

Desenvolvimento Full Stack

Nível 1: Iniciando o Caminho Pelo Java

2023.1

Mundo 3 Período 2024.1

Breno Félix de Souza

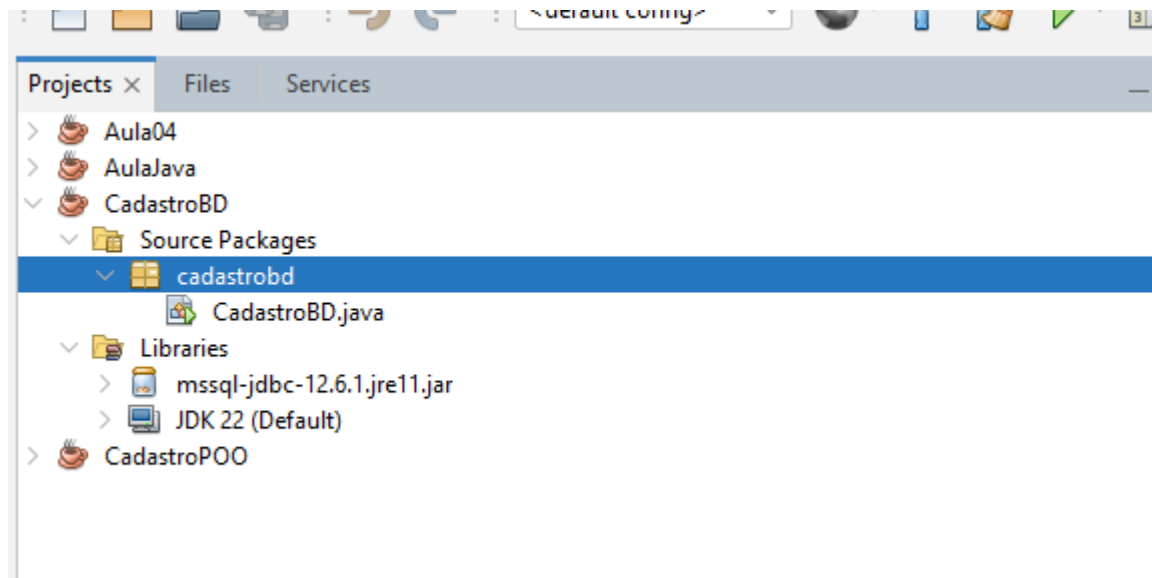
1º Procedimento | Mapeamento Objeto-Relacional e DAO

1) Criar o projeto e configurar as bibliotecas necessárias:

- a. Criar um projeto no NetBeans, utilizando o nome CadastroBD, do tipo Aplicativo Java Padrão (modelo Ant).
- b. Adicionar o driver JDBC para SQL Server ao projeto, com o clique do botão direito sobre bibliotecas (libraries) e escolha da opção jar.
- c. Selecionar o arquivo mssql-jdbc-12.2.0.jre11.jar, que é parte do arquivo zip encontrado no endereço seguinte.

<https://learn.microsoft.com/pt-br/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver16>

- d. Após descompactar o arquivo, copie o arquivo jar necessário para uma pasta de fácil acesso e adicione ao projeto, conforme instrução anterior.



2) Configurar o acesso ao banco pela aba de serviços do NetBeans:

- a. Na aba de Serviços, divisão Banco de Dados, clique com o botão direito em Drivers e escolha Novo Driver.
- b. Na janela que se abrirá, clicar em Add (Adicionar), escolher o arquivo jar utilizado no passo anterior e finalizar com Ok.
- c. O reconhecimento será automático, e podemos definir uma conexão com o clique do botão direito sobre o driver e escolha de Conectar Utilizando.
- d. Para os campos database, user e password, utilizar o valor loja, de acordo com os elementos criados em exercício anterior sobre a criação do banco de dados de exemplo, marcando também a opção Lembrar Senha.

