POLO BARRA WORLD

CURSO DESENVOLVIMENTO FULL STACK

Disciplina: Nível 1 – Iniciando o Caminho pelo Java

Turma: 2024.3 Flex / 3º Semestre

Aluno: Paola Savedra Barreiros

Matrícula: 2023.0701.4731

Repositório Github: [pasavedra/RPG0014-Iniciando-o-caminho-pelo-Java (github.com)]

Missão Prática | Nível 1 | Mundo 3

Título da prática: RPG0014 – Iniciando o Caminho pelo Java

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.

CÓDIGOS SOLICITADOS NO ROTEIRO

PROCEDIMENTO 01:

- **Pessoa.java**

```
package model;

/**
 *
 * @author pasav
 */
import java.io.Serializable;

```

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

    // construtor
    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;
    }

    //Método exibir
    public void exibir(){
        System.out.print("id: "+this.id + "\n" +
"Nome: " + this.nome + "\n");
    }
}
```

- **PessoaFisica.java**

```java
package model;

import java.io.Serializable;

```

```java
public class PessoaFisica extends Pessoa implements Serializable {

    private String cpf;
    private int idade;

    //Construtor
    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;
    }

    //Método exibir
    public void exibir(){
        System.out.print("\n"+"id: "+getId()+"\nNome: "+getNome()+ "\nCPF: "+this.cpf + "\n" + "Idade: " + this.idade + "\n");
    }
}
```

- **PessoaJurica.java**

```
package model;

import java.io.Serializable;

public class PessoaJuridica extends Pessoa
implements Serializable{

    private String cnpj;

     //Constutor
    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 exibir(){
        System.out.print("\n" + "id: "+getId()+
"\nNome: "+getNome()+"\nCNPJ: "+this.cnpj+"\n");
    }
}
```

- **PessoaFisicaRepo.java**

```
package model;
import java.util.ArrayList;
import java.util.NoSuchElementException;
import java.util.Optional;
```

```java
import java.io.*;

public class PessoaFisicaRepo {

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

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

    public void alterar(PessoaFisica pessoaFisica,
String novoNome, String novoCpf, int novaIdade) {
        pessoaFisica.setNome(novoNome);
        pessoaFisica.setCpf(novoCpf);
        pessoaFisica.setIdade(novaIdade);
    }

    public void excluir(int id) {
        try{
        listaPessoasFisicas.remove(obter(id));
        }catch(NoSuchElementException e){
            System.out.println("erro");
        }
    }

    public PessoaFisica obter(int id) {
        Optional<PessoaFisica>
pessoaFisicaLocalizada =
listaPessoasFisicas.stream().
                filter(pessoaFisica ->
pessoaFisica.getId() == id).findFirst();
        if (pessoaFisicaLocalizada.isPresent()) {
           return pessoaFisicaLocalizada.get();
        } else {
           return null;
        }
    }
```

```
•
•     public ArrayList<PessoaFisica> obterTodos(){
•         return listaPessoasFisicas;
•     }
•
•     public void persistir(String arquivo)throws
  IOException {
•         ObjectOutputStream arquivoSaida = new
  ObjectOutputStream(new FileOutputStream(arquivo));
•         arquivoSaida.writeObject(listaPessoasFisicas
  );
•         arquivoSaida.close();
•         System.out.println("Dados de pessoas fisicas
  armazenados.");
•     }
•
•     public void recuperar(String arquivo) throws
  IOException, ClassNotFoundException {
•         ObjectInputStream arquivoEntrada = new
  ObjectInputStream(new FileInputStream(arquivo));
•         listaPessoasFisicas =
  (ArrayList<PessoaFisica>)
  arquivoEntrada.readObject();
•         arquivoEntrada.close();
•         System.out.println("Dados de pessoas fisicas
  recuperados.");
•     }
• }
```

- **PessoaJuridicaRepo.java**

```
• package model;
• import java.util.ArrayList;
• import java.util.Optional;
• import java.io.*;
•
•
• public class PessoaJuridicaRepo {
•
```

