RELATÓRIO DISCENTE DE ACOMPANHAMENTO

Campus Polo Cavalhada – Porto Alegre

DESENVOLVIMENTO FULL STACK

Missão Prática | Nível 1 | Mundo 3

RPG0014 – Iniciando o caminho pelo Java

Turma: 9003

Semestre Letivo: 3º - 2024.1

GUILHERME BERNARDES BASTOS

Repositório:

CADASTROPOO

Objetivo da Prática

Criar um sistema cadastral em Java, utilizando a programação orientada a objetos, herança, polimorfismo, persistência de objetos em arquivos binários, utilizar o controle de exceções e implementar uma interface cadastral em modo texto.

1º Procedimento

CÓDIGOS

```java
package casdastropoo;

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

/**
 * @author guilhermebernardes
 */

public class CasdastroPOO {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        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("DocPessoaFisica.bin");
            System.out.println("Dados de Pessoas Físicas armazenados.");
        } catch (IOException e) {
            System.out.println("ERRO");
        }

        PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

```

```java
        try {
            System.out.println("Dados de Pessoa Física recuperados.");
            repo2.recuperar("DocPessoaFisica.bin");
            repo2.obterTodos()
                    .forEach(pessoaFisica -> {
                        pessoaFisica.exibirInformacoes();
                        System.out.println();
                    });
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("ERRO");
        }

        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("DocPessoaJuridica.bin");
            System.out.println("Dados de Pessoas Jurídicas armazenados.");
        } catch (IOException e) {
            System.out.println("ERRO");
        }

        PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

        try {
            System.out.println("Dados das Pessoas Jurídicas recuperados.");
            repo4.recuperar("DocPessoaJuridica.bin");
            repo4.obterTodos()
                    .forEach(pessoaJuridica -> {
                        pessoaJuridica.exibirInformacoes();
                        System.out.println();
```

```java
                    });
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("ERROR");
        }
    }
}
```

```java
package model;
import java.io.Serializable;
/**
 * @author guilhermebernardes
 */
public class Pessoa implements Serializable {
    private static final long serialVersionUID = 1L;

    private int id;
    private String nome;

    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 exibirInformacoes() {
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
    }
}
```

```java
package model;
import java.io.Serializable;
/**
 * @author guilhermebernardes
 */
public class PessoaFisica extends Pessoa implements Serializable {
    private static final long serialVersionUID = 1L;

    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 exibirInformacoes() {
        super.exibirInformacoes();
        System.out.println("CPF: " + cpf);
        System.out.println("Idade: " + idade);
    }
}
```

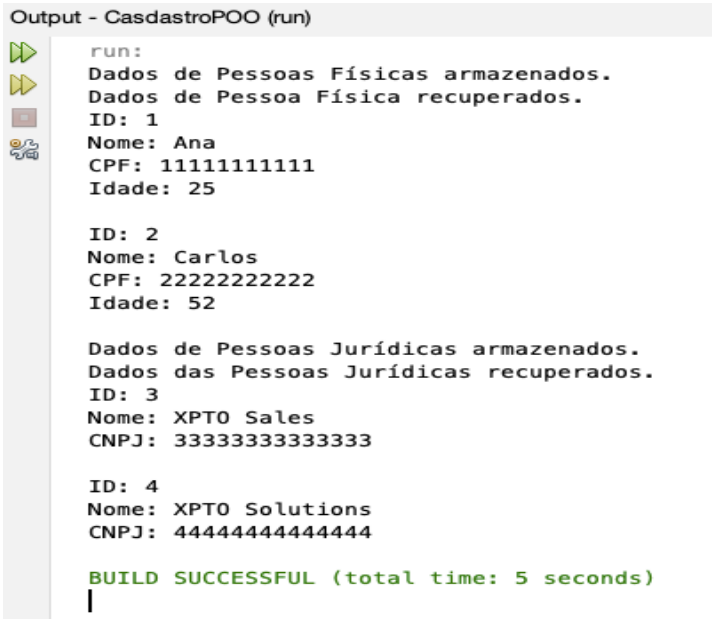
```java
package model;
import java.io.Serializable;
/**
 * @author guilhermebernardes
 */
public class PessoaJuridica extends Pessoa  implements Serializable {
    private static final long serialVersionUID = 1L;

    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 exibirInformacoes() {
        super.exibirInformacoes();
        System.out.println("CNPJ: " + cnpj);
    }
}
```

```java
package model.gerenciadores;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;
import model.PessoaFisica;
/**
 * @author guilhermebernardes
 */
public class PessoaFisicaRepo {
    private List<PessoaFisica> pessoasFisicas;

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

    public void inserir(PessoaFisica pessoaFisica) {
        pessoasFisicas.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) {
        pessoasFisicas.removeIf(pessoaFisica -> pessoaFisica.getId() == id);
    }

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

