



Estácio

Campus: Polo Centro II - Guarulhos - SP

Curso: Desenvolvimento full stack

Disciplina: RPG0014 - Iniciando o caminho pelo Java

Turma: 2024.3

Aluno: Pedro Wilson Araújo Avilar

- **Título da prática**

Missão prática | Mundo 3 | Nível 1

- **Material de apoio**

<https://sway.cloud.microsoft/s/9HQUpWu7S6bzXsqt/embed>

- **Repositório git**

<https://github.com/PedroAvilar/CadastroPOO>

- **Objetivo da prática**

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

- **IDE utilizada**

Apache NetBeans IDE 22.

- **1º Procedimento | Criação das Entidades e Sistema de Persistência**

Entidade Pessoa:

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

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

    //Construtor padrão
    public Pessoa() {}

    //Construtor completo
```

```

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

//Getters e setters
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.println("Id: " + id);
    System.out.println("Nome: " + nome);
}
}

```

Entidade PessoaFisica:

```

package model;
import java.io.Serializable;

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

    //Construtor padrão
    public PessoaFisica() {
        super();
    }

    //Construtor completo
    public PessoaFisica(int id, String nome, String cpf, int idade) {
        super(id, nome);
    }
}

```

```

        this.cpf = cpf;
        this.idade = idade;
    }

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

```

Entidade PessoaJuridica:

```

package model;
import java.io.Serializable;

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

    //Construtor padrão
    public PessoaJuridica() {
        super();
    }

    //Construtor completo
    public PessoaJuridica(int id, String nome, String cnpj) {
        super(id, nome);
    }
}

```

```

        this.cnpj = cnpj;
    }

    //Getter e setter
    public String getCnpj() {
        return cnpj;
    }
    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

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

```

Gerenciador PessoaFisicaRepo:

```

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

public class PessoaFisicaRepo {
    //Lista de PessoaFisica
    private ArrayList<PessoaFisica> pessoasFisicas = new ArrayList<>();

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

    //Método alterar
    public void alterar(PessoaFisica pessoa) {
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            if (pessoasFisicas.get(i).getId() == pessoa.getId()) {
                pessoasFisicas.set(i, pessoa);
                return;
            }
        }
    }
}

```

```

//Método excluir
public void excluir(int id) {
    pessoasFisicas.removeIf(pessoa -> pessoa.getId() == id);
}

//Método obter
public PessoaFisica obter(int id) {
    for (PessoaFisica pessoa : pessoasFisicas) {
        if (pessoa.getId() == id) {
            return pessoa;
        }
    }
    return null;
}

//Método obter todas
public ArrayList<PessoaFisica> obterTodos() {
    return new ArrayList<>(pessoasFisicas);
}

//Método persistir
public void persistir(String arquivo) throws IOException {
    try (ObjectOutputStream arquivoSaida = new
ObjectOutputStream(new FileOutputStream(arquivo))) {
        arquivoSaida.writeObject(pessoasFisicas);
        System.out.println("Dados de Pessoa Fisica Armazenados.");
    }
}

//Método recuperar
public void recuperar(String arquivo) throws IOException,
ClassNotFoundException {
    try (ObjectInputStream arquivoEntrada = new
ObjectInputStream(new FileInputStream(arquivo))) {
        pessoasFisicas = (ArrayList<PessoaFisica>)
arquivoEntrada.readObject();
        System.out.println("Dados de Pessoa Fisica Recuperados.");
    }
}
}

```

Gerenciador PessoaJuridicaRepo:

```
package model;
```

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

public class PessoaJuridicaRepo {
    //Lista de PessoaJuridica
    private ArrayList<PessoaJuridica> pessoasJuridicas = new
ArrayList<>();

    //Método inserir
    public void inserir(PessoaJuridica pessoa) {
        pessoasJuridicas.add(pessoa);
    }

    //Método alterar
    public void alterar(PessoaJuridica pessoa) {
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            if (pessoasJuridicas.get(i).getId() == pessoa.getId()) {
                pessoasJuridicas.set(i, pessoa);
                return;
            }
        }
    }

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

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

    //Método obterTodos
    public ArrayList<PessoaJuridica> obterTodos() {
        return new ArrayList<>(pessoasJuridicas);
    }
}
```

```

        //Método persistir
        public void persistir(String arquivo) throws IOException {
            try (ObjectOutputStream arquivoSaida = new
ObjectOutputStream(new FileOutputStream(arquivo))) {
                arquivoSaida.writeObject(pessoasJuridicas);
                System.out.println("Dados de Pessoa Juridica
Armazenados.");
            }
        }

        //Método recuperar
        public void recuperar(String arquivo) throws IOException,
ClassNotFoundException {
            try (ObjectInputStream arquivoEntrada = new
ObjectInputStream(new FileInputStream(arquivo))) {
                pessoasJuridicas = (ArrayList<PessoaJuridica>)
arquivoEntrada.readObject();
                System.out.println("Dados de Pessoa Juridica
Recuperados.");
            }
        }
    }
}

```

Classe principal CadastroPOO:

```

package cadastrapoo;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

public class CadastroPOO {

    public static void main(String[] args) {
        //Instancia de pessoas físicas - repo1
        PessoaFisicaRepo repo1 = new PessoaFisicaRepo();

        //Pessoas físicas com construtor completo
        PessoaFisica pessoaFisica1 = new PessoaFisica(1, "Ana",
"11111111111", 25);
        PessoaFisica pessoaFisica2 = new PessoaFisica(2, "Carlos",
"22222222222", 52);
        repo1.inserir(pessoaFisica1);
        repo1.inserir(pessoaFisica2);
    }
}

```

```
//Persistência em repo1
try {
    repo1.persistir("ListaPessoasFisicas.dat");
} catch (Exception e) {
    e.printStackTrace();
}

//Instancia de pesssoas físicas - repo2
PessoaFisicaRepo repo2 = new PessoaFisicaRepo();

//Recuperação e exibição de pessoas físicas em repo2
try {
    repo2.recuperar("ListaPessoasFisicas.dat");
    repo2.obterTodos().forEach(pessoaFisica -> {
        pessoaFisica.exibir();
    });
} catch (Exception e) {
    e.printStackTrace();
}

//Instancia de pessoas jurídicas - repo3
PessoaJuridicaRepo repo3 = new PessoaJuridicaRepo();

//Pessoas jurídicas com construtor completo
PessoaJuridica pessoaJuridica1 = new PessoaJuridica(3, "XPTO
Sales", "3333333333333333");
PessoaJuridica pessoaJuridica2 = new PessoaJuridica(4, "XPTO
Solutions", "4444444444444444");
repo3.inserir(pessoaJuridica1);
repo3.inserir(pessoaJuridica2);

//Persistência em repo 3
try {
    repo3.persistir("ListaPessoasJurídicas.dat");
} catch (Exception e) {
    e.printStackTrace();
}

//Intancia de pessoasjurídicas - repo4
PessoaJuridicaRepo repo4 = new PessoaJuridicaRepo();

//Recuperação e exibição de pessoas jurídicas em repo4
```