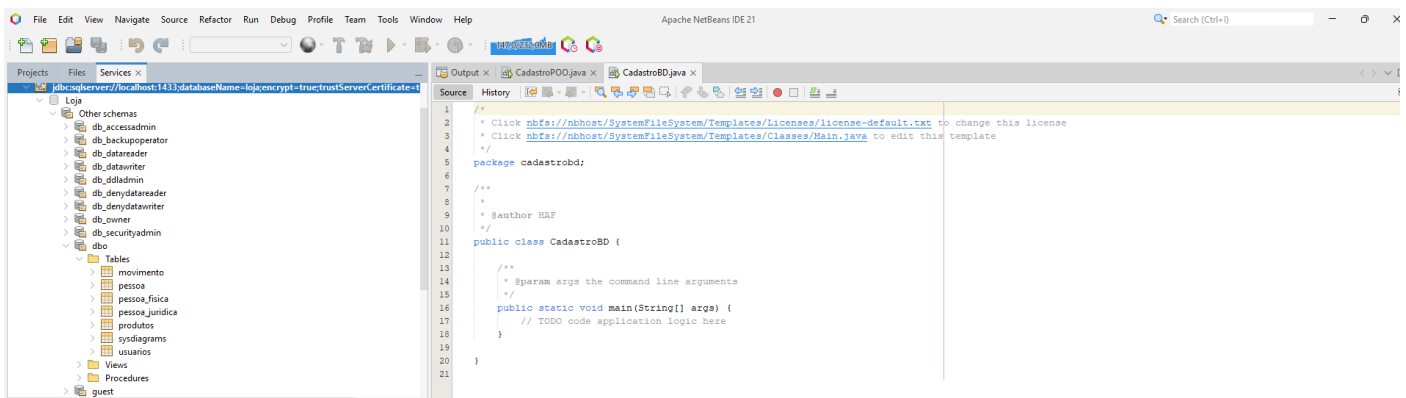
e. Para o campo JDBC URL deve ser utilizada a seguinte expressão:

`jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;`

f. Clicar em Testar Conexão e, estando tudo certo, Finalizar.

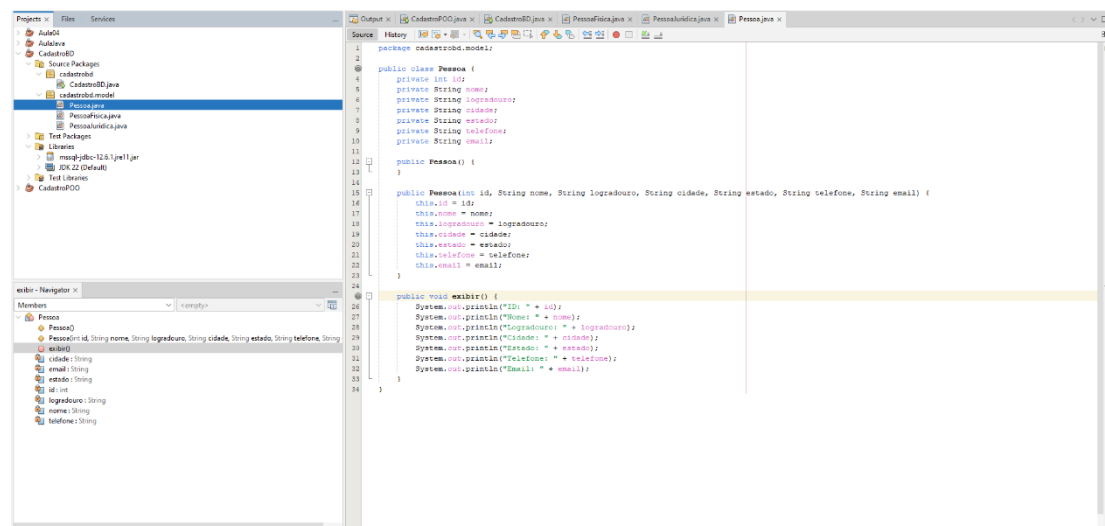
g. Ao clicar duas vezes na nova conexão, os objetos do banco estarão todos disponíveis na árvore de navegação.

h. Utilizar o clique do botão direito sobre as tabelas, e escolher Visualizar Dados (View Data), para consultar os dados atualmente no banco.

