }

    public void alterar(PessoaJuridica pessoaJuridica) {
        int index = encontrarIndicePorId(pessoaJuridica.getId());
        if (index != -1) {
            listaPessoasJuridicas.set(index, pessoaJuridica);
        }
    }

    public void excluir(int id) {
        int index = encontrarIndicePorId(id);
        if (index != -1) {
            listaPessoasJuridicas.remove(index);
        }
    }

    public PessoaJuridica obter(int id) {
        int index = encontrarIndicePorId(id);
        if (index != -1) {
            return listaPessoasJuridicas.get(index);
        }
        return null;
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream outputStream = new
ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            outputStream.writeObject(listaPessoasJuridicas);
            System.out.println("Dados de Pessoa Jurídica Armazenados
no arquivo: " + nomeArquivo);
        } catch (IOException e) {
            throw new IOException("Erro ao persistir os dados", e);
        }
    }

    public void recuperar(String nomeArquivo) throws IOException {
        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            List<PessoaJuridica> pessoasJuridicasRecuperadas =

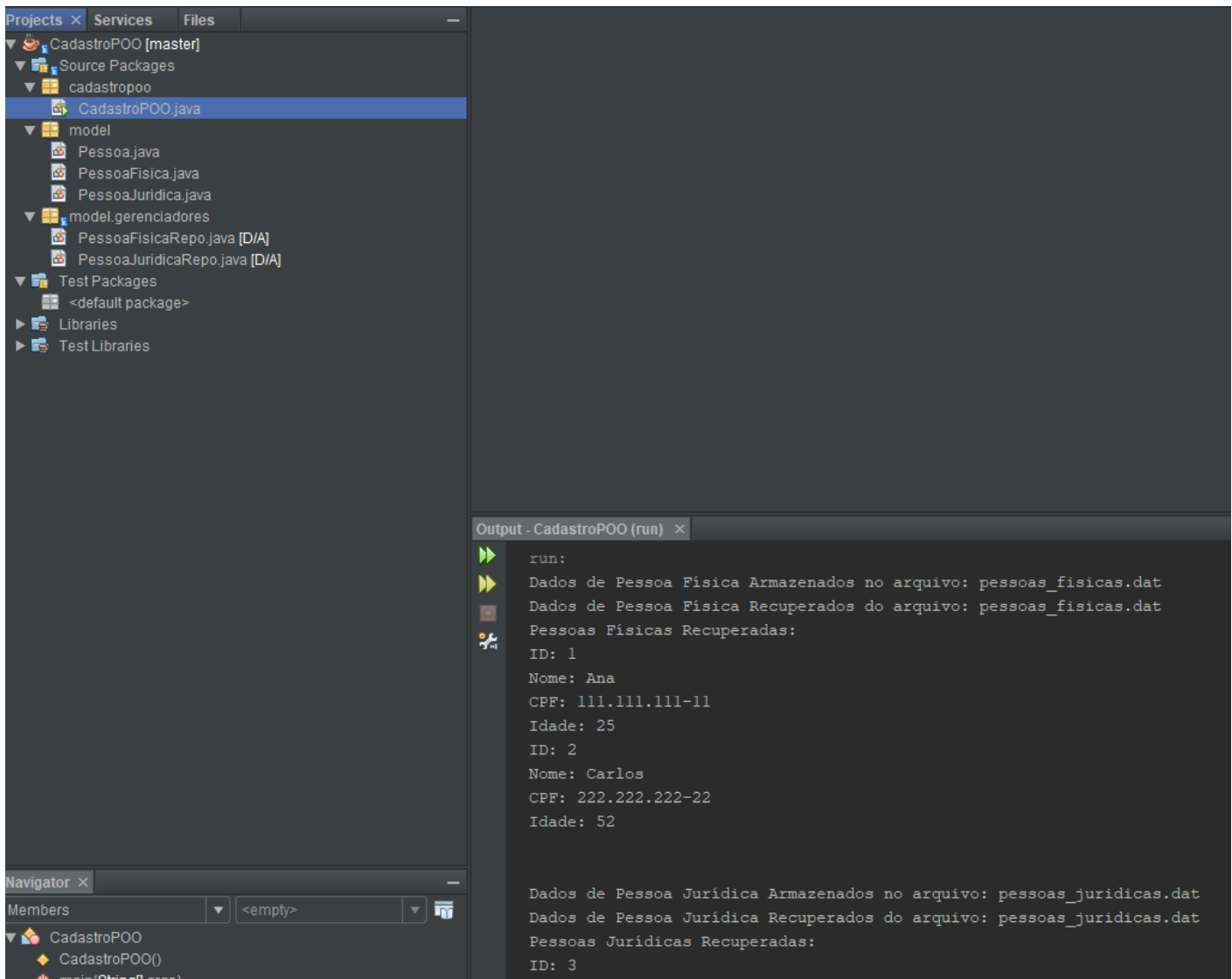
```

```
(List<PessoaJuridica>) inputStream.readObject();
    listaPessoasJuridicas.clear();
    listaPessoasJuridicas.addAll(pessoasJuridicasRecuperadas);
    System.out.println("Dados de Pessoa Jurídica Recuperados
do arquivo: " + nomeArquivo);
} catch (IOException | ClassNotFoundException e) {
    throw new IOException("Erro ao recuperar os dados", e);
}

}

private int encontrarIndicePorId(int id) {
    for (int i = 0; i < listaPessoasJuridicas.size(); i++) {
        if (listaPessoasJuridicas.get(i).getId() == id) {
            return i;
        }
    }
    return -1; // Retorna -1 se não encontrar o ID
}
}
```


RESULTADOS DA EXECUÇÃO DOS CÓDIGOS:



ANÁLISE E CONCLUSÃO:

- a. Quais as vantagens e desvantagens do uso de herança?
 - Reutilização de código
 - Abstração e Generalização

- Extensibilidade
- Polimorfismo

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

A interface Serializable é necessária ao efetuar persistência em arquivos binários em Java por causa da serialização e desserialização de objetos. Quando você deseja salvar objetos Java em um arquivo binário ou transmiti-los pela rede, é necessário transformá-los em uma sequência de bytes e, em seguida, recriá-los a partir desses bytes quando necessário.

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

