UNIVERSIDADE ESTÁCIO DE SÁ

Relatório de Atividade Prática

IDELMAR FERNANDO DE SOUZA

SANTA ROSA
2024

Atividade Prática 2 - ProjetoPOO

Objetivo da Prática

Desenvolver um pequeno projeto com orientação a objeto com a linguagem Java e em linha de comando, com menu de opções para cadastro de pessoa física e pessoa jurídica.

2. Códigos utilizados:

Classe CadastroPOO:

```
package cadastropoo;
import model.Pessoa;
import model.PessoaFisica;
import model.PessoaFisicaRepo;
import model.PessoaJuridica;
import model.PessoaJuridicaRepo;

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

public class CadastroPOO {

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

      do {
         exibirMenu();

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

         switch (opcao) {
            case 1:
               incluirPessoa(repoFisica, repoJuridica, scanner);
               break;
```

```
            case 2:
                alterarPessoa(repoFisica, repoJuridica, scanner);
                break;
            case 3:
                excluirPessoa(repoFisica, repoJuridica, scanner);
                break;
            case 4:
                buscarPorId(repoFisica, repoJuridica, scanner);
                break;
            case 5:
                exibirPessoa(repoFisica, repoJuridica, scanner);
                break;
            case 6:
                salvarPessoa(repoFisica, repoJuridica, scanner);
                break;
            case 7:
                recuperarPessoa(repoFisica, repoJuridica, scanner);
                break;
            case 0:
                System.out.println("Encerrando o programa...");
                break;
            default:
                System.out.println("Opção inválida. Tente novamente.");
                break;
        }
    } while (opcao != 0);

    scanner.close();
}

private static void exibirMenu() {
    System.out.println("=======================");
    System.out.println("     Menu de Opções     ");
```

```
        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 - Salvar Dados");
        System.out.println("7 - Recuperar Dados");
        System.out.println("0 - Finalizar Programa");
        System.out.println("========================");
        System.out.print("Escolha uma opção: ");
    }

    /**
     * @param repoFisica
     * @param repoJuridica
     * @param scanner
     */
    private static void incluirPessoa(PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica,
Scanner scanner) {
        System.out.println("Escolha o tipo de pessoa:");
        System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
        String tipoPessoa = scanner.nextLine().toUpperCase();

        switch (tipoPessoa) {
            case "F":
                incluirPessoaFisica(repoFisica, scanner);
                break;
            case "J":
                incluirPessoaJuridica(repoJuridica, scanner);
                break;
            default:
                System.out.println("Opção inválida.");
```

```
            break;

        }
    }

    private static void incluirPessoaFisica(PessoaFisicaRepo repo, Scanner scanner) {
        int id = 0;
        boolean idValido = false;
        while (!idValido) {
            System.out.print("ID: ");
            if (scanner.hasNextInt()) {
                id = scanner.nextInt();
                idValido = true;
            } else {
                System.out.println("Por favor, insira um número inteiro para o ID.");
                scanner.next();
            }
        }
        scanner.nextLine();
        System.out.print("Nome: ");
        String nome = scanner.nextLine().toUpperCase();
        System.out.print("CPF: ");
        String cpf = scanner.nextLine();
        System.out.print("Idade: ");
        int idade = scanner.nextInt();
        scanner.nextLine();
        PessoaFisica pessoa = new PessoaFisica(id, nome, cpf, idade);
        repo.inserir(pessoa);
        System.out.println("Pessoa Física incluída com sucesso!");
    }

    private static void incluirPessoaJuridica(PessoaJuridicaRepo repo, Scanner scanner) {
        System.out.println("ID:");
```

```
