Universidade Estácio de Sá
Campus Santa Cruz

Relatório Discente de Acompanhamento

Nome: Tamires de Souza Alves
Curso: Desenvolvimento Full Stack
Disciplina: Back-end sem banco não tem
Número da Turma: 9001
Semestre Letivo: 3º Semestre
GitHub: https://github.com/tamiresalves024/MissaoPratica_Nivel3_Mundo3.git

Criação de aplicativo Java, com acesso ao banco de dados SQL Server através do JDBC.

Objetivo da Prática

Criar um aplicativo cadastral com uso do SQL Server na persistência de dados.

Procedimento 1 – Criação das Entidades e Sistema de Persistência

Códigos do Procedimento 1

```
package cadastrobd;

import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;
import cadastro.model.util.ConectorBD;

import java.util.List;
import java.util.Scanner;

public class CadastroBD {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ConectorBD conectorBD = new ConectorBD();

        PessoaFisicaDAO pessoaFisicaDAO = new
PessoaFisicaDAO(conectorBD);
        PessoaJuridicaDAO pessoaJuridicaDAO = new
PessoaJuridicaDAO(conectorBD);

        int opcao;
        do {
            exibirMenu();
            opcao = scanner.nextInt();
            scanner.nextLine(); // Limpar o buffer

            switch (opcao) {
                case 1:
                    incluirPessoa(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    break;

                case 2:
                    alterarPessoa(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    break;
```

```
            case 3:
                excluirPessoa(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                break;

            case 4:
                exibirPessoaPorId(scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                break;

            case 5:
                exibirTodasPessoas(scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                break;

            case 0:
                System.out.println("Encerrando o programa.");
                break;

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

    } while (opcao != 0);

    scanner.close();
  }

  private static void exibirMenu() {
    System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
    System.out.println("Selecione uma opção:");
    System.out.println("1. Incluir");
    System.out.println("2. Alterar");
    System.out.println("3. Excluir");
    System.out.println("4. Exibir pelo ID");
    System.out.println("5. Exibir todos");
    System.out.println("0. Sair");
    System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
  }

  private static void incluirPessoa(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
    System.out.println("Escolha o tipo (F - Física, J - Jurídica):");
    String tipo = scanner.nextLine().toUpperCase();

    if (tipo.equalsIgnoreCase("F")) {
        PessoaFisica pessoaFisica = criarPessoaFisica(scanner);
        pessoaFisicaDAO.incluir(pessoaFisica);
```

```
            System.out.println("Pessoa Física incluída com sucesso!");
        } else if (tipo.equalsIgnoreCase("J")) {
            PessoaJuridica pessoaJuridica = criarPessoaJuridica(scanner);
            pessoaJuridicaDAO.incluir(pessoaJuridica);
            System.out.println("Pessoa Jurídica incluída com sucesso!");
        } else {
            System.out.println("Opção inválida.");
        }
    }

    private static void alterarPessoa(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        System.out.println("Escolha o tipo (F - Física, J - Jurídica):");
        String tipoAlterar = scanner.nextLine().toUpperCase();

        System.out.print("Informe o ID da pessoa: ");
        int idAlterar = scanner.nextInt();
        scanner.nextLine(); // Limpar o buffer

        if (tipoAlterar.equalsIgnoreCase("F")) {
            PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(idAlterar);
            if (pessoaFisica != null) {
                exibirDadosAtuais(pessoaFisica);
                atualizarPessoaFisica(scanner, pessoaFisica);
                pessoaFisicaDAO.alterar(pessoaFisica);
                System.out.println("Pessoa Física alterada com sucesso!");
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipoAlterar.equalsIgnoreCase("J")) {
            PessoaJuridica pessoaJuridica =
pessoaJuridicaDAO.getPessoa(idAlterar);
            if (pessoaJuridica != null) {
                exibirDadosAtuais(pessoaJuridica);
                atualizarPessoaJuridica(scanner, pessoaJuridica);
                pessoaJuridicaDAO.alterar(pessoaJuridica);
                System.out.println("Pessoa Jurídica alterada com sucesso!");
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Opção inválida.");