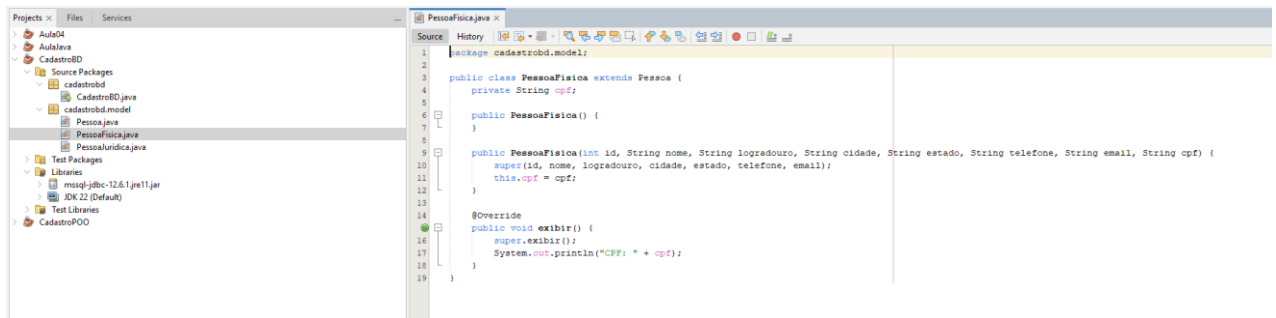


3) Voltando ao projeto, criar o pacote `cadastrobd.model`, e nele criar as classes apresentadas a seguir:

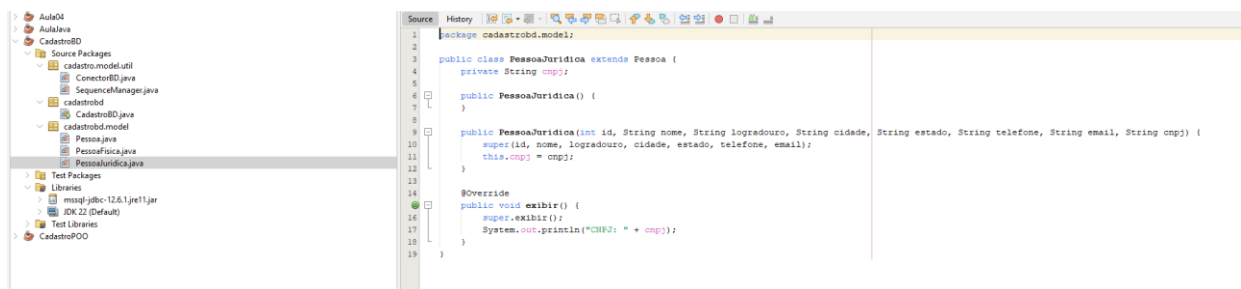
a. Classe `Pessoa`, com os campos `id`, `nome`, `logradouro`, `cidade`, `estado`, `telefone` e `email`, construtor padrão e completo, além de método `exibir`, para impressão dos dados no console.



- b. Classe **PessoaFisica**, herdando de **Pessoa**, com acréscimo do campo **cpf**, além da reescrita dos construtores e uso de polimorfismo em **exibir**.



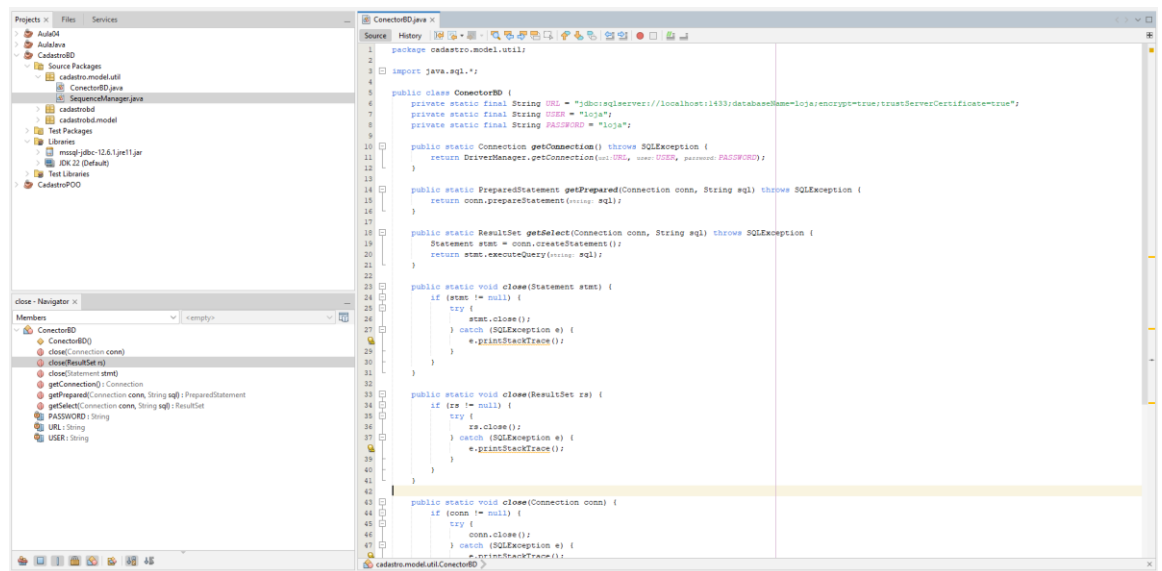
- c. Classe **PessoaJuridica**, herdando de **Pessoa**, com acréscimo do campo **cnpj**, além da reescrita dos construtores e uso de polimorfismo em **exibir**