```java

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

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

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

    public Iterable<PessoaFisica> exibirInformacoes() {
        throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

```java
package model.gerenciadores;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;
import model.PessoaFisica;
import model.PessoaJuridica;
/**
 * @author guilhermebernardes
 */
public class PessoaJuridicaRepo {
    private List<PessoaJuridica> pessoasJuridicas;

    public PessoaJuridicaRepo() {
        this.pessoasJuridicas = new ArrayList<>();
    }

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

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

    public void excluir(int id) {
        pessoasJuridicas.removeIf(pessoaJuridica -> pessoaJuridica.getId() == id);
    }

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

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

    public void persistir(String DocPessoaJuridica) throws IOException {
        try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(DocPessoaJuridica))) {
            outputStream.writeObject(pessoasJuridicas);
        }
    }

    public void recuperar(String DocPessoaJuridica) throws IOException, ClassNotFoundException  {
        try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(DocPessoaJuridica))) {
            pessoasJuridicas = (List<PessoaJuridica>) inputStream.readObject();
        }
    }
}
```

RESULTADO



2º Procedimento

CÓDIGOS

```java
package casdastropoo;

import java.io.IOException;
import java.util.List;
import java.util.Scanner;
import model.PessoaFisica;
import model.PessoaJuridica;
import model.gerenciadores.PessoaFisicaRepo;
import model.gerenciadores.PessoaJuridicaRepo;
/**
 * @author guilhermebernardes
 */
public class CasdastroPOO {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        Scanner scanner = new Scanner(System.in);
        PessoaFisicaRepo pessoaFisicaRepo = new PessoaFisicaRepo();
        PessoaJuridicaRepo pessoaJuridicaRepo = new PessoaJuridicaRepo();

        try {
            pessoaFisicaRepo.recuperar("pessoaFisica.bin");
            pessoaJuridicaRepo.recuperar("pessoaJuridica.bin");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println();
        }

        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 - Buscar pelo Id");
            System.out.println("5 - Exibir Todos");
            System.out.println("6 - Persistir Dados");
            System.out.println("7 - Recuperar Dados");
            System.out.println("0 - Finalizar Programa");
            System.out.println("==================================");

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

            String tipo = "";

            switch (opcao) {
                case 1:
                    while (!tipo.equals("F") && !tipo.equals("J")) {
                        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
                        tipo = scanner.nextLine().toUpperCase();

                        if (!tipo.equals("F") && !tipo.equals("J")) {
                            System.out.println("Opção inválida. Por favor, escolha F ou J.");
                        }
                    }

                    System.out.println("Digite o id da pessoa: ");
                    int id = scanner.nextInt();
                    scanner.nextLine();

                    System.out.println("Insira os dados...");
                    System.out.println("Nome: ");
                    String nome = scanner.nextLine();

                    if (tipo.equals("F")) {
                        System.out.println("CPF: ");
                        String cpf = scanner.nextLine();

                        System.out.println("Idade: ");
                        int idade = scanner.nextInt();

                        pessoaFisicaRepo.inserir(new PessoaFisica(id, nome, cpf, idade));

                        try {
                            pessoaFisicaRepo.persistir("pessoaFisica.bin");
                        } catch (IOException e) {
                            System.out.println("ERRO: " + e.getMessage());
                        }

                    } else if (tipo.equals("J")) {
                        System.out.println("CNPJ: ");
                        String cnpj = scanner.nextLine();

                        pessoaJuridicaRepo.inserir(new PessoaJuridica(id, nome, cnpj));

                        try {
                            pessoaJuridicaRepo.persistir("pessoaJuridica.bin");
```

```
                    } catch (IOException e) {
                        System.out.println("ERRO: " + e.getMessage());
                    }

                } else {
                    System.out.println("DADO INVÁLIDO");
                }
            break;

            case 2:
                while (!tipo.equals("F") && !tipo.equals("J")) {
                    System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
                    tipo = scanner.nextLine().toUpperCase();

                    if (!tipo.equals("F") && !tipo.equals("J")) {
                        System.out.println("Opção inválida. Por favor, escolha F ou J.");
                    }
                }

                System.out.println("ID que será alterado: ");
                int alterarId = scanner.nextInt();
                scanner.nextLine();

                if (tipo.equals("F")) {
                    var pessoaFisica = pessoaFisicaRepo.obter(alterarId);
                    while (pessoaFisica == null) {
                        System.out.println("Nenhum dado encontrado com esse ID. Digite novamente: ");
                        alterarId = scanner.nextInt();
                        scanner.nextLine();
                        pessoaFisica = pessoaFisicaRepo.obter(alterarId);
                    }
                    if (pessoaFisica != null) {
                        System.out.println("Dados atuais: ");
                        System.out.println("Nome: " + pessoaFisica.getNome());
                        System.out.println("CPF: " + pessoaFisica.getCpf());
                        System.out.println("Idade: " + pessoaFisica.getIdade());

                        System.out.println("Novos dados: ");
                        System.out.println("Nome: ");
                        String novoNome = scanner.nextLine();
                        System.out.println("CPF: ");
                        String novoCpf = scanner.nextLine();
```

