



FACULDADE ESTACIO
CURSO TECNÓLOGO EM DESENVOLVIMENTO FULL STACK

RAFAEL VALVERDE FONSECA

MISSÃO PRÁTICA – NÍVEL 1 – MUNDO 3
Iniciando o caminho pelo Java

Serrinha - BA
2024

RAFAEL VALVERDE FONSECA

MISSÃO PRÁTICA – NÍVEL 1 – MUNDO 3

Iniciando o caminho pelo Java

Trabalho apresentado à disciplina Iniciando o caminho pelo Java do Curso Tecnólogo em Desenvolvimento Full Stack, período 2024.4 Flex, como requisito parcial do relatório de acompanhamento.

Tutoria: Maria Manso

Serrinha – BA

2024

OBJETIVOS

Este relatório apresenta a composição do trabalho proposto para o Nível 1: Iniciando o cainho pelo Java o qual está contido no semestre letivo no período 2024, o qual apresenta todos os códigos solicitados, resultados da execução desses códigos e descrição de avaliação sobre o tema abordado, respondendo perguntas propostas pelo tutor.

Palavras-chave: Java, herança, interface Serializable, classes, objetos.

SUMÁRIO

1	CÓDIGOS	9
2	RESULTADOS DE EXECUÇÃO DE CÓDIGOS.....	18
3	ANÁLISE	19

1 CÓDIGOS

```
// CadastroPOO
package cadastrapoo;

import model.*;

import java.util.Scanner;

public class CadastroPOO {

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

            int opcao = -1;

            while (opcao != 0) {
                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("=====");
                System.out.print("Escolha uma opcao: ");

                try {
                    opcao = Integer.parseInt(scanner.nextLine());

                    switch (opcao) {
                        case 1: // Incluir
                            incluir(scanner, repoFisica, repoJuridica);
                            break;

                        case 2: // Alterar
                            alterar(scanner, repoFisica, repoJuridica);
                            break;

                        case 3: // Excluir
                            excluir(scanner, repoFisica, repoJuridica);
                            break;

                        case 4: // Exibir pelo ID
                            exibirPorId(scanner, repoFisica, repoJuridica);
                            break;

                        case 5: // Exibir todos
                            exibirTodos(scanner, repoFisica, repoJuridica);
                            break;

                        case 6: // Salvar dados
                            salvar(scanner, repoFisica, repoJuridica);
                            break;

                        case 7: // Recuperar dados
                            recuperar(scanner, repoFisica, repoJuridica);
                            break;
                    }
                }
            }
        }
    }
}
```

```

        case 0: // Sair
            System.out.println("Encerrando o sistema...");
            break;

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

    } catch (NumberFormatException e) {
        System.out.println("Entrada invalida. Por favor, insira um numero.");
    }
}
}

private static void incluir(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo
repoJuridica) {
    System.out.print("F - Pessoa Fisica | J - Pessoa Juridica: ");
    String tipo = scanner.nextLine().toUpperCase();

    if (tipo.equals("F")) {
        System.out.print("ID: ");
        int id = Integer.parseInt(scanner.nextLine());
        System.out.print("Nome: ");
        String nome = scanner.nextLine();
        System.out.print("CPF: ");
        String cpf = scanner.nextLine();
        System.out.print("Idade: ");
        int idade = Integer.parseInt(scanner.nextLine());

        repoFisica.inserir(new PessoaFisica(id, nome, cpf, idade));
        System.out.println("Pessoa Fisica incluida com sucesso!");

    } else if (tipo.equals("J")) {
        System.out.print("ID: ");
        int id = Integer.parseInt(scanner.nextLine());
        System.out.print("Nome: ");
        String nome = scanner.nextLine();
        System.out.print("CNPJ: ");
        String cnpj = scanner.nextLine();

        repoJuridica.inserir(new PessoaJuridica(id, nome, cnpj));
        System.out.println("Pessoa Juridica incluida com sucesso!");

    } else {
        System.out.println("Tipo invalido!");
    }
}

private static void alterar(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo
repoJuridica) {
    System.out.print("F - Pessoa Fisica | J - Pessoa Juridica: ");
    String tipo = scanner.nextLine().toUpperCase();

    if (tipo.equals("F")) {
        System.out.print("ID da pessoa a ser alterada: ");
        int id = Integer.parseInt(scanner.nextLine());
        PessoaFisica pessoa = repoFisica.obter(id);

        if (pessoa != null) {
            System.out.println("Dados atuais: " + pessoa);
        }
    }
}