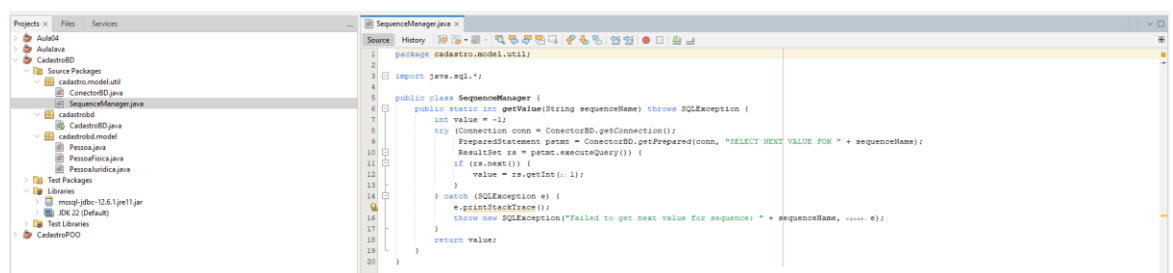
- 4) Criar o pacote **cadastro.model.util**, para inclusão das classes utilitárias que são apresentadas a seguir:

- a. Classe **ConectorBD**, com os métodos **getConnection**, para retornar uma conexão com o banco de dados, **getPrepared**, para retornar um objeto do tipo **PreparedStatement** a partir de um SQL fornecido com parâmetro, e **getSelect**, para retornar o **ResultSet** relacionado a uma consulta.

- b. Ainda na classe ConectorBD, adicionar métodos close sobrecarregados para Statement, ResultSet e Connection, visando garantir o fechamento, ou encerramento, de todos os objetos de acesso ao banco gerados.

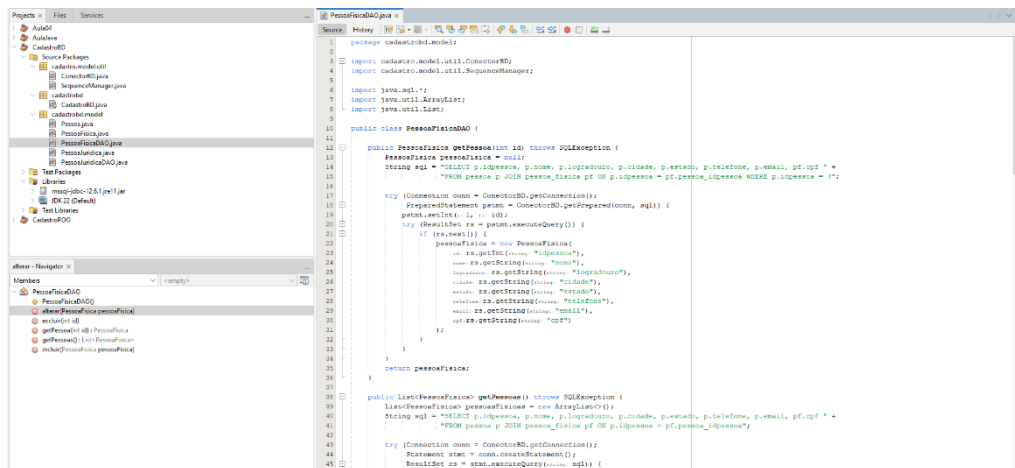


- c. Classe SequenceManager, que terá o método getValue, recebendo o nome da sequência como parâmetro e retornando o próximo valor.