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Nome:");
        String nome = scanner.nextLine();
        System.out.print("CNPJ:");
        String cnpj = scanner.nextLine();
        PessoaJuridica pessoaJuridica = new PessoaJuridica(id, nome, cnpj);
        repo.inserir(pessoaJuridica);
        System.out.println("Pessoa Jurídica incluída com sucesso!");
    }

    private static void alterarPessoa(PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica,
Scanner scanner) {
        System.out.println("Escolha o tipo de pessoa para alterar:");
        System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
        String tipoPessoa = scanner.nextLine().toUpperCase();

        System.out.print("ID:");
        int id = scanner.nextInt();

        if (tipoPessoa.equals("F")) {
            PessoaFisica pessoaFisica = repoFisica.obter(id);
            if (pessoaFisica != null) {
                exibirPessoa(pessoaFisica);

                System.out.println("Insira os 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("Pessoa Física alterada com sucesso!");
        } else {
            System.out.println("Pessoa Física não encontrada com o ID informado.");
        }
    } else if (tipoPessoa.equals("J")) {
        PessoaJuridica pessoaJuridica = repoJuridica.obter(id);
        if (pessoaJuridica != null) {
            exibirPessoa(pessoaJuridica);

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

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

            System.out.println("Pessoa Jurídica alterada com sucesso!");
        } else {
            System.out.println("Pessoa Jurídica não encontrada com o ID informado.");
        }
    } else {
        System.out.println("Tipo de pessoa inválido.");
    }
}

private static void exibirPessoa(Pessoa pessoa) {
```

```
        System.out.println("ID: " + pessoa.getId());
        System.out.println("Nome: " + pessoa.getNome());
        if (pessoa instanceof PessoaFisica) {
            PessoaFisica pessoaFisica = (PessoaFisica) pessoa;
            System.out.println("CPF: " + pessoaFisica.getCpf());
            System.out.println("Idade: " + pessoaFisica.getIdade());
        } else if (pessoa instanceof PessoaJuridica) {
            PessoaJuridica pessoaJuridica = (PessoaJuridica) pessoa;
            System.out.println("CNPJ: " + pessoaJuridica.getCnpj());
        }
    }

    private static void excluirPessoa(PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica,
Scanner scanner) {
        System.out.println("Escolha o tipo de pessoa:");
        System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
        String tipoPessoa = scanner.nextLine().toUpperCase();

        switch (tipoPessoa) {
            case "F":
                System.out.print("Digite o ID da Pessoa Física a ser excluída: ");
                int idPF = scanner.nextInt();

                PessoaFisica pessoaFisica = repoFisica.obter(idPF);

                if (pessoaFisica != null) {
                    exibirPessoa(pessoaFisica);
                    repoFisica.excluir(idPF);
                    System.out.println("Pessoa física excluída com sucesso!");
                } else {
                    System.out.println("ID não encontrado. Operação de exclusão cancelada.");
                }
```

```
            case "J":
                System.out.print("Digite o ID da Pessoa Jurídica a ser excluída: ");
                int idPJ = scanner.nextInt();

                PessoaJuridica pessoaJuridica = repoJuridica.obter(idPJ);

                if (pessoaJuridica != null) {
                    exibirPessoa(pessoaJuridica);
                    repoJuridica.excluir(idPJ);
                    System.out.println("Pessoa jurídica excluída com sucesso!");
                } else {
                    System.out.println("ID não encontrado. Operação de exclusão cancelada.");
                }
            default:
                System.out.println("Opção inválida.");
                break;

        }
    }

    private static void buscarPorId(PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica,
Scanner scanner) {
        System.out.print("Digite o id: ");
        int id = scanner.nextInt();