É utilizado nas operações de processamento de dados de maneira funcional e declarativa em coleções de elementos, como listas, conjuntos, mapas e arrays.

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

Padrão DAO que é uma abordagem que separa a lógica de acesso a dados da lógica de negócios da aplicação. Ele fornece uma maneira estruturada e organizada de interagir com fontes de dados, como arquivos, bancos de dados ou serviços web.

2º Procedimento – Criação do Cadastro em Modo Texto

CÓDIGOS SOLICITADOS NO ROTEIRO DE AULA:

Arquivo: CadastroPOO.java

```
package cadastrapoo;

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

public class CadastroPOO {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

        while (true) {

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. Busca 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 = scanner.nextInt();
            scanner.nextLine(); // Consumir a quebra de linha

            switch (opcao) {
                case 1:
                    System.out.println("Escolha o tipo (F para Pessoa Física, J para Pessoa Jurídica):");
                    String tipoPessoa = scanner.nextLine();

                    if ("F".equalsIgnoreCase(tipoPessoa)) {
                        System.out.println("Digite os dados da Pessoa Física:");

                        System.out.print("ID: ");
                        int idFisica = scanner.nextInt();
                        scanner.nextLine();
                        System.out.print("Nome: ");
                        String nomeFisica = scanner.nextLine();
                        System.out.print("CPF: ");
```

```

        String cpf = scanner.nextLine();
        System.out.print("Idade: ");
        int idade = scanner.nextInt();
        scanner.nextLine(); // Consumir a quebra de
linha

        PessoaFisica pessoaFisica = new
PessoaFisica(idFisica, nomeFisica, cpf, idade);
        repoFisica.inserir(pessoaFisica);
    } else if ("J".equalsIgnoreCase(tipoPessoa)) {
        System.out.println("Digite os dados da Pessoa
Jurídica:");

        System.out.print("ID: ");
        int idJuridica = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Nome: ");
        String nomeJuridica = scanner.nextLine();
        System.out.print("CNPJ: ");
        String cnpj = scanner.nextLine();

        PessoaJuridica pessoaJuridica = new
PessoaJuridica(idJuridica, nomeJuridica, cnpj);
        repoJuridica.inserir(pessoaJuridica);
    } else {
        System.out.println("Opção inválida.");
    }
    break;

    case 2:
        System.out.println("Escolha o tipo (F para Pessoa
Física, J para Pessoa Jurídica):");
        String tipoAlterar = scanner.nextLine();

