Missão Prática Nível 03 Mundo 3

Iggor Motta de Oliveira. Matricula:202303067879

Campus Virtual - EAD
RPG0016 – BackEnd sem banco não tem – Período 2024.3

Objetivo da Prática

- Implementar persistência com base no middleware JDBC.
- Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
- Implementar o mapeamento objeto-relacional em sistemas Java.
- Criar sistemas cadastrais com persistência em banco relacional.
- No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Códigos do roteiro:

```
package cadastrobd;

import java.util.ArrayList;

import java.util.Scanner;

import cadastrobd.model.PessoaFisica;

import cadastrobd.model.PessoaFisicaDAO;

import cadastrobd.model.PessoaJuridica;

import cadastrobd.model.PessoaJuridicaDAO;

import java.sql.SQLException;

import java.util.logging.Logger;

import java.util.logging.Level;

public class CadastroBD {
```

```
private static final Logger LOGGER = Logger.getLogger(CadastroBD.class.getName());
private Scanner in;
private PessoaFisicaDAO pfDao;
private PessoaJuridicaDAO pjDao;

public CadastroBD() {
    in = new Scanner(System.in);
    pfDao = new PessoaFisicaDAO();
    pjDao = new PessoaJuridicaDAO();
}
 private String strAnswerQuestion(String question) {
    System.out.print(question);
    return in.nextLine();
}
   private Integer intAnswerQuestion(String question) {
    System.out.print(question);
    String strValue = in.nextLine();
    Integer intValue = 0;
    try {
        intValue = Integer.valueOf(strValue);
    }
    catch (NumberFormatException e) {
        LOGGER.log(Level.SEVERE, e.toString(), e);
    }
    return intValue;    }
```

Análise e Conclusão:

a) Qual a importância dos componentes de middleware, como o JDBC?

- Forte importâcia pois é assim que é feita a interação entre aplicativos JAVA e banco de dados, facilitando o acesso e a manipulação de dado emu ma aplicação.

b) Qual a diferença no uso de *Statement* ou *PreparedStatement* para a manipulação de dados?

- PreparedStatement é preferida em termos de segurança, pois previne ataques de injeção SQL ao tratar os parâmetros como dados e não como parte da consulta, reduzindo o risco de código malicioso,E em desempenh, é mais eficiente que Statement, especialmente em consultas repetitivas,pois o banco compila a consulta um vez e reultiliza. A manutenção também é melhor, pois os parâmetros ficam separados da consulta, facilitando a compreensão.

c) Como o padrão DAO melhora a manutenibilidade do software?

- Padrão de projeto DAO surgiu com a necessidade de separarmos a lógica de negócios da lógica de persistência de dados. Este padrão permite que possamos mudar a forma de persistência sem que isso influencie em nada na lógica de negócio, além de tornar nossas classes mais legíveis. Toda interação com a base de dados se dará através destas classes, nunca das classes de negócio, muito menos de formulários.Se aplicarmos este padrão corretamente ele vai abstrair completamente o modo de busca e gravação dos dados, tornando isso transparente para aplicação e facilitando muito na hora de fazermos manutenção na aplicação ou migrarmos de banco de dados.

d) Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?

Tabela única (Single Table Inheritance): Todas as subclasses são armazenadas em uma única tabela, com colunas que representam todos os atributos de todas as classes da hierarquia. Uma coluna discriminadora identifica a qual subclasse cada linha pertence. Esse método é eficiente em termos de consulta, pois envolve apenas uma tabela, mas pode gerar muitos valores nulos, caso as subclasses tenham atributos exclusivos.

Tabela por classe (Table per Class): Cada classe na hierarquia possui sua própria tabela. As tabelas das subclasses contêm apenas os atributos específicos, enquanto as colunas da superclasse são replicadas em todas as tabelas de subclasse. Esse padrão evita valores

nulos, mas dificulta consultas que precisam combinar dados de várias subclasses, pois podem exigir junções complexas.

Tabela por hierarquia (Class Table Inheritance): Cada classe tem sua própria tabela, e as tabelas das subclasses se relacionam com a tabela da superclasse por meio de chaves estrangeiras. Apenas os atributos exclusivos são armazenados em cada tabela de subclasse, e os atributos da superclasse ficam na tabela dela. Esse método mantém a normalização e evita valores nulos, mas as consultas exigem junções para recuperar dados de objetos completos.

Herança é refletida de acordo com o tamanho da hierarquia, quantidade de dados ,quantidades de campos comuns e tipo de consultas que serão feitas nos dados.

A escolha de uma estratégia depende dos requisitos de desempenho, integridade dos dados e simplicidade nas consultas, bem como das características das classes e atributos.