        PessoaFisica pessoaFisica = repoFisica.obter(id);
        if (pessoaFisica != null) {
            exibirPessoa(pessoaFisica);
        } else {
            PessoaJuridica pessoaJuridica = repoJuridica.obter(id);
            if (pessoaJuridica != null) {
                exibirPessoa(pessoaJuridica);
            } else {
                System.out.println("Pessoa não encontrada.");
```

```
        }
    }
}

private static void salvarPessoa(PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica,
Scanner scanner) {
    String prefix = "pessoas";
    try {
        FileOutputStream fisicaFile = new FileOutputStream(prefix + "_fisica.bin");
        ObjectOutputStream fisicaOut = new ObjectOutputStream(fisicaFile);

        FileOutputStream juridicaFile = new FileOutputStream(prefix + "_juridica.bin");
        ObjectOutputStream juridicaOut = new ObjectOutputStream(juridicaFile);

        List<PessoaFisica> pessoasFisicas = repoFisica.obterTodos();
        for (PessoaFisica pessoa : pessoasFisicas) {
            fisicaOut.writeObject(pessoa);
        }

        List<PessoaJuridica> pessoasJuridicas = repoJuridica.obterTodos();
        for (PessoaJuridica pessoa : pessoasJuridicas) {
            juridicaOut.writeObject(pessoa);
        }

        fisicaOut.close();
        juridicaOut.close();

        System.out.println("Dados salvos com sucesso!");

    } catch (IOException e) {
        System.err.println("Erro ao salvar os dados: " + e.getMessage());
    }
}
```

```
    private static void exibirPessoa(PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica,
Scanner scanner) {
        System.out.println("Escolha o tipo de pessoa:");
        System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
        String tipoPessoa = scanner.nextLine().toUpperCase();

        switch (tipoPessoa) {
            case "F":
                List<PessoaFisica> pessoasFisicas = repoFisica.obterTodos();
                System.out.println("Lista de Pessoas Físicas");
                System.out.println("--------------------------");
                if (pessoasFisicas.isEmpty()) {
                    System.out.println("Nenhuma pessoa física cadastrada.");
                } else {
                    for (PessoaFisica pessoa : pessoasFisicas) {
                        exibirPessoa(pessoa);
                        System.out.println("--------------------------");
                    }
                }
                break;
            case "J":
                List<PessoaJuridica> pessoasJuridicas = repoJuridica.obterTodos();
                System.out.println("Lista de Pessoas Jurídicas");
                System.out.println("--------------------------");
                if (pessoasJuridicas.isEmpty()) {
                    System.out.println("Nenhuma pessoa juridica cadastrada.");
                } else {
                    for (PessoaJuridica pessoa : pessoasJuridicas) {
                        exibirPessoa(pessoa);
                        System.out.println("--------------------------");
                    }
                }
```

```
            break;
        default:
            System.out.println("Opção inválida.");
            break;
    }
  }

  private static void recuperarPessoa(PessoaFisicaRepo repoFisica, PessoaJuridicaRepo repoJuridica,
Scanner scanner) {

    String prefix = "pessoas";
    try {
      FileInputStream fisicaFile = new FileInputStream(prefix + "_fisica.bin");
      ObjectInputStream fisicaIn = new ObjectInputStream(fisicaFile);
      FileInputStream juridicaFile = new FileInputStream(prefix + "_juridica.bin");
      ObjectInputStream juridicaIn = new ObjectInputStream(juridicaFile);

      while (fisicaFile.available() > 0) {
        PessoaFisica pessoa = (PessoaFisica) fisicaIn.readObject();
        repoFisica.inserir(pessoa);
      }
      while (juridicaFile.available() > 0) {
        PessoaJuridica pessoa = (PessoaJuridica) juridicaIn.readObject();
        repoJuridica.inserir(pessoa);
      }
      fisicaIn.close();
      juridicaIn.close();
      System.out.println("Dados recuperados com sucesso!");

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