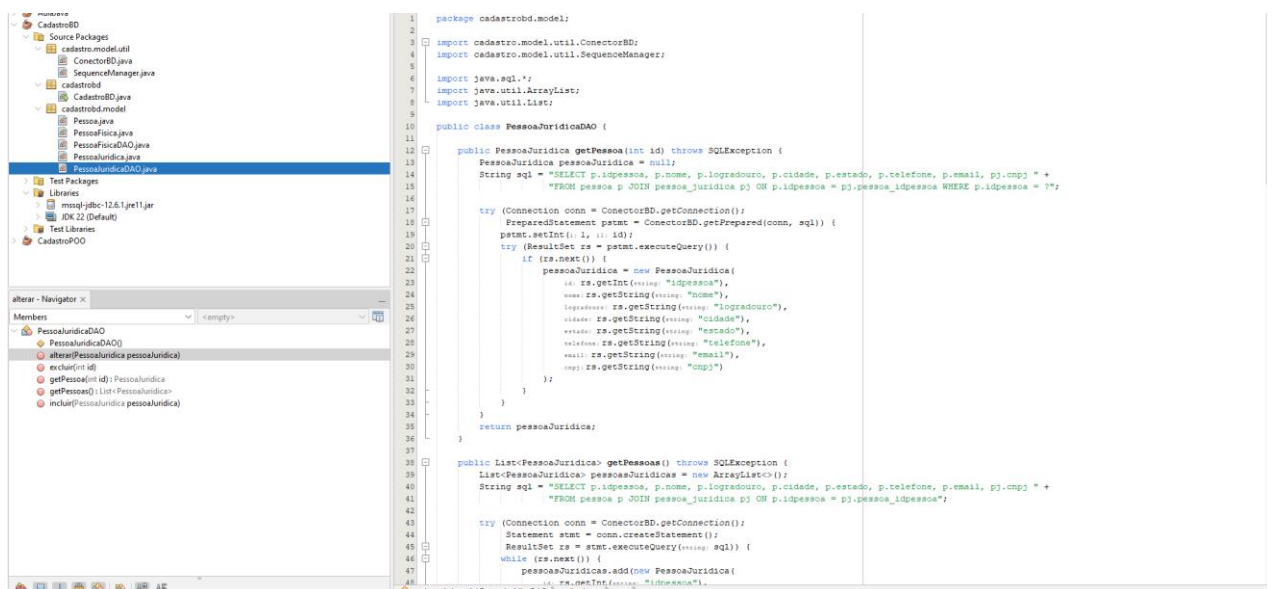
5) Codificar as classes no padrão DAO, no pacote cadastro.model.