2º Procedimento | Alimentando a Base

Código do Roteiro:

```
private void printMenu() {
        System.out.println("\n================================");
        System.out.println("1 - Incluir");
        System.out.println("2 - Alterar");
        System.out.println("3 - Excluir");
        System.out.println("4 - Buscar pelo ID");
        System.out.println("5 - Exibir todos");
        System.out.println("0 - Sair");
        System.out.println("================================");
    }

    public void run() {
        int opcao = -1;
        while (opcao != 0) {
            printMenu();
```

```
        opcao = intAnswerQuestion("ESCOLHA: ");
        switch (opcao) {
           case 1: {
              System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
              String escolhaIncluir = strAnswerQuestion("TIPO DE PESSOA: ").toUpperCase();
              if (escolhaIncluir.equals("F")) {
                 String nome = strAnswerQuestion("Informe o nome: ");
                 String cpf = strAnswerQuestion("Informe o CPF: ");
                 String endereco = strAnswerQuestion("Informe o endereco: ");
                 String cidade = strAnswerQuestion("Informe a cidade: ");
                 String estado = strAnswerQuestion("Informe o estado: ");
                 String telefone = strAnswerQuestion("Informe o telefone: ");
                 String email = strAnswerQuestion("Informe o email: ");
                 try {
                    pfDao.incluir(new PessoaFisica(null, nome, endereco, cidade, estado, telefone, email, cpf));
                 }
                 catch (SQLException e) {
                    LOGGER.log(Level.SEVERE, e.toString(), e);
                 }
              }
              else if (escolhaIncluir.equals("J")) {
                 String nome = strAnswerQuestion("Informe o nome: ");
                 String cnpj = strAnswerQuestion("Informe o CNPJ: ");
                 String endereco = strAnswerQuestion("Informe o endereco: ");
                 String cidade = strAnswerQuestion("Informe a cidade: ");
                 String estado = strAnswerQuestion("Informe o estado: ");
                 String telefone = strAnswerQuestion("Informe o telefone: ");
                 String email = strAnswerQuestion("Informe o email: ");
                 try {
                    pjDao.incluir(new PessoaJuridica(null, nome, endereco, cidade, estado, telefone, email, cnpj));
                 }
                 catch (SQLException e) {
```

```
                    LOGGER.log(Level.SEVERE, e.toString(), e);
                }
            }
            else {
                System.out.println("Erro: Escolha Invalida!");
            }
        }; break;
        case 2: {
            System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
            String escolhaAlterar = strAnswerQuestion("TIPO DE PESSOA: ").toUpperCase();
            if (escolhaAlterar.equals("F")) {
                try {
                    Integer id = intAnswerQuestion("Informe o ID da Pessoa Fisica: ");
                    PessoaFisica pf = pfDao.getPessoa(id);
                    if (pf != null) {
                        pf.setNome(strAnswerQuestion("Informe o nome: "));
                        pf.setCpf(strAnswerQuestion("Informe o CPF: "));
                        pf.setEndereco(strAnswerQuestion("Informe o endereco: "));
                        pf.setCidade(strAnswerQuestion("Informe a cidade: "));
                        pf.setEstado(strAnswerQuestion("Informe o estado: "));
                        pf.setTelefone(strAnswerQuestion("Informe o telefone: "));
                        pf.setEmail(strAnswerQuestion("Informe o email: "));
                        pfDao.alterar(pf);
                    }
                    else {
                        System.out.println("ID nao encontrado!");
                    }
                }
                catch (NullPointerException | SQLException e) {
                    LOGGER.log(Level.SEVERE, e.toString(), e);
                }

            }
            else if (escolhaAlterar.equals("J")) {
```

```
                try {
                    Integer id = intAnswerQuestion("Informe o ID da Pessoa Juridica: ");
                    PessoaJuridica pj = pjDao.getPessoa(id);
                    if (pj != null) {
                        pj.setNome(strAnswerQuestion("Informe o nome: "));
                        pj.setCnpj(strAnswerQuestion("Informe o CNPJ: "));
                        pj.setEndereco(strAnswerQuestion("Informe o endereco: "));
                        pj.setCidade(strAnswerQuestion("Informe a cidade: "));
                        pj.setEstado(strAnswerQuestion("Informe o estado: "));
                        pj.setTelefone(strAnswerQuestion("Informe o telefone: "));
                        pj.setEmail(strAnswerQuestion("Informe o email: "));
                        pjDao.alterar(pj);
                    }
                    else {
                        System.out.println("ID nao encontrado!");
                    }
                }
                catch (NullPointerException | SQLException e) {
                    LOGGER.log(Level.SEVERE, e.toString(), e);
                }
            }
            else {
                System.out.println("Erro: Escolha Invalida!");