        System.out.print("Digite o ID da pessoa que deseja
alterar: ");

        int idAlterar = scanner.nextInt();
        scanner.nextLine(); // Consumir a quebra de linha

        if ("F".equalsIgnoreCase(tipoAlterar)) {
            PessoaFisica pessoaFisica =
repoFisica.obter(idAlterar);
            if (pessoaFisica != null) {
                System.out.println("Dados atuais da Pessoa
Física:");

                pessoaFisica.exibir();

                System.out.println("Digite os novos dados
da Pessoa Física:");

                System.out.print("Nome: ");
                String novoNome = scanner.nextLine();
                System.out.print("CPF: ");
                String novoCpf = scanner.nextLine();
                System.out.print("Idade: ");
                int novaIdade = scanner.nextInt();
                scanner.nextLine(); // Consumir a quebra
de linha

                pessoaFisica.setNome(novoNome);
                pessoaFisica.setCpf(novoCpf);
                pessoaFisica.setIdade(novaIdade);

```

```

        repoFisica.alterar(pessoaFisica);
    } else {
        System.out.println("Pessoa Física não
encontrada com o ID informado.");
    }
    } else if ("J".equalsIgnoreCase(tipoAlterar)) {
        PessoaJuridica pessoaJuridica =
repoJuridica.obter(idAlterar);
        if (pessoaJuridica != null) {
            System.out.println("Dados atuais da Pessoa
Jurídica:");

            pessoaJuridica.exibir();

            System.out.println("Digite os novos dados
da Pessoa Jurídica:");

            System.out.print("Nome: ");
            String novoNome = scanner.nextLine();
            System.out.print("CNPJ: ");
            String novoCnpj = scanner.nextLine();

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

            repoJuridica.alterar(pessoaJuridica);
        } else {
            System.out.println("Pessoa Jurídica não
encontrada com o ID informado.");
        }
    } else {
        System.out.println("Opção inválida.");
    }
    break;

    case 3:
        System.out.println("Escolha o tipo (F para Pessoa
Física, J para Pessoa Jurídica):");
        String tipoExcluir = scanner.nextLine();

        System.out.print("Digite o ID da pessoa que deseja
excluir: ");

        int idExcluir = scanner.nextInt();
        scanner.nextLine(); // Consumir a quebra de linha

        if ("F".equalsIgnoreCase(tipoExcluir)) {
            repoFisica.excluir(idExcluir);
        } else if ("J".equalsIgnoreCase(tipoExcluir)) {
            repoJuridica.excluir(idExcluir);
        } else {
            System.out.println("Opção inválida.");
        }
        break;

    case 4:
        System.out.println("Escolha o tipo (F para Pessoa
Física, J para Pessoa Jurídica):");
        String tipoObter = scanner.nextLine();
        System.out.print("Digite o ID da pessoa que deseja
obter: ");

        int idObter = scanner.nextInt();

```

```

        scanner.nextLine(); // Consumir a quebra de linha

        if ("F".equalsIgnoreCase(tipoObter)) {
            PessoaFisica pessoaFisica =
repoFisica.obter(idObter);
            if (pessoaFisica != null) {
                System.out.println("Dados da Pessoa
Física:");
                pessoaFisica.exibir();
            } else {
                System.out.println("Pessoa Física não
encontrada com o ID informado.");
            }
        } else if ("J".equalsIgnoreCase(tipoObter)) {
            PessoaJuridica pessoaJuridica =
repoJuridica.obter(idObter);
            if (pessoaJuridica != null) {
                System.out.println("Dados da Pessoa
Jurídica:");
                pessoaJuridica.exibir();
            } else {
                System.out.println("Pessoa Jurídica não
encontrada com o ID informado.");
            }
        } else {
            System.out.println("Opção inválida.");
        }
        break;

    case 5:
        System.out.println("Escolha o tipo (F para Pessoa
Física, J para Pessoa Jurídica:");
        String tipoExibirTodos = scanner.nextLine();

        if ("F".equalsIgnoreCase(tipoExibirTodos)) {
            System.out.println("Pessoas Físicas:");
            for (PessoaFisica pessoa :
repoFisica.obterTodos()) {
                pessoa.exibir();
            }
        } else if ("J".equalsIgnoreCase(tipoExibirTodos))
{
            System.out.println("Pessoas Jurídicas:");
            for (PessoaJuridica pessoa :
repoJuridica.obterTodos()) {
                pessoa.exibir();
            }
        } else {
            System.out.println("Opção inválida.");
        }
        break;

    case 6:
        System.out.print("Digite o prefixo dos arquivos
para salvar os dados: ");
        String prefixoSalvar = scanner.nextLine();
        try {
            repoFisica.persistir(prefixoSalvar +
".fisica.bin");
            repoJuridica.persistir(prefixoSalvar +

```