- a. **Classe PessoaFisicaDAO**, com os métodos **getPessoa**, retornando uma pessoa física a partir do seu id, **getPessoas**, para retorno de todas as pessoas físicas do banco de dados, incluir, para inclusão de uma pessoa física, fornecida como parâmetro, nas tabelas Pessoa e PessoaFisica, alterar, para alteração dos dados de uma pessoa física, e excluir, para remoção da pessoa do banco em ambas as tabelas.



```
1 package cadastro.model;
2
3 import cadastro.model.util.ConnectorBD;
4 import cadastro.model.util.SequenciaManager;
5
6 import java.sql.*;
7 import java.util.ArrayList;
8 import java.util.List;
9
10 public class PessoaFisicaDAO {
11
12     public PessoaFisica getPessoa(int id) throws SQLException {
13         PessoaFisica pessoaFisica = null;
14         String sql = "SELECT p.idpessoa, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, pf.cpf " +
15             "FROM pessoa p JOIN pessoa_fisica pf ON p.idpessoa = pf.pessoa_idpessoa WHERE p.idpessoa = ?";
16
17         try (Connection conn = ConnectorBD.getConnection();
18             PreparedStatement pstmt = ConnectorBD.getPreparedStatement(conn, sql)) {
19             pstmt.setInt(1, id);
20             try (ResultSet rs = pstmt.executeQuery()) {
21                 if (rs.next()) {
22                     pessoaFisica = new PessoaFisica(
23                         rs.getInt("idpessoa"),
24                         rs.getString("nome"),
25                         rs.getString("logradouro"),
26                         rs.getString("cidade"),
27                         rs.getString("estado"),
28                         rs.getString("telefone"),
29                         rs.getString("email"),
30                         rs.getString("cpf")
31                     );
32                 }
33             }
34         }
35         return pessoaFisica;
36     }
37
38     public List<PessoaFisica> getPessoas() throws SQLException {
39         List<PessoaFisica> pessoasFisicas = new ArrayList<>();
40         String sql = "SELECT p.idpessoa, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, pf.cpf " +
41             "FROM pessoa p JOIN pessoa_fisica pf ON p.idpessoa = pf.pessoa_idpessoa";
42
43         try (Connection conn = ConnectorBD.getConnection();
44             Statement stmt = conn.createStatement()) {
45             ResultSet rs = stmt.executeQuery(sql);
46             while (rs.next()) {
47                 PessoaFisica p = new PessoaFisica(
48                     rs.getInt("idpessoa"),
49                     rs.getString("nome"),
50                     rs.getString("logradouro"),
51                     rs.getString("cidade"),
52                     rs.getString("estado"),
53                     rs.getString("telefone"),
54                     rs.getString("email"),
55                     rs.getString("cpf")
56                 );
57                 pessoasFisicas.add(p);
58             }
59         }
60         return pessoasFisicas;
61     }
62
63     public void incluir(PessoaFisica pessoaFisica) throws SQLException {
64         String sql = "INSERT INTO pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES (" +
65             "'" + pessoaFisica.getNome() + "', " +
66             "'" + pessoaFisica.getLogradouro() + "', " +
67             "'" + pessoaFisica.getCidade() + "', " +
68             "'" + pessoaFisica.getEstado() + "', " +
69             "'" + pessoaFisica.getTelefone() + "', " +
70             "'" + pessoaFisica.getEmail() + "')";
71         try (Connection conn = ConnectorBD.getConnection();
72             PreparedStatement pstmt = ConnectorBD.getPreparedStatement(conn, sql)) {
73             pstmt.executeUpdate();
74             int id = SequenciaManager.getId("pessoa_idpessoa");
75             sql = "INSERT INTO pessoa_fisica (pessoa_idpessoa, cpf) VALUES (" +
76                 id + ", " +
77                 "'" + pessoaFisica.getCpf() + "')";
78             pstmt = ConnectorBD.getPreparedStatement(conn, sql);
79             pstmt.executeUpdate();
80         }
81     }
82
83     public void alterar(PessoaFisica pessoaFisica) throws SQLException {
84         String sql = "UPDATE pessoa SET nome = '" + pessoaFisica.getNome() + "', logradouro = '" +
85             pessoaFisica.getLogradouro() + "', cidade = '" + pessoaFisica.getCidade() + "', estado = '" +
86             pessoaFisica.getEstado() + "', telefone = '" + pessoaFisica.getTelefone() + "', email = '" +
87             pessoaFisica.getEmail() + "' WHERE idpessoa = " + pessoaFisica.getIdpessoa();
88         try (Connection conn = ConnectorBD.getConnection();
89             PreparedStatement pstmt = ConnectorBD.getPreparedStatement(conn, sql)) {
90             pstmt.executeUpdate();
91         }
92     }
93
94     public void excluir(int id) throws SQLException {
95         String sql = "DELETE FROM pessoa_fisica WHERE idpessoa = " + id;
96         try (Connection conn = ConnectorBD.getConnection();
97             PreparedStatement pstmt = ConnectorBD.getPreparedStatement(conn, sql)) {
98             pstmt.executeUpdate();
99         }
100         sql = "DELETE FROM pessoa WHERE idpessoa = " + id;
101         try (Connection conn = ConnectorBD.getConnection();
102             PreparedStatement pstmt = ConnectorBD.getPreparedStatement(conn, sql)) {
103             pstmt.executeUpdate();
104         }
105     }
106 }
```