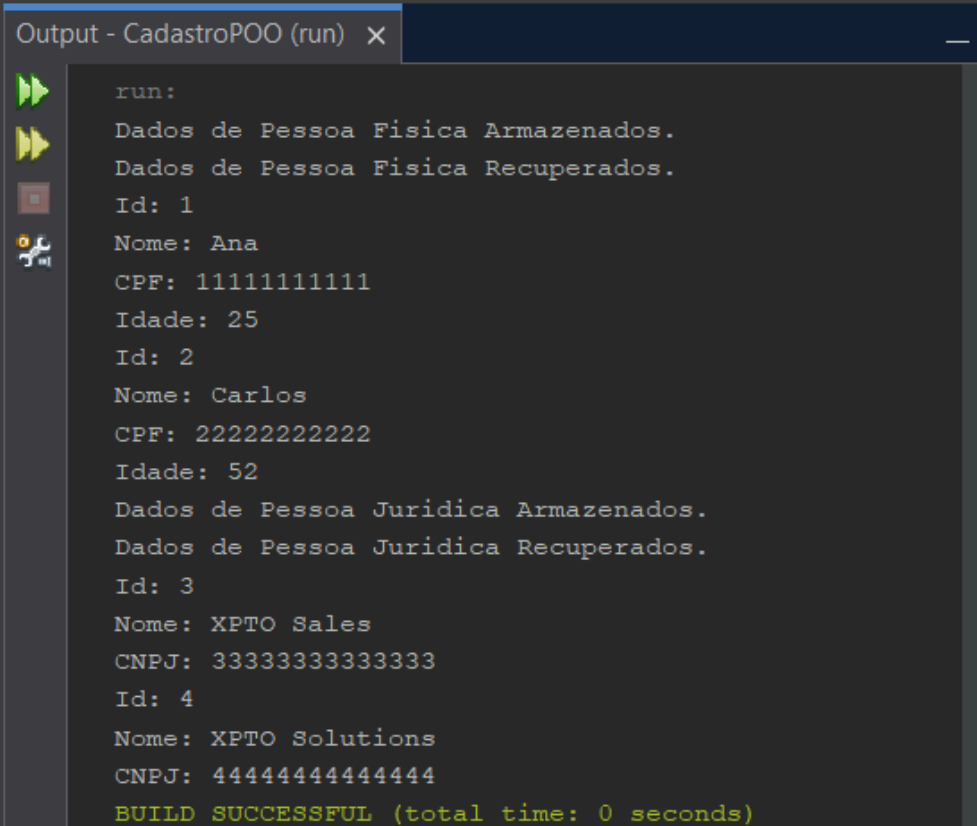


```

    try {
        repo4.recuperar("ListaPessoasJurídicas.dat");
        repo4.obterTodos().forEach(pessoaJuridica -> {
            pessoaJuridica.exibir();
        });
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

- **Resultados da execução dos códigos - procedimento 1:**



```

run:
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Fisica Recuperados.
Id: 1
Nome: Ana
CPF: 111111111111
Idade: 25
Id: 2
Nome: Carlos
CPF: 222222222222
Idade: 52
Dados de Pessoa Juridica Armazenados.
Dados de Pessoa Juridica Recuperados.
Id: 3
Nome: XPTO Sales
CNPJ: 33333333333333
Id: 4
Nome: XPTO Solutions
CNPJ: 4444444444444444
BUILD SUCCESSFUL (total time: 0 seconds)

```

- **Análise e conclusão do procedimento 1**

- **Quais as vantagens e desvantagens do uso de herança?**

Vantagens:

Como principais pontos de vantagens no uso de herança temos a reutilização de código, já que subclasses podem reutilizar o código de uma superclasse, reaproveitando métodos, organização lógica, modelando uma relação entre as entidades e facilidade de manutenção, alterações na superclasse podem automaticamente serem refletidas nas subclasses.

Desvantagens:

Para desvantagens temos, por exemplo, acoplamento forte, podendo deixar o sistema menos flexível e mais difícil de modificar, pois as subclasses ficam fortemente acopladas à superclasse, restrição de herança múltipla, sendo uma limitação para o Java não permitir a herança múltipla, podendo até deixar o código mais complexo, deixando muitos níveis de hierarquia.

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

Sendo implementado em uma classe, marca como capaz de ser convertida em um fluxo de bytes, assim, podendo ser gravado em um arquivo binário para serem persistidos e restaurados, sem implementar a interface Serializable, o Java não permite que um objeto seja convertido em um fluxo de bytes e gera uma exceção.

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

A API Stream do Java adota conceitos do paradigma funcional para tornar o processamento de coleções mais declarativo e conciso, permitindo operações funcionais sobre coleções.

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

O padrão de desenvolvimento DAO (Data Access Object), onde as classes precisam implementar a interface Serializable, para persistir e recuperar dados em arquivos binários, temos classes que gerenciam a interação com os arquivos, usando ObjectOutputStream para gravar objetos e ObjectInputStream para recuperar os objetos.

- **2º Procedimento | Criação do Cadastro em Modo Texto**

Classe principal CadastroPOO:

```
package cadastrapoo;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;
import java.util.Scanner;
import java.util.InputMismatchException;

public class CadastroPOO {

    private static Scanner scanner = new Scanner(System.in);

    //Método para tratamento de exceção de leitura de números
    private static int lerInt(String mensagem) {
        int valor = -1;
```

```

        while (valor == -1) {
            System.out.print(mensagem);
            try {
                valor = scanner.nextInt();
                scanner.nextLine();
            } catch (InputMismatchException e) {
                System.out.println("Entrada invalida. Insira um numero
inteiro.");
                scanner.nextLine();
            }
        }
        return valor;
    }

    public static void main(String[] args) {
        //Instâncias
        PessoaFisicaRepo repoFisica = new PessoaFisicaRepo();
        PessoaJuridicaRepo repoJuridica = new PessoaJuridicaRepo();

        //Laço que mantém o programa em execução
        while (true) {
            //Opções do programa
            int opcao = -1;
            while (opcao == -1) {

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 - Exibir 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.println("=====");

                opcao = lerInt("Digite o numero da opcao: ");
            }

            //Finalizar o programa
            if (opcao == 0) {
                System.out.println("Programa finalizado.");
                break;
            }
        }
    }
}

```

```

    }

    switch (opcao) {
        //Incluir
        case 1:
            System.out.println("F - Pessoa Fisica | J - Pessoa
Juridica");

            String tipoIncluir = scanner.nextLine();
            if (tipoIncluir.equalsIgnoreCase("F")) { //Pessoa
Física

                System.out.println("Insira os dados da pessoa
fisica.");

                //Capturando e inserindo pessoa física
                int id = lerInt("ID: ");
                System.out.print("Nome: ");
                String nome = scanner.nextLine();
                System.out.print("CPF: ");
                String cpf = scanner.nextLine();
                int idade = lerInt("Idade: ");
                PessoaFisica pessoaFisica = new
PessoaFisica(id, nome, cpf, idade);
                repoFisica.inserir(pessoaFisica);
                System.out.println("Pessoa fisica incluida com
sucesso.");

            } else if (tipoIncluir.equalsIgnoreCase("J")) {
//Pessoa Jurídica

                System.out.println("Insira os dados da pessoa
juridica.");

                //Capturando e inserindo pessoa jurídica
                int id = lerInt("ID: ");
                System.out.print("Nome: ");
                String nome = scanner.nextLine();
                System.out.print("CNPJ: ");
                String cnpj = scanner.nextLine();
                PessoaJuridica pessoaJuridica = new
PessoaJuridica(id, nome, cnpj);
                repoJuridica.inserir(pessoaJuridica);
                System.out.println("Pessoa juridica incluida
com sucesso.");

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

```

```

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

            String tipoAlterar = scanner.nextLine();
            if (tipoAlterar.equalsIgnoreCase("F")) { //Pessoa
Física

                int id = lerInt("Digite o ID da pessoa fisica
para alterar: ");

                PessoaFisica pessoaAtual =
repoFisica.obter(id);

                //Exibe os dados atuais e exige os novos
                if (pessoaAtual != null) {
                    System.out.println("Dados atuais: ");
                    pessoaAtual.exibir();
                    System.out.println("Insira os novos
dados.");