```

public int getId() {
    return id;
}

// Setter para o campo 'id'
public void setId(int id) {
    this.id = id;
}

// Getter para o campo 'nome'
public String getNome() {
    return nome;
}

// Setter para o campo 'nome'
public void setNome(String nome) {
    this.nome = nome;
}

// Método para exibir os dados da pessoa
public void exibir() {
    System.out.println("ID: " + id);
    System.out.println("Nome: " + nome);
}
}

```

Arquivo: PessoaFisica.java

```

package model;

// Classe PessoaFisica que herda de Pessoa e implementa Serializable
import java.io.Serializable;
import model.Pessoa;

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;
    }
}

```



```

@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;

// Classe PessoaJuridica que herda de Pessoa e implementa 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 void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }
}

```

Arquivo: PessoaFisicaRepo.java

```

package model.gerenciadores;

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

public class PessoaFisicaRepo {
    private final List<PessoaFisica> listaPessoasFisicas;

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

    public void inserir(PessoaFisica pessoaFisica) {
        listaPessoasFisicas.add(pessoaFisica);
    }
}

```

```

    public void alterar(PessoaFisica pessoaFisica) {
        int index = encontrarIndicePorId(pessoaFisica.getId());
        if (index != -1) {
            listaPessoasFisicas.set(index, pessoaFisica);
        }
    }

    public void excluir(int id) {
        int index = encontrarIndicePorId(id);
        if (index != -1) {
            listaPessoasFisicas.remove(index);
        }
    }

    public PessoaFisica obter(int id) {
        int index = encontrarIndicePorId(id);
        if (index != -1) {
            return listaPessoasFisicas.get(index);
        }
        return null;
    }

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

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream outputStream = new
ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            outputStream.writeObject(listaPessoasFisicas);
            System.out.println("Dados de Pessoa Física Armazenados no
arquivo: " + nomeArquivo);
        } catch (IOException e) {
            throw new IOException("Erro ao persistir os dados", e);
        }
    }

    public void recuperar(String nomeArquivo) throws IOException {
        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            List<PessoaFisica> pessoasFisicasRecuperadas =
(List<PessoaFisica>) inputStream.readObject();
            listaPessoasFisicas.clear();
            listaPessoasFisicas.addAll(pessoasFisicasRecuperadas);
            System.out.println("Dados de Pessoa Física Recuperados do
arquivo: " + nomeArquivo);
        } catch (IOException | ClassNotFoundException e) {
            throw new IOException("Erro ao recuperar os dados", e);
        }
    }

    private int encontrarIndicePorId(int id) {
        for (int i = 0; i < listaPessoasFisicas.size(); i++) {
            if (listaPessoasFisicas.get(i).getId() == id) {
                return i;
            }
        }
        return -1; // Retorna -1 se não encontrar o ID
    }
}

```

Arquivo: PessoaJuridicaRepo.java

```
package model.gerenciadores;

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

public class PessoaJuridicaRepo {
    private final List<PessoaJuridica> listaPessoasJuridicas;

    public PessoaJuridicaRepo() {
        listaPessoasJuridicas = new ArrayList<>();
    }

    public void inserir(PessoaJuridica pessoaJuridica) {
        listaPessoasJuridicas.add(pessoaJuridica);
    }

    public void alterar(PessoaJuridica pessoaJuridica) {
        int index = encontrarIndicePorId(pessoaJuridica.getId());
        if (index != -1) {
            listaPessoasJuridicas.set(index, pessoaJuridica);
        }
    }

    public void excluir(int id) {
        int index = encontrarIndicePorId(id);
        if (index != -1) {
            listaPessoasJuridicas.remove(index);
        }
    }

    public PessoaJuridica obter(int id) {
        int index = encontrarIndicePorId(id);
        if (index != -1) {
            return listaPessoasJuridicas.get(index);
        }
        return null;
    }