- b. **Classe PessoaJuridicaDAO**, com os métodos **getPessoa**, retornando uma pessoa jurídica a partir do seu id, **getPessoas**, para retorno de todas as pessoas jurídicas do banco de dados, incluir, para inclusão de uma pessoa jurídica, fornecida como parâmetro, nas tabelas Pessoa e PessoaJuridica, alterar, para alteração dos dados de uma pessoa jurídica, e excluir, para remoção da pessoa do banco em ambas as tabelas.



```
1 package cadastro.model;
2
3 import cadastro.model.util.ConnectorBD;
4 import cadastro.model.util.SequenciaManager;
5
6 import java.sql.*;
7 import java.util.ArrayList;
8 import java.util.List;
9
10 public class PessoaJuridicaDAO {
11
12     public PessoaJuridica getPessoa(int id) throws SQLException {
13         PessoaJuridica pessoaJuridica = null;
14         String sql = "SELECT p.idpessoa, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, pj.cnpj " +
15             "FROM pessoa p JOIN pessoa_juridica pj ON p.idpessoa = pj.pessoa_idpessoa WHERE p.idpessoa = ?";
16
17         try (Connection conn = ConnectorBD.getConnection();
18             PreparedStatement pstmt = ConnectorBD.getPreparedStatement(conn, sql)) {
19             pstmt.setInt(1, id);
20             try (ResultSet rs = pstmt.executeQuery()) {
21                 if (rs.next()) {
22                     pessoaJuridica = new PessoaJuridica(
23                         rs.getInt("idpessoa"),
24                         rs.getString("nome"),
25                         rs.getString("logradouro"),
26                         rs.getString("cidade"),
27                         rs.getString("estado"),
28                         rs.getString("telefone"),
29                         rs.getString("email"),
30                         rs.getString("cnpj")
31                     );
32                 }
33             }
34         }
35         return pessoaJuridica;
36     }
37
38     public List<PessoaJuridica> getPessoas() throws SQLException {
39         List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
40         String sql = "SELECT p.idpessoa, p.nome, p.logradouro, p.cidade, p.estado, p.telefone, p.email, pj.cnpj " +
41             "FROM pessoa p JOIN pessoa_juridica pj ON p.idpessoa = pj.pessoa_idpessoa";
42
43         try (Connection conn = ConnectorBD.getConnection();
44             Statement stmt = conn.createStatement()) {
45             ResultSet rs = stmt.executeQuery(sql);
46             while (rs.next()) {
47                 PessoaJuridica p = new PessoaJuridica(
48                     rs.getInt("idpessoa"),
49                     rs.getString("nome"),
50                     rs.getString("logradouro"),
51                     rs.getString("cidade"),
52                     rs.getString("estado"),
53                     rs.getString("telefone"),
54                     rs.getString("email"),
55                     rs.getString("cnpj")
56                 );
57                 pessoasJuridicas.add(p);
58             }
59         }
60         return pessoasJuridicas;
61     }
62
63     public void incluir(PessoaJuridica pessoaJuridica) throws SQLException {
64         String sql = "INSERT INTO pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES (" +
65             "'" + pessoaJuridica.getNome() + "', " +
66             "'" + pessoaJuridica.getLogradouro() + "', " +
67             "'" + pessoaJuridica.getCidade() + "', " +
68             "'" + pessoaJuridica.getEstado() + "', " +
69             "'" + pessoaJuridica.getTelefone() + "', " +
70             "'" + pessoaJuridica.getEmail() + "')";
71         try (Connection conn = ConnectorBD.getConnection();
72             PreparedStatement pstmt = ConnectorBD.getPreparedStatement(conn, sql)) {
73             pstmt.executeUpdate();
74             int id = SequenciaManager.getId("pessoa_idpessoa");
75             sql = "INSERT INTO pessoa_juridica (pessoa_idpessoa, cnpj) VALUES (" +
76                 id + ", " +
77                 "'" + pessoaJuridica.getCnpj() + "')";
78             pstmt = ConnectorBD.getPreparedStatement(conn, sql);
79             pstmt.executeUpdate();
80         }
81     }
82
83     public void alterar(PessoaJuridica pessoaJuridica) throws SQLException {
84         String sql = "UPDATE pessoa SET nome = '" + pessoaJuridica.getNome() + "', logradouro = '" +
85             pessoaJuridica.getLogradouro() + "', cidade = '" + pessoaJuridica.getCidade() + "', estado = '" +
86             pessoaJuridica.getEstado() + "', telefone = '" + pessoaJuridica.getTelefone() + "', email = '" +
87             pessoaJuridica.getEmail() + "' WHERE idpessoa = " + pessoaJuridica.getIdpessoa();
88         try (Connection conn = ConnectorBD.getConnection();
89             PreparedStatement pstmt = ConnectorBD.getPreparedStatement(conn, sql)) {
90             pstmt.executeUpdate();
91         }
92     }
93
94     public void excluir(int id) throws SQLException {
95         String sql = "DELETE FROM pessoa_juridica WHERE idpessoa = " + id;
96         try (Connection conn = ConnectorBD.getConnection();
97             PreparedStatement pstmt = ConnectorBD.getPreparedStatement(conn, sql)) {
98             pstmt.executeUpdate();
99         }
100         sql = "DELETE FROM pessoa WHERE idpessoa = " + id;
101         try (Connection conn = ConnectorBD.getConnection();
102             PreparedStatement pstmt = ConnectorBD.getPreparedStatement(conn, sql)) {
103             pstmt.executeUpdate();
104         }
105     }
106 }
```