        }
    }

    private static void excluirPessoa(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        System.out.println("Escolha o tipo (F - Física, J - Jurídica):");
        String tipoExcluir = scanner.nextLine().toUpperCase();
```

```
        System.out.print("Informe o ID da pessoa: ");
        int idExcluir = scanner.nextInt();
        scanner.nextLine();

        if (tipoExcluir.equalsIgnoreCase("F")) {
            PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(idExcluir);
            if (pessoaFisica != null) {
                pessoaFisicaDAO.excluir(pessoaFisica);
                System.out.println("Pessoa Física excluída com sucesso!");
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipoExcluir.equalsIgnoreCase("J")) {
            PessoaJuridica pessoaJuridica =
pessoaJuridicaDAO.getPessoa(idExcluir);
            if (pessoaJuridica != null) {
                pessoaJuridicaDAO.excluir(pessoaJuridica);
                System.out.println("Pessoa Jurídica excluída com sucesso!");
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Opção inválida.");
        }
    }

    private static void exibirPessoaPorId(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        System.out.println("Escolha o tipo (F - Física, J - Jurídica):");
        String tipoExibirId = scanner.nextLine().toUpperCase();

        System.out.print("Informe o ID da pessoa: ");
        int idExibirId = scanner.nextInt();
        scanner.nextLine();

        if (tipoExibirId.equalsIgnoreCase("F")) {
            PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(idExibirId);
            if (pessoaFisica != null) {
                pessoaFisica.exibir();
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipoExibirId.equalsIgnoreCase("J")) {
            PessoaJuridica pessoaJuridica =
pessoaJuridicaDAO.getPessoa(idExibirId);
            if (pessoaJuridica != null) {
                pessoaJuridica.exibir();
            } else {
```

```
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Opção inválida.");
        }
    }

    private static void exibirTodasPessoas(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        System.out.println("Escolha o tipo (F - Física, J - Jurídica):");
        String tipoExibirTodos = scanner.nextLine().toUpperCase();

        if (tipoExibirTodos.equalsIgnoreCase("F")) {
            System.out.println("Exibindo dados de Pessoa Fisica...");
            List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
            for (PessoaFisica pf : pessoasFisicas) {
                pf.exibir();
            }
        } else if (tipoExibirTodos.equalsIgnoreCase("J")) {
            System.out.println("Exibindo dados de Pessoa Juridica...");
            List<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.getPessoas();
            for (PessoaJuridica pj : pessoasJuridicas) {
                pj.exibir();
            }
        } else {
            System.out.println("Opção inválida.");
        }
    }

    private static PessoaFisica criarPessoaFisica(Scanner scanner) {
        PessoaFisica pessoaFisica = new PessoaFisica();
        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
        System.out.print("Nome: ");
        pessoaFisica.setNome(scanner.nextLine());

        System.out.print("Logradouro: ");
        pessoaFisica.setLogradouro(scanner.nextLine());

        System.out.print("Cidade: ");
        pessoaFisica.setCidade(scanner.nextLine());

        System.out.print("Estado: ");
        pessoaFisica.setEstado(scanner.nextLine());

        System.out.print("Telefone: ");
        pessoaFisica.setTelefone(scanner.nextLine());

        System.out.print("E-mail: ");
```

```
        pessoaFisica.setEmail(scanner.nextLine());

        System.out.print("CPF: ");
        pessoaFisica.setCpf(scanner.nextLine());
        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
        return pessoaFisica;
    }

    private static PessoaJuridica criarPessoaJuridica(Scanner scanner) {
        PessoaJuridica pessoaJuridica = new PessoaJuridica();
        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
        System.out.print("Nome: ");
        pessoaJuridica.setNome(scanner.nextLine());

        System.out.print("Logradouro: ");
        pessoaJuridica.setLogradouro(scanner.nextLine());

        System.out.print("Cidade: ");
        pessoaJuridica.setCidade(scanner.nextLine());

        System.out.print("Estado: ");
        pessoaJuridica.setEstado(scanner.nextLine());

        System.out.print("Telefone: ");
        pessoaJuridica.setTelefone(scanner.nextLine());

        System.out.print("E-mail: ");