```

```

        System.out.print("Novo Nome: ");
        String nome = scanner.nextLine();
        System.out.print("Novo CPF: ");
        String cpf = scanner.nextLine();
        System.out.print("Nova Idade: ");
        int idade = Integer.parseInt(scanner.nextLine());

        repoFisica.alterar(new PessoaFisica(id, nome, cpf, idade));
        System.out.println("Pessoa Fisica alterada com sucesso!");
    } else {
        System.out.println("Pessoa nao encontrada.");
    }

} else if (tipo.equals("J")) {
    System.out.print("ID da pessoa a ser alterada: ");
    int id = Integer.parseInt(scanner.nextLine());
    PessoaJuridica pessoa = repoJuridica.obter(id);

    if (pessoa != null) {
        System.out.println("Dados atuais: " + pessoa);
        System.out.print("Novo Nome: ");
        String nome = scanner.nextLine();
        System.out.print("Novo CNPJ: ");
        String cnpj = scanner.nextLine();

        repoJuridica.alterar(new PessoaJuridica(id, nome, cnpj));
        System.out.println("Pessoa Juridica alterada com sucesso!");
    } else {
        System.out.println("Pessoa nao encontrada.");
    }
} else {
    System.out.println("Tipo invalido!");
}
}

private static void excluir(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo
repoJuridica) {
    System.out.print("F - Pessoa Fisica | J - Pessoa Juridica: ");
    String tipo = scanner.nextLine().toUpperCase();

    if (tipo.equals("F")) {
        System.out.print("ID da pessoa a ser excluida: ");
        int id = Integer.parseInt(scanner.nextLine());
        repoFisica.excluir(id);
        System.out.println("Pessoa Fisica excluida com sucesso!");
    } else if (tipo.equals("J")) {
        System.out.print("ID da pessoa a ser excluida: ");
        int id = Integer.parseInt(scanner.nextLine());
        repoJuridica.excluir(id);
        System.out.println("Pessoa Juridica excluída com sucesso!");
    } else {
        System.out.println("Tipo invalido!");
    }
}

private static void exibirPorId(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo
repoJuridica) {
    System.out.print("F - Pessoa Fisica | J - Pessoa Juridica: ");
    String tipo = scanner.nextLine().toUpperCase();

```

```

        if (tipo.equals("F")) {
            System.out.print("ID da pessoa: ");
            int id = Integer.parseInt(scanner.nextLine());
            PessoaFisica pessoa = repoFisica.obter(id);

            if (pessoa != null) {
                System.out.println("Dados: " + pessoa);
            } else {
                System.out.println("Pessoa nao encontrada.");
            }
        }

        } else if (tipo.equals("J")) {
            System.out.print("ID da pessoa: ");
            int id = Integer.parseInt(scanner.nextLine());
            PessoaJuridica pessoa = repoJuridica.obter(id);

            if (pessoa != null) {
                System.out.println("Dados: " + pessoa);
            } else {
                System.out.println("Pessoa nao encontrada.");
            }
        } else {
            System.out.println("Tipo invalido!");
        }
    }

    private static void exibirTodos(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
        System.out.print("F - Pessoa Fisica | J - Pessoa Juridica: ");
        String tipo = scanner.nextLine().toUpperCase();

        if (tipo.equals("F")) {
            repoFisica.obterTodos().forEach(System.out::println);
        } else if (tipo.equals("J")) {
            repoJuridica.obterTodos().forEach(System.out::println);
        } else {
            System.out.println("Tipo invalido!");
        }
    }

    private static void salvar(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
        System.out.print("Digite o prefixo do arquivo: ");
        String prefixo = scanner.nextLine();

        try {
            repoFisica.persistir(prefixo + ".fisica.bin");
            repoJuridica.persistir(prefixo + ".juridica.bin");
            System.out.println("Dados salvos com sucesso!");
        } catch (Exception e) {
            System.out.println("Erro ao salvar os dados: " + e.getMessage());
        }
    }

    private static void recuperar(Scanner scanner, PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica) {
        System.out.print("Digite o prefixo do arquivo: ");
        String prefixo = scanner.nextLine();

        try {