c. Utilizar nas classes objetos dos tipos ConectorBD e SequenceManager.

6) Criar uma classe principal de testes com o nome CadastroBDTeste, efetuando as operações seguintes no método main:

- a. Instanciar uma pessoa física e persistir no banco de dados.
- b. Alterar os dados da pessoa física no banco.
- c. Consultar todas as pessoas físicas do banco de dados e listar no console.
- d. Excluir a pessoa física criada anteriormente no banco.
- e. Instanciar uma pessoa jurídica e persistir no banco de dados.
- f. Alterar os dados da pessoa jurídica no banco.
- g. Consultar todas as pessoas jurídicas do banco e listar no console.
- h. Excluir a pessoa jurídica criada anteriormente no banco.

```
try {
    PessoaFisica pessoaFisica = new PessoaFisica(id: 0, nome: "João Silva", logradouro: "Rua A", cidade: "Cidade B", estado: "Estado C", telefone: "123456789", em
    pessoaFisicaDAO.incluir(pessoaFisica);
    System.out.println("Pessoa física incluída.");

    pessoaFisica.setNome(joao_da_silva: "João da Silva");
    pessoaFisicaDAO.alterar(pessoaFisica);
    System.out.println("Pessoa física alterada.");

    List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
    System.out.println("Lista de pessoas físicas:");
    for (PessoaFisica pf : pessoasFisicas) {
        System.out.println(pf);
    }

    pessoaFisicaDAO.excluir(id: pessoaFisica.getId());
    System.out.println("Pessoa física excluída.");

    PessoaJuridica pessoaJuridica = new PessoaJuridica(id: 0, nome: "Empresa X", logradouro: "Avenida Y", cidade: "Cidade Z", estado: "Estado W", telefone: "98765
    pessoaJuridicaDAO.incluir(pessoaJuridica);
    System.out.println("Pessoa jurídica incluída.");

    pessoaJuridica.setNome(empresa_xyz: "Empresa XYZ");
    pessoaJuridicaDAO.alterar(pessoaJuridica);
    System.out.println("Pessoa jurídica alterada.");

    List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoas();
    System.out.println("Lista de pessoas jurídicas:");
    for (PessoaJuridica pj : pessoasJuridicas) {
        System.out.println(pj);
    }

    pessoaJuridicaDAO.excluir(id: pessoaJuridica.getId());
    System.out.println("Pessoa jurídica excluída.");

} catch (SQLException e) {
    e.printStackTrace();
}
```

7) A saída do sistema deverá ser semelhante à que é apresentada a seguir:

Relatório discente de acompanhamento

- 1. Título da Prática;**
Mapeamento Objeto-Relacional e DAO
- 2. 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
- 3. Todos os códigos solicitados neste roteiro de aula;**
- 4. Os resultados da execução dos códigos também devem ser apresentados;**
- 5. Análise e Conclusão:**
 - a. Qual a importância dos componentes de middleware, como o JDBC?**
 - b. Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?**
 - c. Como o padrão DAO melhora a manutenibilidade do software?**
 - d. Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?**

Endereço do projeto no GITHUB

https://github.com/BrenoSouza2023/Missao-Pratica_N-vel-3-Mundo-3.git