        pessoaJuridica.setEmail(scanner.nextLine());

        System.out.print("CNPJ: ");
        pessoaJuridica.setCnpj(scanner.nextLine());
        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
        return pessoaJuridica;
    }

    private static void exibirDadosAtuais(PessoaFisica pessoaFisica) {
        pessoaF

    private static void atualizarPessoaFisica(Scanner scanner, PessoaFisica
pessoaFisica) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    private static void exibirDadosAtuais(PessoaJuridica pessoaJuridica) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
```

```
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    private static void atualizarPessoaJuridica(Scanner scanner, PessoaJuridica
pessoaJuridica) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    private static void atualizarDados(Scanner scanner, PessoaJuridica
pessoaJuridica) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    private static void atualizarDados(Scanner scanner, PessoaJuridica
pessoaJuridica) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    private static class ConectorBD {

        public ConectorBD() {
        }
    }

package cadastrobd.model;

public record Pessoa(int idPessoa, String nome, String logradouro, String
cidade, String estado, String telefone, String email) {

    //const padrão
    public Pessoa {
        validarCampos();
    }

    private void validarCampos() {
        if (idPessoa < 0) {
            throw new IllegalArgumentException("ID não pode ser negativo");
        }
    }
```

```
    //método p exibir dados no console
    public void exibir() {
        System.out.println(this);
    }

    @Override
    public String toString() {
        return "Id: " + idPessoa +
                "\nNome: " + nome +
                "\nLogradouro: " + logradouro +
                "\nCidade: " + cidade +
                "\nEstado: " + estado +
                "\nTelefone: " + telefone +
                "\nE-mail: " + email;
    }
}

package cadastrobd.model;

public class PessoaFisica extends Pessoa {
    private String cpf;

    public PessoaFisica() {
        // const padrão
    }

    public PessoaFisica(int id, String nome, String logradouro, String cidade,
String estado, String telefone, String email, String cpf) {
        super(id, nome, logradouro, cidade, estado, telefone, email);
        this.cpf = cpf;
    }

    PessoaFisica(int idPessoa, String nome, String logradouro, String cidade,
String estado, String telefone, String email, String cpf) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

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

    //getters e setters para o CPF
    public String getCpf() {
```

```
        return cpf;
    }

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

    String getNome() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    String getLogradouro() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    String getCidade() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    String getEstado() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    String getTelefone() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    String getEmail() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    int getIdPessoa() {
```

```
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    public void setNome(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    public void setLogradouro(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    public void setCidade(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    public void setEstado(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    public void setTelefone(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    public void setEmail(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

}
```

```
package cadastrobd.model;

public class PessoaJuridica extends Pessoa {
    private String cnpj;

    public PessoaJuridica() {
        //const padrão
    }

    public PessoaJuridica(int id, String nome, String logradouro, String cidade,
String estado, String telefone, String email, String cnpj) {
        super(id, nome, logradouro, cidade, estado, telefone, email);
        this.cnpj = cnpj;
    }

    PessoaJuridica(int idPessoa, String nome, String logradouro, String cidade,
String estado, String telefone, String email, String cnpj) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

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

    //getters e setters para o CNPJ
    public String getCnpj() {
        return cnpj;
    }

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