```
    private ArrayList<PessoaJuridica>
listaPessoasJuridicas = new ArrayList<>();

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

    public void alterar(PessoaJuridica
pessoaJuridica, String novoNome, String novoCnpj) {
        pessoaJuridica.setNome(novoNome);
        pessoaJuridica.setCnpj(novoCnpj);
    }

    public void excluir(int id){
        listaPessoasJuridicas.remove(obter(id));
    }

    public PessoaJuridica obter(int id) {
        Optional<PessoaJuridica>
pessoaJuridicaLocalizada =
listaPessoasJuridicas.stream().
                filter(pessoaJuridica ->
pessoaJuridica.getId() == id).findFirst();
        if (pessoaJuridicaLocalizada.isPresent()) {
           return pessoaJuridicaLocalizada.get();
        } else {
           return null;
        }
    }

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

    public void persistir(String arquivo)throws
IOException {
```

```
        ObjectOutputStream arquivoSaida = new
ObjectOutputStream(new FileOutputStream(arquivo));
        arquivoSaida.writeObject(listaPessoasJuridic
as);
        arquivoSaida.close();
        System.out.println("\nDados das pessoas
juridicas armazenados.");
    }

    public void recuperar(String arquivo)throws
IOException, ClassNotFoundException {
        ObjectInputStream arquivoEntrada = new
ObjectInputStream(new FileInputStream(arquivo));
        listaPessoasJuridicas =
(ArrayList<PessoaJuridica>)
arquivoEntrada.readObject();
        arquivoEntrada.close();
        System.out.println("Dados de pessoas
juridicas recuperados.");
    }
}
```

- **CadastroPOO**

```
package model;
import java.io.IOException;

/**
 *
 * @author pasav
 */
public class CadastroPOO {
    public static void main(String[] args) {

        PessoaFisicaRepo repo1 = new
PessoaFisicaRepo();
        PessoaFisica pessoaFisica1 = new
PessoaFisica(1, "Ana", "11111111111", 25);
```

```
        PessoaFisica pessoaFisica2 = new
PessoaFisica(2, "Carlos Jose", "22222222222", 52);
        repo1.inserir(pessoaFisica1);
        repo1.inserir(pessoaFisica2);
        try {
            repo1.persistir("listaPessoasFisicas.bin
");
        } catch (IOException erro) {
            System.out.println("Erro ao persistir os
dados: " + erro.getMessage());
        }

        PessoaFisicaRepo repo2 = new
PessoaFisicaRepo();

        try {
            repo2.recuperar("listaPessoasFisicas.bin
");
            repo2.obterTodos()
                    .forEach(pessoaFisica -> {
                        pessoaFisica.exibir();
                    });

        } catch (IOException |
ClassNotFoundException erro) {
            System.out.println("Erro ao recuperar os
dados: " + erro.getMessage());
        }


        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("listaPessoasJuridicas.bin");
        } catch (IOException erro) {
            System.out.println("Erro ao persistir os dados: " + erro.getMessage());
        }

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();
        try {
            repo4.recuperar("listaPessoasJuridicas.bin");
            repo4.obterTodos()
                    .forEach(pessoaJuridica -> {
                        pessoaJuridica.exibir();
                    });
        } catch (IOException | ClassNotFoundException erro) {
            System.out.println("Erro ao recuperar os dados: " + erro.getMessage());
        }
    }
}

```

PROCEDIMENTO 02:

```
package model;
import java.io.IOException;
import java.util.Scanner;
/**
 *
 * @author pasav
 */
public class CadastroPOO2 {

    public static void main(String[] args) {

        PessoaFisicaRepo pfRepo = new
PessoaFisicaRepo();
        PessoaJuridicaRepo pjRepo = new
PessoaJuridicaRepo();

        Scanner scan = new Scanner(System.in);
        String escolha;

        do {
            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/Salvar
Dados");
            System.out.println("7 -
Recuperar/Carregar Dados");
            System.out.println("0 - Finalizar
Programa");
```