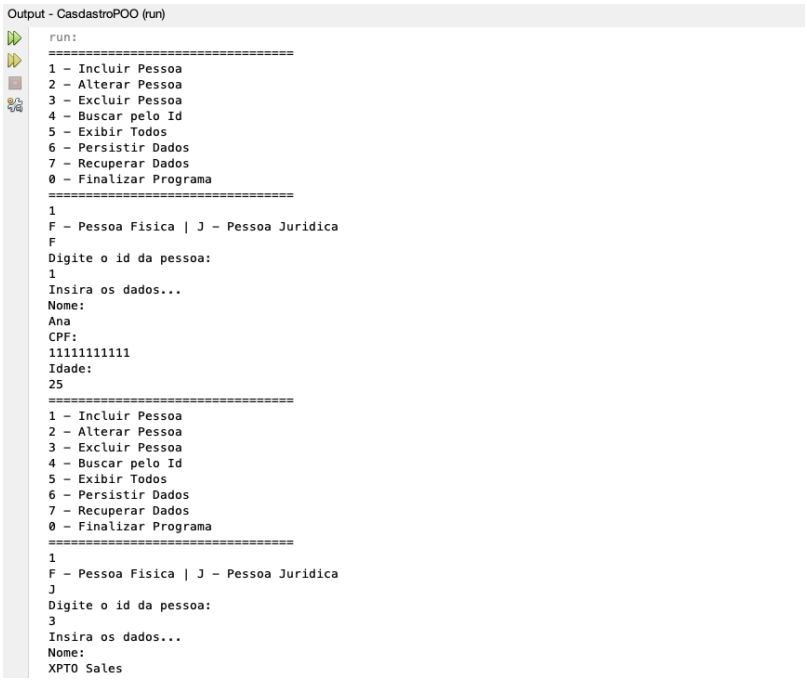
```
                        System.out.println("Idade: ");
                        int novaIdade = scanner.nextInt();
                        scanner.nextLine();

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

                        System.out.println("Dados alterados com sucesso.");
                    } else {
                        System.out.println("Nenhum dado encontrado com esse ID.");
                    }

                    try {
                        pessoaFisicaRepo.persistir("pessoaFisica.bin");
                    } catch (IOException e) {
                        System.out.println("ERRO: " + e.getMessage());
                    }

                } else if (tipo.equals("J")) {
                    PessoaJuridica pessoaJuridica = pessoaJuridicaRepo.obter(alterarId);
                    while (pessoaJuridica == null) {
                        System.out.println("Nenhum dado encontrado com esse ID. Digite novamente: ");
                        alterarId = scanner.nextInt();
                        scanner.nextLine();
                        pessoaJuridica = pessoaJuridicaRepo.obter(alterarId);
                    }
                    if (pessoaJuridica != null) {
                        System.out.println("Dados atuais: ");
                        System.out.println("Nome: " + pessoaJuridica.getNome());
                        System.out.println("CNPJ: " + pessoaJuridica.getCnpj());

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

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