    String getNome() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    String getLogradouro() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
```

```
dy
    }

    String getCidade() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    String getEstado() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    String getTelefone() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    String getEmail() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    int getIdPessoa() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    public void setNome(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    public void setLogradouro(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
```

```
    }

    public void setCidade(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    public void setEstado(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    public void setTelefone(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    public void setEmail(String nextLine) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    int getId() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    int getId() {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

}

package cadastrobd.model;

import cadastro.model.PessoaFisica;
import cadastro.model.PessoaFisicaDAO;
```

```
import cadastro.model.PessoaJuridica;
import cadastro.model.PessoaJuridicaDAO;

import java.util.Scanner;

public class CadastroBDTeste {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int opcao;

        do {
            exibirMenu();
            opcao = scanner.nextInt();

            switch (opcao) {
                case 1:
                    incluirPessoa(scanner);
                    break;
                case 2:
                    alterarPessoa(scanner);
                    break;
                case 3:
                    excluirPessoa(scanner);
                    break;
                case 4:
                    exibirPorId(scanner);
                    break;
                case 5:
                    exibirTodos();
                    break;
                case 0:
                    System.out.println("Saindo do programa.");
                    break;
                default:
                    System.out.println("Opção inválida. Tente novamente.");
                    break;
            }
        } while (opcao != 0);

        scanner.close();
    }

    private static void exibirMenu() {
        System.out.println("Escolha uma opção:");
        System.out.println("1 - Incluir");
        System.out.println("2 - Alterar");
        System.out.println("3 - Excluir");
        System.out.println("4 - Exibir pelo ID");
        System.out.println("5 - Exibir todos");
```

```
        System.out.println("0 - Sair");
    }

    private static void incluirPessoa(Scanner scanner) {
        System.out.println("Escolha o tipo (1 para Pessoa Física, 2 para Pessoa
Jurídica):");
        int tipo = scanner.nextInt();

        if (tipo == 1) {
            PessoaFisica pessoaFisica = lerDadosPessoaFisica(scanner);
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
            pessoaFisicaDAO.incluir(pessoaFisica);
            System.out.println("Pessoa Física incluída com sucesso.");
        } else if (tipo == 2) {
            PessoaJuridica pessoaJuridica = lerDadosPessoaJuridica(scanner);
            PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
            pessoaJuridicaDAO.incluir(pessoaJuridica);
            System.out.println("Pessoa Jurídica incluída com sucesso.");
        } else {
            System.out.println("Opção inválida.");
        }
    }

    private static PessoaFisica lerDadosPessoaFisica(Scanner scanner) {
        System.out.println("Implemente a leitura dos dados de Pessoa Física
aqui.");

        return null;
    }

    private static PessoaJuridica lerDadosPessoaJuridica(Scanner scanner) {
        System.out.println("Implemente a leitura dos dados de Pessoa Jurídica
aqui.");

        return null;
    }

    private static void alterarPessoa(Scanner scanner) {
        System.out.println("Implemente a opção de alterar Pessoa Física ou
Jurídica aqui.");
    }

    private static void excluirPessoa(Scanner scanner) {
        System.out.println("Implemente a opção de excluir Pessoa Física ou
Jurídica aqui.");
    }

    private static void exibirPorId(Scanner scanner) {
        System.out.println("Implemente a opção de exibir por ID Pessoa Física ou
```

```
Jurídica aqui.");
    }

    private static void exibirTodos() {
        System.out.println("Implemente a opção de exibir todas as pessoas
aqui.");
    }
}
```