```
            System.out.println("==================
==========");

            escolha = scan.next();

            switch (escolha) {

                // Incluir
                case "1":
                    do {
                        System.out.println("=======
=====================");
                        System.out.println("F -
Pessoa Fisica | J - Pessoa Juridica | M - Menu");

                        escolha = scan.next();
                        scan.nextLine();

                        switch
(escolha.toUpperCase()) {

                            case "F":
                                System.out.print("Di
gite o id da pessoa: ");
                                int idInformado =
scan.nextInt();
                                System.out.println("
Insira os dados... ");
                                scan.nextLine();
                                System.out.print("No
me: ");
                                String nome =
scan.nextLine();
                                System.out.print("CP
F: ");
                                String cpf =
scan.nextLine();
```

```
                                    System.out.print("Id
ade: ");
                                    int idade =
scan.nextInt();

                                    int pfRepoSize =
pfRepo.obterTodos().size();

                                    PessoaFisica
pessoaFisica = new PessoaFisica(idInformado, nome,
cpf, idade);
                                    pfRepo.inserir(pesso
aFisica);

                                    System.out.println("
Inclusao realizada com sucesso!");
                                    pessoaFisica.exibir(
);
                                    break;

                                case "J":
                                    System.out.print("Di
gite o id da pessoa: ");
                                    int idInformado2 =
scan.nextInt();
                                    scan.nextLine();
                                    System.out.print("No
me: ");
                                    nome =
scan.nextLine();
                                    System.out.print("CN
PJ: ");
                                    String cnpj =
scan.nextLine();

                                    int pjRepoSize =
pjRepo.obterTodos().size();

```

```
                                PessoaJuridica pessoaJuridica = new PessoaJuridica(idInformado2, nome, cnpj);
                                pjRepo.inserir(pessoaJuridica);

                                System.out.println("Inclusao realizada com sucesso!");
                                pessoaJuridica.exibir();
                                break;

                            case "M":
                                break;

                            default:
                                System.out.println("Opcao invalida.");
                                break;
                        }
                    } while (!escolha.equalsIgnoreCase("M"));
                    break;

                // Alterar
                case "2":
                    do {
                        System.out.println("==========================");
                        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");

                        escolha = scan.next();
                        scan.nextLine();

                        switch (escolha.toUpperCase()) {
```

```
                                case "F":
                                    System.out.println("
Digite o ID da pessoa: ");
                                    int idPessoaFisica =
scan.nextInt();
                                    scan.nextLine();

                                    PessoaFisica
pessoaFisicaLocalizada =
pfRepo.obter(idPessoaFisica);

                                    if
(pessoaFisicaLocalizada != null) {
                                        pessoaFisicaLoca
lizada.exibir();

                                        System.out.print
ln("Nome atual: " +
pessoaFisicaLocalizada.getNome());
                                        System.out.print
("Novo nome: ");
                                        String novoNome
= scan.nextLine();

                                        System.out.print
ln("CPF atual: " + pessoaFisicaLocalizada.getCpf());
                                        System.out.print
("Novo CPF: ");
                                        String novoCPF =
scan.nextLine();

                                        System.out.print
ln("Idade atual: " +
pessoaFisicaLocalizada.getIdade());
                                        System.out.print
("Nova Idade: ");
```

```
                                    int novaIdade =
scan.nextInt();

                                    pfRepo.alterar(p
essoaFisicaLocalizada, novoNome, novoCPF,
novaIdade);

                                    System.out.print
ln("Pessoa alterada com sucesso!");
                                } else
                                    System.out.print
ln("Pessoa nao localizada! ");
                                break;

                            case "J":
                                System.out.println("
Digite o ID da pessoa: ");
                                int idPessoaJuridica
= scan.nextInt();
                                scan.nextLine();

                                PessoaJuridica
pessoaJuridicaLocalizada =
pjRepo.obter(idPessoaJuridica);