```
}
```

Resultado da execução:
Define um scanner com switch para definições das opções chamando as funções conforme a escolha do usuário;

Classe Pessoa:

```
package model;

import java.io.Serializable;

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

}]
```

Define o objeto principal com os seus atributos primários (id, nome);
Define também getID e getNome para retornar id e nome;

Classe PessoaFisica:

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

    public PessoaFisica() {
        super(0, "");
    }

    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 int getIdade() {
        return idade;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

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

Define o primeiro objeto herdado de Pessoa, adiciona os atributos cpf e idade e também os métodos getCpf,getIdade e setCpf;

Repositório Pessoa Física:

```
public class PessoaFisicaRepo {

    private List<PessoaFisica> listaPessoas;

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

    public void inserir(PessoaFisica pessoa) {
        listaPessoas.add(pessoa);
    }

    public void alterar(int id, PessoaFisica pessoaAtualizada) {
        for (int i = 0; i < listaPessoas.size(); i++) {
            if (listaPessoas.get(i).getId() == id) {
                listaPessoas.set(i, pessoaAtualizada);
                break;
            }
        }
    }

    public void excluir(int id) {
        Iterator<PessoaFisica> iterator = listaPessoas.iterator();
        while (iterator.hasNext()) {
            PessoaFisica pessoa = iterator.next();
            if (pessoa.getId() == id) {
                iterator.remove();
                break;
            }
        }
    }

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

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

    public void persistir(String arquivo) throws IOException {
        try (FileOutputStream fileOut = new FileOutputStream(arquivo);
             ObjectOutputStream objectOut = new ObjectOutputStream(fileOut)) {
            objectOut.writeObject(listaPessoas);

        } catch (IOException e) {
            throw e;
        }
    }

    public void recuperar(String arquivo) throws IOException, ClassNotFoundException {
        try (FileInputStream fileIn = new FileInputStream(arquivo);
```

```
            ObjectInputStream objectIn = new ObjectInputStream(fileIn)) {
            listaPessoas = (List<PessoaFisica>) objectIn.readObject();

        } catch (IOException | ClassNotFoundException e) {
            throw e;
        }
    }
}
```

Adiciona um vetor com os dados da pessoa física, para manipulação dos dados possui os métodos inserir, alterar, excluir, obter e obter todos e os métodos persistir e recuperar para gravar e retornar os dados;

Classe Pessoa Jurídica:

```
public class PessoaJuridica extends Pessoa {

    private String cnpj;

    public PessoaJuridica() {
        super(0, "");
    }

    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.println("Cnpj: " + cnpj);
    }
}
```

Segunda herança da classe principal, adiciona atributo cnpj, e métodos getCnpj,setCnpj e exibir (cnpj);

Repositório Pessoa Jurídica:

```
public class PessoaJuridicaRepo {

    private List<PessoaJuridica> listaPessoaJuridica;

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

```
    public void inserir(PessoaJuridica pessoa) {
        listaPessoaJuridica.add(pessoa);
    }

    public void alterar(int id, PessoaJuridica pessoaJuridicaAtualizada) {
        for (int i = 0; i < listaPessoaJuridica.size(); i++) {
            if (listaPessoaJuridica.get(i).getId() == id) {
                listaPessoaJuridica.set(i, pessoaJuridicaAtualizada);
                break;
            }
        }
    }

    public void excluir(int id) {
        Iterator<PessoaJuridica> iterator = listaPessoaJuridica.iterator();
        while (iterator.hasNext()) {
            PessoaJuridica pessoa = iterator.next();
            if (pessoa.getId() == id) {
                iterator.remove();
                break;
            }
        }
    }

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

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

    public void persistir(String arquivo) throws IOException {
        try (FileOutputStream fileOut = new FileOutputStream(arquivo);
                ObjectOutputStream objectOut = new ObjectOutputStream(fileOut)) {
            objectOut.writeObject(listaPessoaJuridica);

        } catch (IOException e) {
            throw e;
        }
    }

    public void recuperar(String arquivo) throws IOException, ClassNotFoundException {
        try (FileInputStream fileIn = new FileInputStream(arquivo);
                ObjectInputStream objectIn = new ObjectInputStream(fileIn)) {
            listaPessoaJuridica = (List<PessoaJuridica>) objectIn.readObject();

        } catch (IOException | ClassNotFoundException e) {
            throw e;
        }
    }
}
```

Adiciona um vetor com os dados da pessoa jurídica, possui os métodos inserir, alterar, excluir, obter e obter todos e os métodos persistir e recuperar para gravar e retornar os dados;

Análise e Conclusão:

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

São elementos que pertencem a classe, métodos e variáveis estáticos são componentes que pertencem a classe e mantêm seu valor declarado e não irá pertencer a outras instâncias da classe; é uma forma segura de declarar componentes que devem ter essa caracteristicas; E são renderizados em compilação melhorando o desempenho;

Para que serve a classe Scanner?

Serve para realizar a leitura dos dados, transformando tudo em string independente do tipo declarado originalmente;

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

Usar repositórios ajuda na organização do código pois separa as funções de manipulação dos dados e isso ajuda bastante na manutenção do código.