Análise e Conclusão

Qual a importância dos componentes de middleware, como o JDBC?
Facilitam a comunicação: Permitem que aplicativos se conectem e interajam com diferentes sistemas e recursos de dados.
Padronizam o acesso: Oferecem uma camada comum para acessar informações, abstraindo as complexidades específicas dos sistemas subjacentes.
Promovem a portabilidade: Permitem que aplicativos sejam mais facilmente transferidos entre diferentes ambientes devido à sua capacidade de se adaptar a diferentes tecnologias de banco de dados.
Contribuem para a segurança: Fornecem recursos para evitar vulnerabilidades comuns, garantindo práticas mais seguras no acesso e manipulação de dados.

Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?
A principal diferença está na forma como lidam com consultas SQL e parâmetros:
Statement: É utilizado para consultas SQL estáticas e diretas ao banco de dados. As consultas são escritas diretamente na string SQL e executadas.
PreparedStatement: É preferível para consultas parametrizadas, onde os valores dos parâmetros são fornecidos separadamente da consulta SQL. Isso aumenta a segurança, pois previne ataques e melhora o desempenho, já que o banco de dados pode reutilizar consultas preparadas.

Como o padrão DAO melhora manutenibilidade do software?
Melhora a manutenibilidade do software ao separar a lógica de acesso aos dados da lógica de negócios, proporcionando uma organização estruturada, reutilização de código, facilidade de manutenção e testabilidade.

Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?
Neste modelo, a herança pode ser representada de três maneiras principais:
Herança por Tabelas Concretas: Cada classe se torna uma tabela separada no banco de dados, resultando em redundância de dados.
Herança por Tabelas Separadas: Cada classe é mapeada para uma tabela própria, reduzindo a redundância, mas pode demandar mais joins para obter todos os dados de uma hierarquia.
Herança por Tabelas de Junção: Utiliza tabelas comuns para dados compartilhados e tabelas específicas para dados exclusivos de cada classe,

mantendo a estrutura mais normalizada, mas exigindo joins para reunir informações completas.