                                if
(pessoaJuridicaLocalizada != null) {
                                    pessoaJuridicaLo
calizada.exibir();

                                    System.out.print
ln("Nome atual: " +
pessoaJuridicaLocalizada.getNome());
                                    System.out.print
ln("Novo nome: ");
                                    String novoNome
= scan.nextLine();

```

```
                                    System.out.println("CNPJ atual: " +
pessoaJuridicaLocalizada.getCnpj());
                                    System.out.println("Novo CNPJ: ");
                                    String novoCNPJ = scan.nextLine();

                                    pjRepo.alterar(pessoaJuridicaLocalizada, novoNome, novoCNPJ);

                                    System.out.println("Pessoa alterada com sucesso!");
                                } else
                                    System.out.println("Pessoa nao localizada!");
                                break;

                            case "M":
                                break;

                            default:
                                System.out.println("Opcao invalida.");
                                break;
                        }
                    } while (!escolha.equalsIgnoreCase("M"));
                    break;

                // EXCLUIR
                case "3":
                    do {
                        System.out.println("==========================");
                        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");
```

```
                        escolha = scan.next();
                        scan.nextLine();

                        switch
(escolha.toUpperCase()) {

                            case "F":
                                System.out.println("
Digite o ID da pessoa: ");
                                int idPessoaFisica =
scan.nextInt();

                                PessoaFisica
pessoaFisicaLocalizada =
pfRepo.obter(idPessoaFisica);

                                if
(pessoaFisicaLocalizada != null) {
                                    pessoaFisicaLoca
lizada.exibir();
                                    pfRepo.excluir(i
dPessoaFisica);

                                    System.out.print
ln("Pessoa excluida com sucesso!");
                                } else
                                    System.out.print
ln("Pessoa nao localizada!");
                                break;

                            case "J":
                                System.out.println("
Digite o ID da pessoa: ");
                                int idPessoaJuridica
= scan.nextInt();

```

```
                                    PessoaJuridica
    pessoaJuridicaLocalizada =
    pjRepo.obter(idPessoaJuridica);

                                    if
    (pessoaJuridicaLocalizada != null) {
                                        pessoaJuridicaLo
    calizada.exibir();

                                        pjRepo.excluir(i
    dPessoaJuridica);

                                        System.out.print
    ln("Pessoa excluida com sucesso!");
                                    } else
                                        System.out.print
    ln("Pessoa nao localizada!");
                                    break;

                                case
    "M":
                                    break;

                                default:
                                    System.out.println("
    Opcao invalida.");
                                    break;
                            }

                        } while
    (!escolha.equalsIgnoreCase("M"));
                        break;

                    // obterId
                    case "4":
                        do {
                            System.out.println("=======
    ======================");
```

```
                        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | M - Menu");

                        escolha = scan.next();
                        scan.nextLine();

                        switch (escolha.toUpperCase()) {

                            case "F":
                                System.out.println("Digite o ID da pessoa: ");
                                int idPessoaFisica = scan.nextInt();

                                PessoaFisica pessoaFisicaLocalizada = pfRepo.obter(idPessoaFisica);

                                if (pessoaFisicaLocalizada != null) {
                                    System.out.println("Pessoa localizada!");
                                    pessoaFisicaLocalizada.exibir();
                                } else
                                    System.out.println("Pessoa nao localizada!");
                                break;

