

Missão prática | Nível 1 | Mundo 3



Estácio

CAMPUS: POLO DISTRITO JK - ANÁPOLIS - GO

CURSO: DESENVOLVIMENTO FULL STACK

NÚMERO DA TURMA: 2024.3

SEMESTRE LETIVO: Segundo semestre de 2024

ALUNO: Henrique Rodrigues Rabello Vieira

MATRÍCULA: 202301230527

RPG0016 - BackEnd sem banco não tem

Objetivos:

- 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.

Primeiro procedimento:

Códigos solicitados:

```
CadastroBD.java x CadastroBDTeste.java x Pessoa.java x
Source History
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package cadastrobd.model;
6
7  /**
8   *
9   * @author Camz
10  */
11
12  public class Pessoa {
13
14      protected int id;
15      protected String nome;
16      protected String logradouro;
17      protected String cidade;
18      protected String estado;
19      protected String telefone;
20      protected String email;
21
22      public Pessoa() {
23      }
24
25      public Pessoa(int id, String nome, String logradouro, String cidade,
26          String estado, String telefone, String email) {
27          this.id = id;
28          this.nome = nome;
29          this.logradouro = logradouro;
30          this.cidade = cidade;
31          this.estado = estado;
32          this.telefone = telefone;
33          this.email = email;
34      }
35
36      public int getId() {
37          return id;
38      }
39
40      public void setId(int id) {
41          this.id = id;
42      }
43
44      public String getNome() {
```

```
45      }
46
47      public void setNome(String nome) {
48          this.nome = nome;
49      }
50
51      public String getLogradouro() {
52          return logradouro;
53      }
54
55      public void setLogradouro(String logradouro) {
56          this.logradouro = logradouro;
57      }
58
59      public String getCidade() {
60          return cidade;
61      }
62
63      public void setCidade(String cidade) {
64          this.cidade = cidade;
65      }
66
67      public String getEstado() {
68          return estado;
69      }
70
71      public void setEstado(String estado) {
72          this.estado = estado;
73      }
74
75      public String getTelefone() {
76          return telefone;
77      }
78
79      public void setTelefone(String telefone) {
80          this.telefone = telefone;
81      }
82
83      public String getEmail() {
```

```
84          return email;
85      }
86
87      public void setEmail(String email) {
88          this.email = email;
89      }
90
91      public void exibir(){
92          System.out.println("-----");
93          System.out.println("ID: " + id);
94          System.out.println("Nome: " + nome);
95          System.out.println("Logradouro: " + logradouro);
96          System.out.println("Cidade: " + cidade);
97          System.out.println("Estado: " + estado);
98          System.out.println("Telefone: " + telefone);
99          System.out.println("Email: " + email);
100      }
101  }
```

```
CadastroBD.java X CadastroBDTeste.java X PessoaFisica.java X
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package cadastrbd.model;
6
7   /**
8    *
9    * @author Camz
10   */
11   public class PessoaFisica extends Pessoa {
12
13       protected String cpf;
14
15       public PessoaFisica() {
16
17       }
18
19       public PessoaFisica(int id, String nome, String logradouro, String cidade,
20           String estado, String telefone, String email, String cpf) {
21           super(id, nome, logradouro, cidade, estado, telefone, email);
22           this.cpf = cpf;
23       }
24
25       public String getCpf() {
26           return cpf;
27       }
28
29       public void setCpf(String cpf) {
30           this.cpf = cpf;
31       }
32
33       @Override
34       public void exibir() {
35           super.exibir();
36           System.out.println("CPF: " + cpf);
37       }
38   }
```

```
CadastroBD.java X CadastroBDTeste.java X PessoaFisica.java X PessoaJuridica.java X
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package cadastrbd.model;
6
7   /**
8    *
9    * @author Camz
10   */
11   public class PessoaJuridica extends Pessoa {
12
13       protected String cnpj;
14
15       public PessoaJuridica() {
16
17       }
18
19       public PessoaJuridica(int id, String nome, String logradouro, String cidade,
20           String estado, String telefone, String email, String cnpj) {
21           super(id, nome, logradouro, cidade, estado, telefone, email);
22           this.cnpj = cnpj;
23       }
24
25       public String getCnpj() {
26           return cnpj;
27       }
28
29       public void setCnpj(String cnpj) {
30           this.cnpj = cnpj;
31       }
32
33       @Override
34       public void exibir() {
35           super.exibir();
36           System.out.println("CPF: " + cnpj);
37       }
38   }
```

```
CadastroBD.java X CadastroBDTeste.java X PessoaFisica.java X PessoaJuridica.java X ConectorBD.java X
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package cadastrbd.model.util;
6
7   import java.sql.Connection;
8   import java.sql.DriverManager;
9   import java.sql.PreparedStatement;
10  import java.sql.ResultSet;
11  import java.sql.SQLException;
12  import java.sql.Statement;
13
14  /**
15   *
16   * @author Camz
17   */
18
19  public class ConectorBD {
20
21      private static final String URL = "jdbc:sqlserver://localhost:1433;databaseName=Loja;"
22          + "encrypt=true;trustServerCertificate=true";
23      private static final String USER = "loja";
24      private static final String PASSWORD = "loja";
25
26      public static Connection getConnection() throws SQLException {
27          return DriverManager.getConnection(URL, USER, PASSWORD);
28      }
29
30      public static PreparedStatement getPrepared(String sql) throws SQLException {
31          return getConnection().prepareStatement(sql);
32      }
33
34      public static ResultSet getSelect(PreparedStatement stmt) throws SQLException {
35          return stmt.executeQuery();
36      }
37
38      public static void close(Connection conn) throws SQLException {
39          if (conn != null) {
40              conn.close();
41          }
42      }
43  }
```

```

42
43     public static void close(Statement stmt) throws SQLException {
44         if (stmt != null) {
45             stmt.close();
46         }
47     }
48
49     public static void close(ResultSet rs) throws SQLException {
50         if (rs != null) {
51             rs.close();
52         }
53     }
54 }