                    System.out.print("Novo nome: ");
                    String nome = scanner.nextLine();
                    System.out.print("Novo CPF: ");
                    String cpf = scanner.nextLine();
                    int idade = lerInt("Nova idade: ");
                    PessoaFisica pessoaNova = new
PessoaFisica(id, nome, cpf, idade);
                    repoFisica.alterar(pessoaNova);
                    System.out.println("Pessoa fisica alterada
com sucesso.");

                } else {
                    System.out.println("Nao foi encontrada uma
pessoa fisica com o ID " + id);
                }

            } else if (tipoAlterar.equalsIgnoreCase("J")) {
//Pessoa Jurídica

                int id = lerInt("Digite o ID da pessoa juridica
para alterar: ");

                PessoaJuridica empresaAtual =
repoJuridica.obter(id);

                //Exibe os dados atuais e exige os novos
                if (empresaAtual != null) {
                    System.out.println("Dados atuais: ");
                    empresaAtual.exibir();

```

```

        System.out.println("Insira os novos
dados.");

        System.out.print("Novo nome: ");
        String nome = scanner.nextLine();
        System.out.print("Novo CNPJ: ");
        String cnpj = scanner.nextLine();
        PessoaJuridica empresaNova = new
PessoaJuridica(id, nome, cnpj);
        repoJuridica.alterar(empresaNova);
        System.out.println("Pessoa juridica
alterada com sucesso.");
    } else {
        System.out.println("Nao foi encontrada uma
pessoa juridica com o ID " + id);
    }
} else {
    System.out.println("Opcao invalida.");
}
break;

//Excluir
case 3:
    System.out.println("F - Pessoa Fisica | J - Pessoa
Juridica");

    String tipoExcluir = scanner.nextLine();
    if (tipoExcluir.equalsIgnoreCase("F")) { //Pessoa
Fisica

        //Excluindo pessoa física pelo ID
        int id = lerInt("Digite o ID da pessoa fisica
para excluir: ");

        repoFisica.excluir(id);
        System.out.println("Pessoa fisica excluida com
sucesso.");

    } else if (tipoExcluir.equalsIgnoreCase("J")) {
//Pessoa Juridica

        //Excluindo pessoa jurídica pelo ID
        int id = lerInt("Digite o ID da pessoa juridica
para excluir: ");

        repoJuridica.excluir(id);
        System.out.println("Pessoa juridica excluida
com sucesso.");

    } else {
        System.out.println("Opcao invalida.");
    }
}
}

```

```

    }
    break;

    //Buscar pelo ID
    case 4:
        System.out.println("F - Pessoa Fisica | J - Pessoa
Juridica");

        String tipoBuscar = scanner.nextLine();
        if (tipoBuscar.equalsIgnoreCase("F")) { //Pessoa
Física

            //Exibindo pessoa fisica pelo ID
            int id = lerInt("Digite o ID da pessoa fisica
para exibir: ");

            PessoaFisica pessoa = repoFisica.obter(id);
            if (pessoa != null) {
                pessoa.exibir();
            } else {
                System.out.println("Nao foi encontrada uma
pessoa fisica com o ID " + id);
            }
        } else if (tipoBuscar.equalsIgnoreCase("J")) {
//Pessoa Jurídica

            //Exibindo pessoa jurídica pelo ID
            int id = lerInt("Digite o ID da pessoa juridica
para exibir: ");

            PessoaJuridica empresa =
repoJuridica.obter(id);

            if (empresa != null) {
                empresa.exibir();
            } else {
                System.out.println("Nao foi encontrada uma
pessoa juridica com o ID " + id);
            }
        } else {
            System.out.println("Opcao invalida.");
        }
        break;

        //Exibir todos
        case 5:
            System.out.println("F - Pessoa Fisica | J - Pessoa
Juridica");