```



```

        repoFisica.recuperar(prefixo + ".fisica.bin");
        repoJuridica.recuperar(prefixo + ".juridica.bin");
        System.out.println("Dados recuperados com sucesso!");
    } catch (Exception e) {
        System.out.println("Erro ao recuperar os dados: " + e.getMessage());
    }
}
}
}

```

```

// Pessoa
package model;

```

```

import java.io.Serializable;

```

```

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

```

```

    // Construtor padrão
    public Pessoa() {}

```

```

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

```

```

    // Método exibir
    public void exibir() {
        System.out.println("ID: " + id + ", 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;
    }
}

```

```
// PessoaFisica
package model;

import java.io.Serializable;

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

    // Construtor padrão
    public PessoaFisica() {}

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

    // Método exibir (polimórfico)
    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf + ", 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;
    }
}
```

```
// PessoaJuridica
package model;

import java.io.Serializable;

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

    // Construtor padrão
    public PessoaJuridica() {}

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

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

    // Getters e Setters
    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }
}
```

```

// PessoaFisicaRepo
package model;

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

public class PessoaFisicaRepo {
    private List<PessoaFisica> lista = new ArrayList<>();

    // Método para inserir uma nova PessoaFisica
    public void inserir(PessoaFisica pessoaFisica) {
        lista.add(pessoaFisica);
    }

    // Método para alterar uma PessoaFisica existente
    public void alterar(PessoaFisica pessoaFisica) {
        for (int i = 0; i < lista.size(); i++) {
            if (lista.get(i).getId() == pessoaFisica.getId()) {
                lista.set(i, pessoaFisica);
                return;
            }
        }
    }

    // Método para excluir uma PessoaFisica pelo ID
    public void excluir(int id) {
        lista.removeIf(pessoa -> pessoa.getId() == id);
    }

    // Método para obter uma PessoaFisica pelo ID
    public PessoaFisica obter(int id) {
        for (PessoaFisica pessoa : lista) {
            if (pessoa.getId() == id) {
                return pessoa;
            }
        }
        return null;
    }

    // Método para obter todas as PessoasFisicas
    public List<PessoaFisica> obterTodos() {
        return new ArrayList<>(lista);
    }

    // Método para persistir os dados em arquivo
    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
        FileOutputStream(nomeArquivo))) {
            oos.writeObject(lista);
        }
    }

    // Método para recuperar os dados de um arquivo
    @SuppressWarnings("unchecked")
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            lista = (List<PessoaFisica>) ois.readObject();
        }
    }
}