                            case "J":
                                System.out.println("Digite o ID da pessoa: ");
                                int idPessoaJuridica = scan.nextInt();

```

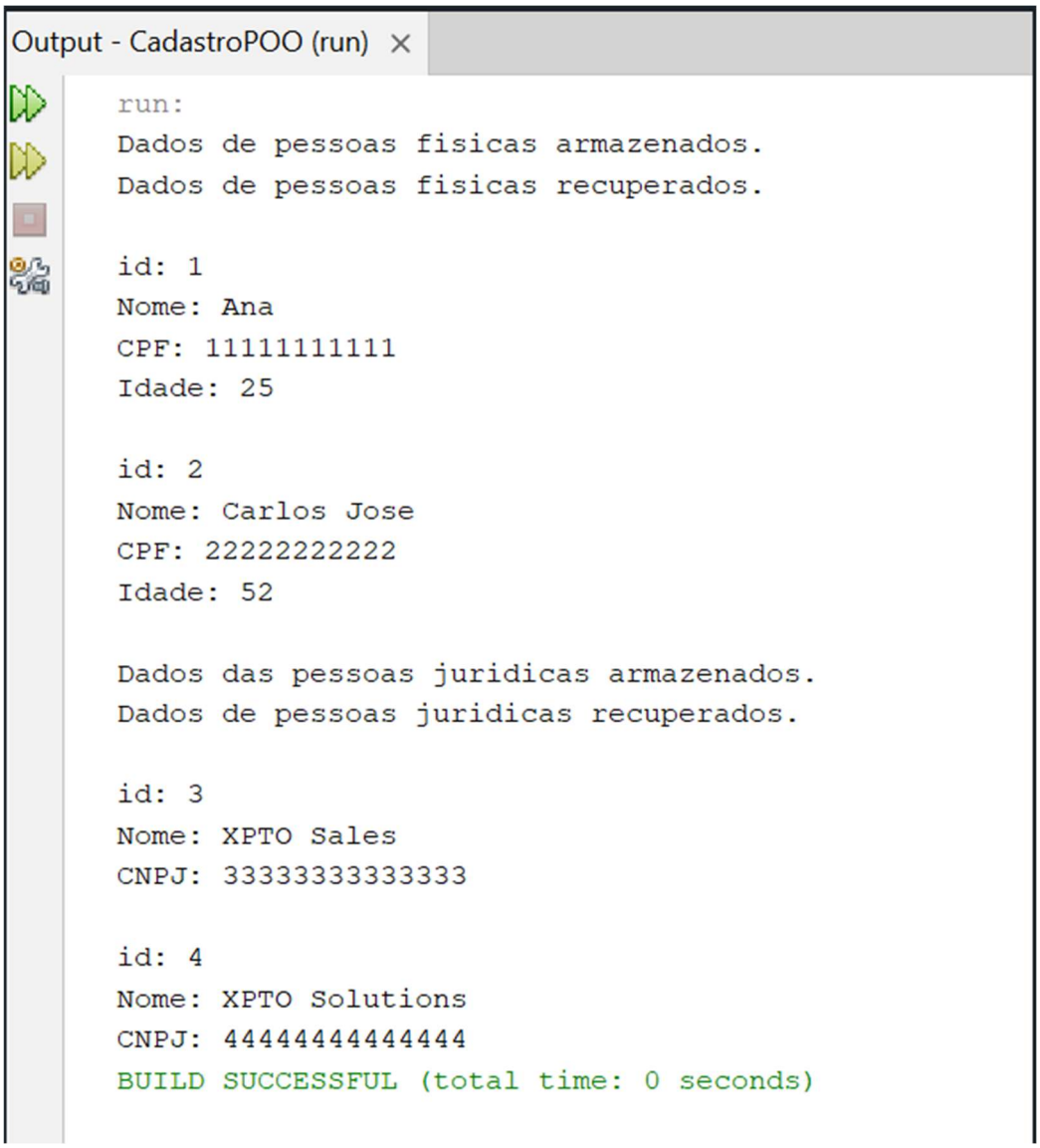
```
                                PessoaJuridica
pessoaJuridicaLocalizada =
pjRepo.obter(idPessoaJuridica);

                                if
(pessoaJuridicaLocalizada != null) {
                                    System.out.print
ln("Pessoa localizada!");

                                    pessoaJuridicaLo
calizada.exibir();
                                } else
                                    System.out.print
ln("Pessoa nao localizada!");
                                break;

                            case
"M":
                                break;

                            default:
                                System.out.println("
Opcao invalida.");
                                break;
                        }

                    } while
(!escolha.equalsIgnoreCase("M"));
                    break;