            }
        }; break;
        case 3: {
            System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
            String escolhaExcluir = strAnswerQuestion("TIPO DE PESSOA: ").toUpperCase();
            if (escolhaExcluir.equals("F")) {
                try {
                    Integer id = intAnswerQuestion("Informe o ID da Pessoa Fisica: ");
                    PessoaFisica pf = pfDao.getPessoa(id);
                    if (pf != null) {
                        pfDao.excluir(pf);
```

```
                    System.out.println("Excluido com sucesso.");
                }
                else {
                    System.out.println("ID nao encontrado.");
                }
            }
            catch (NullPointerException | SQLException e) {
                LOGGER.log(Level.SEVERE, e.toString(), e);
            }
        }
        else if (escolhaExcluir.equals("J")) {
            try {
                Integer id = intAnswerQuestion("Informe o ID da Pessoa Juridica: ");
                PessoaJuridica pj = pjDao.getPessoa(id);
                if (pj != null) {
                    pjDao.excluir(pj);
                    System.out.println("Excluido com sucesso.");
                }
                else {
                    System.out.println("ID nao encontrado.");
                }
            }
            catch (NullPointerException | SQLException e) {
                LOGGER.log(Level.SEVERE, e.toString(), e);
            }
        }
        else {
            System.out.println("Erro: Escolha Invalida!");
        }
    }; break;
    case 4: {
        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
        String escolhaExibir = strAnswerQuestion("TIPO DE PESSOA: ").toUpperCase();
        if (escolhaExibir.equals("F")) {
```

```
                try {
                    PessoaFisica pf = pfDao.getPessoa(intAnswerQuestion("Informe o ID da
Pessoa Fisica: "));
                    if (pf != null) {
                        pf.exibir();
                    }
                }
                catch (SQLException e){
                    System.err.println("Pessoa nao encontrada!");
                    LOGGER.log(Level.SEVERE, e.toString(), e);
                }
            }
            else if (escolhaExibir.equals("J")) {
                try {
                    PessoaJuridica pj = pjDao.getPessoa(intAnswerQuestion("Informe o ID
da Pessoa Juridica: "));
                    if (pj != null) {
                        pj.exibir();
                    }
                }
                catch (SQLException e){
                    System.err.println("Pessoa nao encontrada!");
                    LOGGER.log(Level.SEVERE, e.toString(), e);
                }
            }
            else {
                System.out.println("Erro: Escolha Invalida!");
            }
        }; break;
        case 5: {
            try {
                ArrayList<PessoaFisica> listaPf = pfDao.getPessoas();
                for (PessoaFisica pessoa : listaPf) {
                    System.out.println("--------------------------------");
                    pessoa.exibir();
```

```
                }
                System.out.println("--------------------------------");
                ArrayList<PessoaJuridica> listaPj = pjDao.getPessoas();
                for (PessoaJuridica pessoa : listaPj) {
                    pessoa.exibir();
                    System.out.println("--------------------------------");
                }
            }
            catch (SQLException e) {
                LOGGER.log(Level.SEVERE, e.toString(), e);
            }
        }; break;
        default: System.out.println("Escolha invalida!");
      }
    }
  }


  public static void main(String[] args) {
    new CadastroBD().run();
  }

}
```

Análise e Conclusão:

a) Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?

- A principal diferença entre a persistência em arquivo e a persistência em banco de dados é que o banco de dados é um local de armazenamento que permite uma recuperação e organização mais eficientes dos dados. A persistência de dados pode ser feita em vários locais, como em um banco de dados, em uma planilha do Excel ou em um arquivo de texto. No entanto, para uma análise de dados mais eficiente, é recomendado o uso de um banco de dados.

b) Como o uso de operador *lambda* simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?

- As funções lambda em Java, introduzidas no Java 8, trazem técnicas de programação funcional para a linguagem, inspiradas em linguagens como Scala e LISP. Simplificando com a quantidade de código, especialmente em situações que normalmente exigem classes internas, como Listeners e Threads. Em resumo, uma função lambda é uma função sem nome, tipo de retorno ou modificador de acesso, definida diretamente no local onde será usada.

c) Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como *static*?

- A palavra-chave "static" é usada para indicar que o método "main" pertence à classe, não a uma instância específica da classe. Isso significa que o método "main" pode ser chamado sem criar um objeto da classe. Isso é necessário porque o método "main" é chamado pelo sistema de execução do programa, antes que qualquer objeto seja criado.