            String tipoExibirTodos = scanner.nextLine();

```

```

        if (tipoExibirTodos.equalsIgnoreCase("F")) {
//Pessoas Físicas

repoFisica.obterTodos().forEach(PessoaFisica::exibir);
        } else if (tipoExibirTodos.equalsIgnoreCase("J")) {
//Pessoas Jurídicas

repoJuridica.obterTodos().forEach(PessoaJuridica::exibir);
        } else {
            System.out.println("Opcao invalida.");
        }
        break;

//Salvar dados
case 6:
    System.out.print("Digite o prefixo dos arquivos
para salvar: ");
    String prefixoSalvar = scanner.nextLine();
    try {
        repoFisica.persistir(prefixoSalvar +
".fisica.bin");
        repoJuridica.persistir(prefixoSalvar +
".juridica.bin");
        System.out.println("Dados salvos com
sucesso.");
    } catch (Exception e) {
        System.out.println("Erro ao salvar os dados: "
+ e.getMessage());
    }
    break;

//Recuperar dados
case 7:
    System.out.print("Digite o prefixo dos arquivos
para recuperar: ");
    String prefixoRecuperar = scanner.nextLine();
    try {
        repoFisica.recuperar(prefixoRecuperar +
".fisica.bin");
        repoJuridica.recuperar(prefixoRecuperar +
".juridica.bin");
        System.out.println("Dados recuperados com
sucesso.");
    }

```



```

        } catch (Exception e) {
            System.out.println("Erro ao recuperar os dados:
" + e.getMessage());
        }
        break;

        default:
            System.out.println("Opcao invalida. Tente
novamente.");
        }
    }
    scanner.close();
}
}

```

- Resultados da execução dos códigos - procedimento 2:
 - Incluindo pessoa física:

```

run:
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Exibir pelo ID
5 - Exibir Todos
6 - Salvar Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
Digite o numero da opcao: 1
F - Pessoa Fisica | J - Pessoa Juridica
F
Insira os dados da pessoa fisica.
ID: 1
Nome: Pedro Wilson Araujo Avilar
CPF: 58911522970
Idade: 28
Pessoa fisica incluida com sucesso.

```

- Incluindo pessoa jurídica:

```

=====
Digite o numero da opcao: 1
F - Pessoa Fisica | J - Pessoa Juridica
J
Insira os dados da pessoa juridica.
ID: 6
Nome: Will Avilar
CNPJ: 25498756325485
Pessoa juridica incluida com sucesso.

```

- **Alterando pessoa física:**

```
=====
Digite o numero da opcao: 2
F - Pessoa Fisica | J - Pessoa Juridica
F
Digite o ID da pessoa fisica para alterar: 1
Dados atuais:
Id: 1
Nome: Pedro Wilson Araujo Avilar
CPF: 58911522970
Idade: 28
Insira os novos dados.
Novo nome: Pedro Wilson Araujo Avilar
Novo CPF: 12345678910
Nova idade: 30
Pessoa fisica alterada com sucesso.
```

- **Alterando pessoa jurídica:**

```
=====
Digite o numero da opcao: 2
F - Pessoa Fisica | J - Pessoa Juridica
J
Digite o ID da pessoa juridica para alterar: 6
Dados atuais:
Id: 6
Nome: Will Avilar
CNPJ: 25498756325485
Insira os novos dados.
Novo nome: Will Araujo Avilar
Novo CNPJ: 98765432198765
Pessoa juridica alterada com sucesso.
```

- **Excluindo pessoa física:**

```
=====
Digite o numero da opcao: 3
F - Pessoa Fisica | J - Pessoa Juridica
F
Digite o ID da pessoa fisica para excluir: 2
Pessoa fisica excluida com sucesso.
```

- **Excluindo pessoa jurídica:**

```
=====
Digite o numero da opcao: 3
F - Pessoa Fisica | J - Pessoa Juridica
J
Digite o ID da pessoa juridica para excluir: 7
Pessoa juridica excluida com sucesso.
```

- **Exibindo pessoa física:**