Procedimento 2 – Alimentando a base

Códigos do Procedimento 2

```
package cadastrobd;

import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;
import cadastro.model.util.ConectorBD;

import java.util.List;
import java.util.Scanner;

public class CadastroBD {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ConectorBD conectorBD = new ConectorBD();

        PessoaFisicaDAO pessoaFisicaDAO = new
PessoaFisicaDAO(conectorBD);
        PessoaJuridicaDAO pessoaJuridicaDAO = new
PessoaJuridicaDAO(conectorBD);

        int opcao;
        do {
            exibirMenu();
            opcao = scanner.nextInt();
            scanner.nextLine(); // Limpar o buffer

            switch (opcao) {
                case 1:
                    incluirPessoa(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    break;

                case 2:
                    alterarPessoa(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    break;

                case 3:
                    excluirPessoa(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                    break;

                case 4:
                    exibirPessoaPorId(scanner, pessoaFisicaDAO,
```

```
pessoaJuridicaDAO);
                    break;

                case 5:
                    exibirTodasPessoas(scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                    break;

                case 0:
                    System.out.println("Encerrando o programa.");
                    break;

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

        } while (opcao != 0);

        scanner.close();
    }

    private static void exibirMenu() {
        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
        System.out.println("Selecione uma opção:");
        System.out.println("1. Incluir");
        System.out.println("2. Alterar");
        System.out.println("3. Excluir");
        System.out.println("4. Exibir pelo ID");
        System.out.println("5. Exibir todos");
        System.out.println("0. Sair");
        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
    }

    private static void incluirPessoa(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        System.out.println("Escolha o tipo (F - Física, J - Jurídica):");
        String tipo = scanner.nextLine().toUpperCase();

        if (tipo.equalsIgnoreCase("F")) {
            PessoaFisica pessoaFisica = criarPessoaFisica(scanner);
            pessoaFisicaDAO.incluir(pessoaFisica);
            System.out.println("Pessoa Física incluída com sucesso!");
        } else if (tipo.equalsIgnoreCase("J")) {
            PessoaJuridica pessoaJuridica = criarPessoaJuridica(scanner);
            pessoaJuridicaDAO.incluir(pessoaJuridica);
            System.out.println("Pessoa Jurídica incluída com sucesso!");
        } else {
            System.out.println("Opção inválida.");
```

```
        }
    }

    private static void alterarPessoa(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        System.out.println("Escolha o tipo (F - Física, J - Jurídica):");
        String tipoAlterar = scanner.nextLine().toUpperCase();

        System.out.print("Informe o ID da pessoa: ");
        int idAlterar = scanner.nextInt();
        scanner.nextLine(); // Limpar o buffer

        if (tipoAlterar.equalsIgnoreCase("F")) {
            PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(idAlterar);
            if (pessoaFisica != null) {
                exibirDadosAtuais(pessoaFisica);
                atualizarPessoaFisica(scanner, pessoaFisica);
                pessoaFisicaDAO.alterar(pessoaFisica);
                System.out.println("Pessoa Física alterada com sucesso!");
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipoAlterar.equalsIgnoreCase("J")) {
            PessoaJuridica pessoaJuridica =
pessoaJuridicaDAO.getPessoa(idAlterar);
            if (pessoaJuridica != null) {
                exibirDadosAtuais(pessoaJuridica);
                atualizarPessoaJuridica(scanner, pessoaJuridica);
                pessoaJuridicaDAO.alterar(pessoaJuridica);
                System.out.println("Pessoa Jurídica alterada com sucesso!");
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Opção inválida.");
        }
    }

    private static void excluirPessoa(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        System.out.println("Escolha o tipo (F - Física, J - Jurídica):");
        String tipoExcluir = scanner.nextLine().toUpperCase();

        System.out.print("Informe o ID da pessoa: ");
        int idExcluir = scanner.nextInt();
        scanner.nextLine();

        if (tipoExcluir.equalsIgnoreCase("F")) {
            PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(idExcluir);
```

```
            if (pessoaFisica != null) {
                pessoaFisicaDAO.excluir(pessoaFisica);
                System.out.println("Pessoa Física excluída com sucesso!");
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipoExcluir.equalsIgnoreCase("J")) {
            PessoaJuridica pessoaJuridica =
pessoaJuridicaDAO.getPessoa(idExcluir);
            if (pessoaJuridica != null) {
                pessoaJuridicaDAO.excluir(pessoaJuridica);
                System.out.println("Pessoa Jurídica excluída com sucesso!");
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Opção inválida.");
        }
    }

    private static void exibirPessoaPorId(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        System.out.println("Escolha o tipo (F - Física, J - Jurídica):");
        String tipoExibirId = scanner.nextLine().toUpperCase();

        System.out.print("Informe o ID da pessoa: ");
        int idExibirId = scanner.nextInt();
        scanner.nextLine();

        if (tipoExibirId.equalsIgnoreCase("F")) {
            PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(idExibirId);