                //obterTodos
                case "5":
                    do {
                        System.out.println("========
=======================");
                        System.out.println("F -
Pessoa Fisica | J - Pessoa Juridica | M - Menu");

```

```
                        escolha = scan.next();
                        scan.nextLine();

                        switch
(escolha.toUpperCase()) {

                            case "F":
                                System.out.println("
Lista de pessoas Fisicas: ");
                                pfRepo.obterTodos()
                                        .forEach(pes
soaFisica -> {
                                            pessoaFi
sica.exibir();
                                            System.o
ut.println();
                                        });
                                break;

                            case "J":
                                System.out.println("
Lista de pessoas juridicas: ");
                                pjRepo.obterTodos()
                                        .forEach(pes
soaJuridica -> {
                                            pessoaJu
ridica.exibir();
                                            System.o
ut.println();
                                        });
                                break;

                            case
"M":
                                break;

                            default:
```

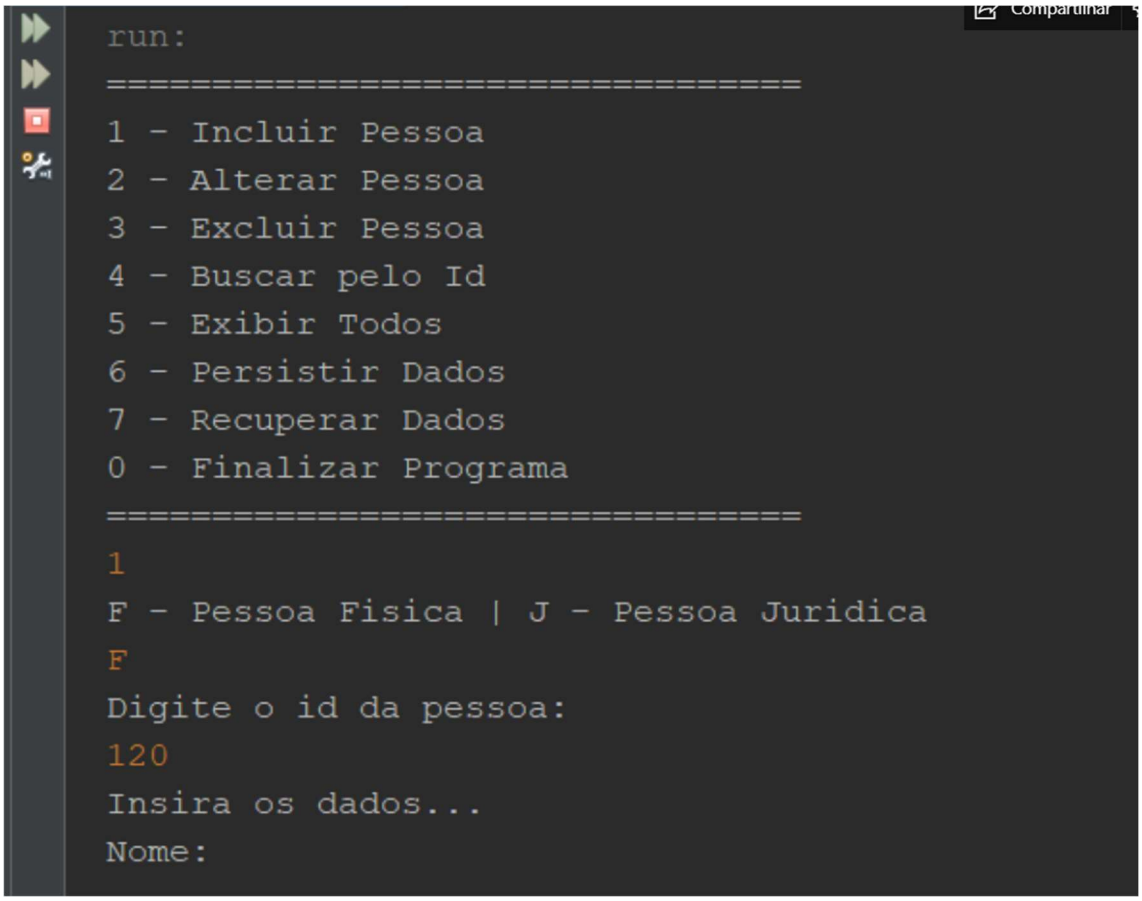
```
                                    System.out.println("
Opcao invalida");
                                    break;
                            }