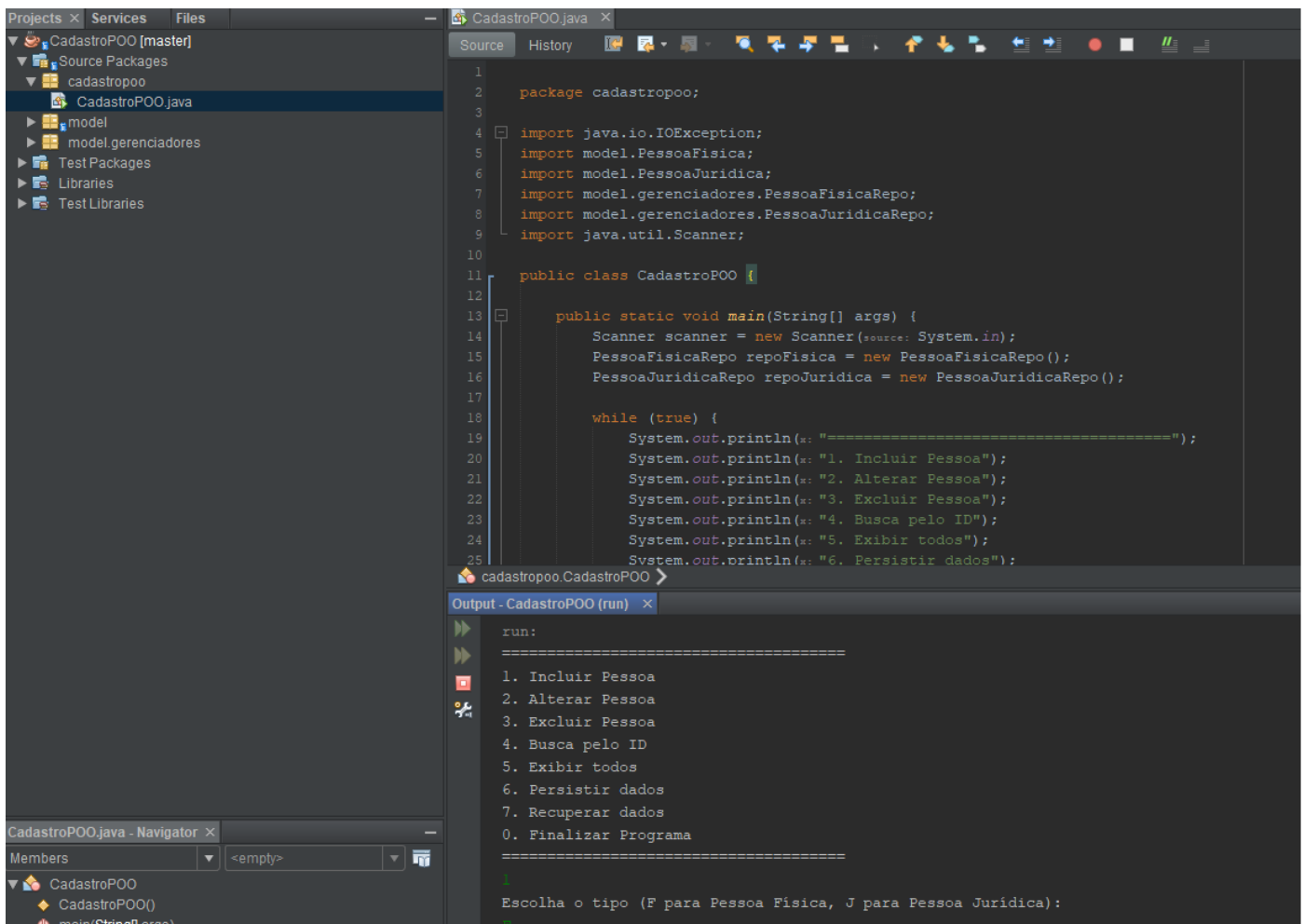
    public List<PessoaJuridica> obterTodos() {
        return listaPessoasJuridicas;
    }