            if (pessoaFisica != null) {
                pessoaFisica.exibir();
            } else {
                System.out.println("Pessoa Física não encontrada.");
            }
        } else if (tipoExibirId.equalsIgnoreCase("J")) {
            PessoaJuridica pessoaJuridica =
pessoaJuridicaDAO.getPessoa(idExibirId);
            if (pessoaJuridica != null) {
                pessoaJuridica.exibir();
            } else {
                System.out.println("Pessoa Jurídica não encontrada.");
            }
        } else {
            System.out.println("Opção inválida.");
        }
    }
```

```
    private static void exibirTodasPessoas(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        System.out.println("Escolha o tipo (F - Física, J - Jurídica):");
        String tipoExibirTodos = scanner.nextLine().toUpperCase();

        if (tipoExibirTodos.equalsIgnoreCase("F")) {
            System.out.println("Exibindo dados de Pessoa Fisica...");
            List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
            for (PessoaFisica pf : pessoasFisicas) {
                pf.exibir();
            }
        } else if (tipoExibirTodos.equalsIgnoreCase("J")) {
            System.out.println("Exibindo dados de Pessoa Juridica...");
            List<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.getPessoas();
            for (PessoaJuridica pj : pessoasJuridicas) {
                pj.exibir();
            }
        } else {
            System.out.println("Opção inválida.");
        }
    }

    private static PessoaFisica criarPessoaFisica(Scanner scanner) {
        PessoaFisica pessoaFisica = new PessoaFisica();
        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
        System.out.print("Nome: ");
        pessoaFisica.setNome(scanner.nextLine());

        System.out.print("Logradouro: ");
        pessoaFisica.setLogradouro(scanner.nextLine());

        System.out.print("Cidade: ");
        pessoaFisica.setCidade(scanner.nextLine());

        System.out.print("Estado: ");
        pessoaFisica.setEstado(scanner.nextLine());

        System.out.print("Telefone: ");
        pessoaFisica.setTelefone(scanner.nextLine());

        System.out.print("E-mail: ");
        pessoaFisica.setEmail(scanner.nextLine());

        System.out.print("CPF: ");
        pessoaFisica.setCpf(scanner.nextLine());
        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
        return pessoaFisica;
    }
```

```
    private static PessoaJuridica criarPessoaJuridica(Scanner scanner) {
        PessoaJuridica pessoaJuridica = new PessoaJuridica();
        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
        System.out.print("Nome: ");
        pessoaJuridica.setNome(scanner.nextLine());

        System.out.print("Logradouro: ");
        pessoaJuridica.setLogradouro(scanner.nextLine());

        System.out.print("Cidade: ");
        pessoaJuridica.setCidade(scanner.nextLine());

        System.out.print("Estado: ");
        pessoaJuridica.setEstado(scanner.nextLine());

        System.out.print("Telefone: ");
        pessoaJuridica.setTelefone(scanner.nextLine());

        System.out.print("E-mail: ");
        pessoaJuridica.setEmail(scanner.nextLine());

        System.out.print("CNPJ: ");
        pessoaJuridica.setCnpj(scanner.nextLine());
        System.out.println("-=-=-=-=-=-=-=-=-=-=-=-=-");
        return pessoaJuridica;
    }

    private static void exibirDadosAtuais(PessoaFisica pessoaFisica) {
        pessoaF

    private static void atualizarPessoaFisica(Scanner scanner, PessoaFisica
pessoaFisica) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    private static void exibirDadosAtuais(PessoaJuridica pessoaJuridica) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    private static void atualizarPessoaJuridica(Scanner scanner, PessoaJuridica
pessoaJuridica) {
        throw new UnsupportedOperationException("Not supported yet."); //
```

```
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    private static void atualizarDados(Scanner scanner, PessoaJuridica
pessoaJuridica) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    private static void atualizarDados(Scanner scanner, PessoaJuridica
pessoaJuridica) {
        throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBo
dy
    }

    private static class ConectorBD {

        public ConectorBD() {
        }
    }
```

Análise e Conclusão – Procedimento 2

Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?
A persistência em arquivo tende a ser mais simples, porém menos robusta em termos de recursos, segurança e controle, enquanto a persistência em banco de dados oferece recursos avançados para gerenciamento, segurança e manipulação de dados, embora seja mais complexa de administrar.

Como o uso de operador lambda simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?
Simplificou a impressão dos valores contidos nas entidades ao permitir a expressão de funções de forma mais concisa e direta. Nas versões mais recentes do Java, os operadores lambda, combinados com métodos como ‘forEach’ em coleções, permitem uma escrita mais enxuta e legível para imprimir os valores das entidades. Isso reduz a necessidade de loops explícitos, oferecendo uma abordagem mais funcional e declarativa para operar operar sobre os dados, tornando o código mais claro e sucinto.

Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static?
Métodos acionados diretamente pelo método ‘main’ precisam ser marcados como ‘static’ para serem acessados sem a necessidade de criar uma instância

da classe. O método ‘main’ também é estático e, portanto, pode chamar apenas métodos estáticos diretamente. Isso permite que o programa seja iniciado sem criar um objeto da classe, já que métodos estáticos pertencem à classe e não instâncias específicas, permitindo seu acesso direto.