```
=====
Digite o numero da opcao: 4
F - Pessoa Fisica | J - Pessoa Juridica
F
Digite o ID da pessoa fisica para exibir: 2
Id: 2
Nome: Claudio Pontes dos Santos
CPF: 45612378910
Idade: 40
```

- **Exibindo pessoa jurídica:**

```
=====
Digite o numero da opcao: 4
F - Pessoa Fisica | J - Pessoa Juridica
J
Digite o ID da pessoa juridica para exibir: 7
Id: 7
Nome: Pet Claudio
CNPJ: 78532415940265
```

- **Exibindo todas as pessoas físicas:**

```
=====
Digite o numero da opcao: 5
F - Pessoa Fisica | J - Pessoa Juridica
F
Id: 1
Nome: Pedro Wilson Araujo Avilar
CPF: 12345678910
Idade: 30
Id: 2
Nome: Claudio Pontes dos Santos
CPF: 45612378910
Idade: 40
```

- **Exibindo todas as pessoas jurídicas:**

```
=====
Digite o numero da opcao: 5
F - Pessoa Fisica | J - Pessoa Juridica
J
Id: 6
Nome: Will Araujo Avilar
CNPJ: 98765432198765
Id: 7
Nome: Pet Claudio
CNPJ: 78532415940265
```

- **Salvando dados:**

```
=====
Digite o numero da opcao: 6
Digite o prefixo dos arquivos para salvar: Clientes
Dados de Pessoa Fisica Armazenados.
Dados de Pessoa Juridica Armazenados.
Dados salvos com sucesso.
```

- **Recuperando dados:**

```
=====
Digite o numero da opcao: 7
Digite o prefixo dos arquivos para recuperar: Clientes
Dados de Pessoa Fisica Recuperados.
Dados de Pessoa Juridica Recuperados.
Dados recuperados com sucesso.
```

- **Tratando exceções:**

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Exibir pelo ID
5 - Exibir Todos
6 - Salvar Dados
7 - Recuperar Dados
0 - Finalizar Programa
=====
Digite o numero da opcao: h
Entrada invalida. Insira um numero inteiro.
Digite o numero da opcao: 1
F - Pessoa Fisica | J - Pessoa Juridica
F
Insira os dados da pessoa fisica.
ID: e
Entrada invalida. Insira um numero inteiro.
ID: 3
Nome: Vanessa
CPF: 458132987410
Idade: w
Entrada invalida. Insira um numero inteiro.
Idade: 30
Pessoa fisica incluida com sucesso.
```

- **Finalizando o programa:**

```
=====
Digite o numero da opcao: 0
Programa finalizado.
BUILD SUCCESSFUL (total time: 23 minutes 30 seconds)
```

- **Análise e conclusão do procedimento 2**

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

Em Java, elementos estáticos são associados à classe em si, e não a instâncias específicas dessa classe, podendo assim, serem acessados diretamente usando o nome da classe, sem necessidade de criar um objeto. O motivo para o método main ser estático é por ser o ponto de partida de um programa Java, sendo chamado diretamente pela JVM. se o método main não for estático, a JVM teria que instanciar a classe para conseguir chamá-lo.

- **Para que serve a classe Scanner?**

A classe Scanner é do pacote java.util, sendo usada para ler a entrada de diferentes tipos de dados pelo usuário no console, tornando possível a captura e modificações de dados com interação entre o usuário e o sistema.

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

O uso de classes de repositórios impactou a organização e manutenção do código, deixando as classes gerenciadoras de objetos atuarem centralizando as operações, como inserção, alteração, exclusão, busca e persistência. Também encapsulam a lógica de armazenamento e recuperação dos objetos, criando uma camada de abstração, separando a lógica de negócio no método main, ficando concentrada na interação com o usuário.