                        } while
(!escolha.equalsIgnoreCase("M"));
                        break;

                    // Persistir/Salvar
                    case "6":
                        System.out.println("Escolha o
nome do arquivo");
                        escolha = scan.next();
                        scan.nextLine();
                        try {
                            pfRepo.persistir(escolha+".f
isica.bin");
                            pjRepo.persistir(escolha+".j
uridica.bin");
                        } catch (IOException erro) {
                            System.out.println("Erro ao
persistir/salvar os dados: " + erro.getMessage());
                        }
                        break;

                    //Recuperar/Carregar
                    case "7":
                        System.out.println("Informe o
nome do arquivo salvo");
                        escolha = scan.next();
                        scan.nextLine();
                        try {
                            pfRepo.recuperar(escolha+".f
isica.bin");
                            pjRepo.recuperar(escolha+".j
uridica.bin");
```

```
                    } catch (ClassNotFoundException
| IOException erro) {
                        System.out.println("Erro ao
recuperar os dados: " + erro.getMessage());
                    }
                    break;

                case "0":
                    System.out.println("Sistema
Finalizado com sucesso.");
                    break;

                default:
                    System.out.println("Opcao
invalida");
                    break;
            }
        } while (!escolha.equals("0"));
        scan.close();
    }
}
```

Resultado da Execução 01:



RESULTADO EXECUÇÃO 02:



ANÁLISE E CONCLUSÃO

PROCEDIMENTO 01:

1. **Quais as vantagens e desvantagens do uso de herança?**
   Vantagens:
   - A herança permite que você reutilize código existente, evitando duplicação.
   - Classes derivadas podem herdar métodos e atributos de classes base.
   - Ajuda organizar código de forma hierárquica.

   Desvantagens:
   - A herança pode aumentar o acomplamento entre classes.
   - Pode adicionar complexidade ao design do sistema.
   - Alteração na classe base pode ter efeitos colaterais inesperados nas classes derivadas.

2. **Por que a interface Serializable é necessária ao efetuar persistência em arquivos binários?**
   - Essencial para a persistência de objetos em arquivos binários em Java porque ela permite que os objetos sejam convertidos em uma sequência de bytes, um processo conhecido como serialização.

3. **Como o paradigma funcional é utilizado pela API stream no Java?**
   - A API Stream é usada para manipular Collections de uma forma mais eficiente, utilizando funções. Ela possibilita uma interação sobre estas coleções de objetos e a cada elemento realizar alguma ação.

4. **Quando trabalhamos com Java, qual padrão de desenvolvimento é adotado na persistência de dados em arquivos?**
   - Em Java, a persistência de dados em arquivos pode ser realizada de várias maneiras, dependendo das necessidades específicas do projeto. Neste projeto foi usada a classe ObjectOutputStream para escrever objetos em um arquivo [prefixo].fisica.bin e [prefixo].juridica.bin e a classe ObjectInputStream para ler objetos dos mesmos arquivos.

PROCEDIMENTO 02:

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

    - Elementos estáticos são aqueles que pertencem à classe em si, e não a instâncias específicas da classe. Isso significa que eles podem ser acessados sem a necessidade de criar um objeto da classe.

2. **Para que serve a classe Scanner?**

    - A classe Scanner fornece métodos para analisar dados de entrada em diferentes tipos primitivos, como inteiros, números de ponto flutuante, strings e muito mais.

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

    - Serviram para gerenciar , centralizar e organizar as atividades de inserir, excluir , alterar , localizar, recuperar e salvar os dados de pessoa física ou jurídica.