```
                    System.out.println("Dados alterados com sucesso.");
                } else {
                    System.out.println("Nenhum dado encontrado com esse ID.");
                }

                try {
                    pessoaJuridicaRepo.persistir("pessoaJuridica.bin");
                } catch (IOException e) {
                    System.out.println("ERRO: " + e.getMessage());
                }
            } else {
                System.out.println("DADO INVÁLIDO");
            }
        break;
        case 3:
            while (!tipo.equals("F") && !tipo.equals("J")) {
                System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
                tipo = scanner.nextLine().toUpperCase();

                if (!tipo.equals("F") && !tipo.equals("J")) {
                    System.out.println("Opção inválida. Por favor, escolha F ou J.");
                }
            }

            System.out.println("ID que será excluido: ");
            int excluirId = scanner.nextInt();
            scanner.nextLine();

            if (tipo.equals("F")) {
                if (pessoaFisicaRepo.obter(excluirId) != null) {
                    pessoaFisicaRepo.excluir(excluirId);
                    System.out.println("Pessoa Fisica excluída.");
                } else {
                    System.out.println("Nenhum ID de Pessoa Física encontrado.");
                }
            } else if (tipo.equals("J")) {
                if (pessoaJuridicaRepo.obter(excluirId) != null) {
                    pessoaJuridicaRepo.excluir(excluirId);
                    System.out.println("Pessoa Jurídica excluída.");
                } else {
                    System.out.println("Nenhum ID de Pessoa Jurídica encontrado.2");
                }
```

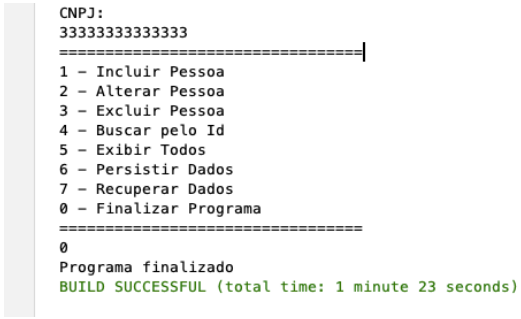
```
            } else {
                System.out.println("DADO INVÁLIDO");
            }
        break;
        case 4:
            while (!tipo.equals("F") && !tipo.equals("J")) {
                System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
                tipo = scanner.nextLine().toUpperCase();

                if (!tipo.equals("F") && !tipo.equals("J")) {
                    System.out.println("Opção inválida. Por favor, escolha F ou J.");
                }
            }

            System.out.println("ID que será visualizado: ");
            int obterId = scanner.nextInt();
            scanner.nextLine();

            if (tipo.equals("F")) {
                PessoaFisica pessoaFisica = pessoaFisicaRepo.obter(obterId);
                if (pessoaFisica != null) {
                    System.out.println("Nome: " + pessoaFisica.getNome());
                    System.out.println("CPF: " + pessoaFisica.getCpf());
                    System.out.println("Idade: " + pessoaFisica.getIdade());
                } else {
                    System.out.println("Nenhum dado encontrado por esse ID.");
                }
            } else if (tipo.equals("J")) {
                PessoaJuridica pessoaJuridica = pessoaJuridicaRepo.obter(obterId);
                if (pessoaJuridica != null) {
                    System.out.println("Nome: " + pessoaJuridica.getNome());
                    System.out.println("CNPJ: " + pessoaJuridica.getCnpj());
                } else {
                    System.out.println("Nenhum dado encontrado por esse ID.");
                }
            } else {
                System.out.println("DADO INVÁLIDO");
            }
        break;
        case 5:
            while (!tipo.equals("F") && !tipo.equals("J")) {
                System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
```

```
                    tipo = scanner.nextLine().toUpperCase();

                    if (!tipo.equals("F") && !tipo.equals("J")) {
                        System.out.println("Opção inválida. Por favor, escolha F ou J.");
                    }
                }