```

```

// PessoaJuridicaRepo
package model;

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

public class PessoaJuridicaRepo {
    private List<PessoaJuridica> lista = new ArrayList<>();

    // Método para inserir uma nova PessoaJuridica
    public void inserir(PessoaJuridica pessoaJuridica) {
        lista.add(pessoaJuridica);
    }

    // Método para alterar uma PessoaJuridica existente
    public void alterar(PessoaJuridica pessoaJuridica) {
        for (int i = 0; i < lista.size(); i++) {
            if (lista.get(i).getId() == pessoaJuridica.getId()) {
                lista.set(i, pessoaJuridica);
                return;
            }
        }
    }

    // Método para excluir uma PessoaJuridica pelo ID
    public void excluir(int id) {
        lista.removeIf(pessoa -> pessoa.getId() == id);
    }

    // Método para obter uma PessoaJuridica pelo ID
    public PessoaJuridica obter(int id) {
        for (PessoaJuridica pessoa : lista) {
            if (pessoa.getId() == id) {
                return pessoa;
            }
        }
        return null;
    }

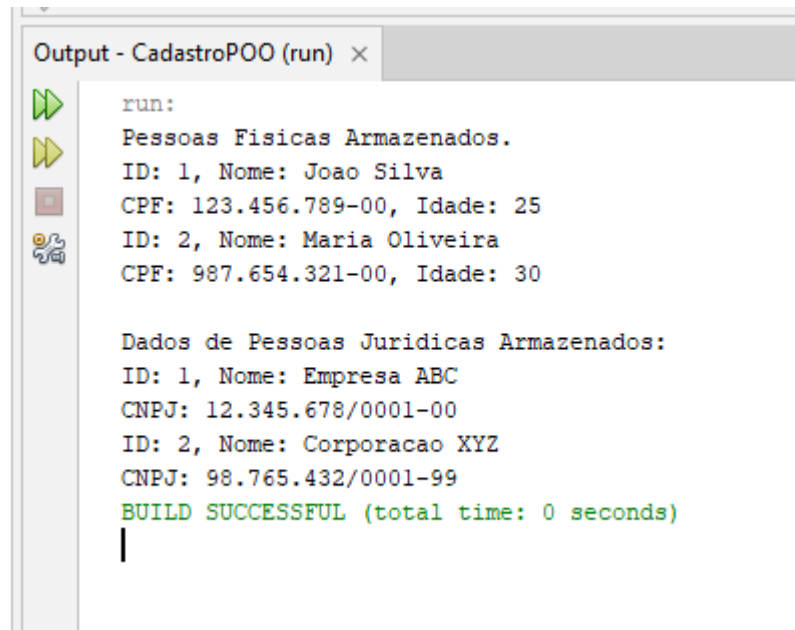
    // Método para obter todas as PessoasJuridicas
    public List<PessoaJuridica> obterTodos() {
        return new ArrayList<>(lista);
    }

    // Método para persistir os dados em arquivo
    public void persistir(String nomeArquivo) throws IOException {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
        FileOutputStream(nomeArquivo))) {
            oos.writeObject(lista);
        }
    }

    // Método para recuperar os dados de um arquivo
    @SuppressWarnings("unchecked")
    public void recuperar(String nomeArquivo) throws IOException, ClassNotFoundException {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(nomeArquivo))) {
            lista = (List<PessoaJuridica>) ois.readObject();
        }
    }
}

```

2 RESULTADOS DE EXECUÇÃO DE CÓDIGOS



```
run:
Pessoas Fisicas Armazenados.
ID: 1, Nome: Joao Silva
CPF: 123.456.789-00, Idade: 25
ID: 2, Nome: Maria Oliveira
CPF: 987.654.321-00, Idade: 30

Dados de Pessoas Juridicas Armazenados:
ID: 1, Nome: Empresa ABC
CNPJ: 12.345.678/0001-00
ID: 2, Nome: Corporacao XYZ
CNPJ: 98.765.432/0001-99
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

3 ANÁLISE

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

Em Java, elementos estáticos são atributos, métodos, blocos de código e classes internas que pertencem à classe, ou seja, esses elementos podem ser acessados sem a necessidade de criar um objeto da classe.

O método main em Java é o ponto de entrada de um programa. Ele deve ser estático por dois motivos principais: acesso sem instanciar a classes.

Para que serve a classe Scanner?

A classe Scanner em Java é utilizada para ler entradas do usuário. Ela faz parte do pacote java.util e permite ler dados de diferentes fontes, como o teclado, arquivos ou até mesmo strings. O Scanner facilita a captura de dados em tempo real, como números, strings e outros tipos básicos, e converte-os para os tipos apropriados.

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

O uso de classes de repositório tem um impacto significativo na organização e manutenção do código. Quem tem persistência de dados, como nesse sistema de cadastro, as classes de repositório ajudam a separar responsabilidades e promovem boas práticas. Alguns dos impactos positivos do uso dessas classes: facilidade na manutenção, reusabilidade, facilidade de testes, código mais organizado.