    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream outputStream = new
ObjectOutputStream(new FileOutputStream(nomeArquivo))) {
            outputStream.writeObject(listaPessoasJuridicas);
            System.out.println("Dados de Pessoa Jurídica Armazenados
no arquivo: " + nomeArquivo);
        } catch (IOException e) {
            throw new IOException("Erro ao persistir os dados", e);
        }
    }
}
```

```
    public void recuperar(String nomeArquivo) throws IOException {
        try (ObjectInputStream inputStream = new ObjectInputStream(new
FileInputStream(nomeArquivo))) {
            List<PessoaJuridica> pessoasJuridicasRecuperadas =
(List<PessoaJuridica>) inputStream.readObject();
            listaPessoasJuridicas.clear();
            listaPessoasJuridicas.addAll(pessoasJuridicasRecuperadas);
            System.out.println("Dados de Pessoa Jurídica Recuperados
do arquivo: " + nomeArquivo);
        } catch (IOException | ClassNotFoundException e) {
            throw new IOException("Erro ao recuperar os dados", e);
        }
    }

    private int encontrarIndicePorId(int id) {
        for (int i = 0; i < listaPessoasJuridicas.size(); i++) {
            if (listaPessoasJuridicas.get(i).getId() == id) {
                return i;
            }
        }
        return -1; // Retorna -1 se não encontrar o ID
    }
}
```

RESULTADOS DA EXECUÇÃO DOS CÓDIGOS:



The screenshot displays an IDE with the following components:

- Left Panel (Project Explorer):** Shows the project structure for 'CadastroPOO [master]'. It includes 'Source Packages' with 'cadastropoo' and 'model'. The 'cadastropoo' package contains 'CadastroPOO.java'. The 'model' package contains 'model.gerenciadores' and 'Test Packages'. The 'Libraries' and 'Test Libraries' are also listed.
- Top Panel (Source Editor):** Displays the source code of 'CadastroPOO.java'. The code is as follows:

```
1 package cadastropoo;
2
3
4 import java.io.IOException;
5 import model.PessoaFisica;
6 import model.PessoaJuridica;
7 import model.gerenciadores.PessoaFisicaRepo;
8 import model.gerenciadores.PessoaJuridicaRepo;
9 import java.util.Scanner;
10
11 public class CadastroPOO {
12
13     public static void main(String[] args) {
14         Scanner scanner = new Scanner(System.in);
15         PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
16         PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();
17
18         while (true) {
19             System.out.println("=====");
20             System.out.println("1. Incluir Pessoa");
21             System.out.println("2. Alterar Pessoa");
22             System.out.println("3. Excluir Pessoa");
23             System.out.println("4. Busca pelo ID");
24             System.out.println("5. Exibir todos");
25             System.out.println("6. Persistir dados");
```
- Bottom Panel (Output Console):** Shows the output of the program execution. It starts with 'run:' followed by a list of menu options: '1. Incluir Pessoa', '2. Alterar Pessoa', '3. Excluir Pessoa', '4. Busca pelo ID', '5. Exibir todos', '6. Persistir dados', '7. Recuperar dados', and '0. Finalizar Programa'. The output ends with 'Escolha o tipo (F para Pessoa Fisica, J para Pessoa Juridica):'.

ANÁLISE E CONCLUSÃO:

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

Um método estático é um método que pertence à classe em vez de a instâncias individuais. O método `public static void main(String[] args)` é o ponto de entrada de um programa Java. Quando você inicia um programa Java, a JVM (Java Virtual Machine) procura por este método específico e o executa.

b. Para que serve a classe Scanner?

É usada para ler entrada de dados do usuário ou de outras fontes, como arquivos. Ela fornece métodos para ler diferentes tipos de dados, como números inteiros, números de ponto flutuante, caracteres e strings, de forma fácil e eficiente. A classe Scanner é frequentemente usada para criar programas interativos que podem receber entrada do usuário.

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

A classe de repositório permitiu agrupar todas as funcionalidades relacionadas às operações com arquivos, organizando melhor o projeto

CONCLUSÃO FINAL

Essa atividade conseguiu atingir o objetivo de implantar um Sistema cadastral em JAVA utilizando POO (programação orientada a objeto) e persistência em arquivos binários. Durante o desenvolvimento tivemos contato práticos com conceitos como herança, polimorfismo, construtores, importações, interface do usuário e tratamento de exceções.