                if (tipo.equals("F")) {
                    List<PessoaFisica> pessoasFisicas = pessoaFisicaRepo.obterTodos();
                    if(!pessoasFisicas.isEmpty()) {
                        System.out.println("PESSOAS FÍSICAS: ");
                        for (PessoaFisica pessoaFisica : pessoaFisicaRepo.obterTodos()) {
                            System.out.println("ID: " + pessoaFisica.getId());
                            System.out.println("Nome: " + pessoaFisica.getNome());
                            System.out.println("CPF: " + pessoaFisica.getCpf());
                            System.out.println("Idade: " + pessoaFisica.getIdade());
                            System.out.println();
                        }
                    } else {
                        System.out.println("Nenhum dado encontrado de pessoas físicas.");
                    }
                } else if (tipo.equals("J")) {
                    List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaRepo.obterTodos();
                    if (!pessoasJuridicas.isEmpty()) {
                        System.out.println("PESSOAS JURÍDICAS: ");
                        for (PessoaJuridica pessoaJuridica : pessoaJuridicaRepo.obterTodos()) {
                            System.out.println("ID: " + pessoaJuridica.getId());
                            System.out.println("Nome: " + pessoaJuridica.getNome());
                            System.out.println("CNPJ: " + pessoaJuridica.getCnpj());
                            System.out.println();
                        }
                    } else {
                        System.out.println("Nenhum dado encontrado de pessoas jurídicas.");
                    }
                } else {
                    System.out.println("DADO INVÁLIDO");
                }
            break;
            case 6:
                System.out.println("Digite o nome do arquivo a ser salvo: ");
                String prefixo = scanner.nextLine();

                try {
                    pessoaFisicaRepo.persistir(prefixo + "fisica.bin");
                    pessoaJuridicaRepo.persistir(prefixo + "juridica.bin");
                    System.out.println("DADOS SALVOS COM SUCESSO");
                } catch (IOException e) {
                    System.out.println("ERRO: " + e.getMessage());
                }
            break;
            case 7:
                System.out.println("Digite o nome do arquivo a ser recuperado: ");
                String prefixoRecuperado = scanner.nextLine();

                try {
                    pessoaFisicaRepo.recuperar(prefixoRecuperado + "fisica.bin");
                    pessoaJuridicaRepo.recuperar(prefixoRecuperado + "juridica.bin");
                    System.out.println("DADOS RECUPERADOS COM SUCESSO");
                } catch (IOException | ClassNotFoundException e) {
                    System.out.println("ERRO: " + e.getMessage());
                }
            break;
            case 0:
                System.out.println("Programa finalizado");
                return;
            default:
                System.out.println("Opção Inválida");
        }

    }
}
}
```

RESULTADO



```
Output - CasdastroPOO (run)
run:
================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
================================
1
F - Pessoa Fisica | J - Pessoa Juridica
F
Digite o id da pessoa:
1
Insira os dados...
Nome:
Ana
CPF:
11111111111
Idade:
25
================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Persistir Dados
7 - Recuperar Dados
0 - Finalizar Programa
================================
1
F - Pessoa Fisica | J - Pessoa Juridica
J
Digite o id da pessoa:
3
Insira os dados...
Nome:
XPTO Sales
```



```
CNPJ:
33333333333333
================================
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
Programa finalizado
BUILD SUCCESSFUL (total time: 1 minute 23 seconds)
```

ANÁLISE E CONCLUSÃO

Procedimento 1

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

Uma das vantagens mais significativas vejo que seria a reutilização do código em si, com a herança podemos criar classes que herdam métodos e atributos de outras, não precisando reescrever tudo novamente. Também permite alterar o comportamento de classes para adicionar novos métodos em uma subclasse. A manutenção do condigo, quando temos que alterar algo, com a herança, alteramos na classe base e assim todas as outras subclasses herdarão essas mudanças.

A principal desvantagem do uso de herança se dá na rigidez do código, no qual onde queremos alterar apenas um detalhe na classe, todas as outras subclasses serão alteradas e passiveis de erros. A complexidade de hierarquia pode se tornar difícil de entender e resultar em possíveis problemas.

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

Ela é necessária porque converte o código em uma sequência de bites em um arquivo para ser transmitido em rede.

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

Ela uso o paradigma funcional que permite operações de transformação e agregação de dados em coleções de uma forma mais declarativa.

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

O padrão mais comum usado é o DAO (Data Access Object).

Procedimento 2

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

São métodos ou variáveis que pertencem a própria classe ao invés de instâncias. O motivo do método main adotar esse modificador é que assim ele pode ser chamado diretamente para execução do programa sem a necessidade de depender de uma instância.

b. Para que serve a classe Scanner?

Serve para ler entradas no programa.

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

Ela impactou diretamente na manutenção do código, separando cada responsabilidade, reutilização de códigos e acesso a dados.