```

```

CadastroBD.java x CadastroBDTeste.java x PessoaFisica.java x PessoaJuridica.java x ConectorBD.java x SequenceManager.java x
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package cadastrobd.model.util;
6
7   import java.sql.Connection;
8   import java.sql.PreparedStatement;
9   import java.sql.ResultSet;
10  import java.sql.SQLException;
11
12  /**
13   *
14   * @author Camz
15   */
16  public class SequenceManager {
17
18      public static int getValue(String sequenceName) throws SQLException {
19          String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS nextval";
20          try (Connection conn = ConectorBD.getConnection();
21              PreparedStatement stmt = conn.prepareStatement(sql);
22              ResultSet rs = stmt.executeQuery()) {
23              if (rs.next()) {
24                  return rs.getInt("nextval");
25              } else {
26                  throw new SQLException("Não foi possível obter o próximo valor da sequência "
27                      + sequenceName);
28              }
29          }
30      }
31  }

```

```

CadastroBD.java x CadastroBDTeste.java x PessoaFisica.java x PessoaJuridica.java x ConectorBD.java x SequenceManager.java x PessoaFisicaDAO.java x
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5
6   package cadastrobd.model;
7
8   import cadastrobd.model.util.ConectorBD;
9   import java.sql.Connection;
10  import java.sql.PreparedStatement;
11  import java.sql.ResultSet;
12  import java.sql.SQLException;
13  import java.util.ArrayList;
14  import java.util.List;
15
16  /**
17   *
18   * @author Camz
19   */
20
21  public class PessoaFisicaDAO {
22
23      public PessoaFisica getPessoa(int id) throws SQLException {
24          String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
25              "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaFisica.cpf " +
26              "FROM Pessoa " +
27              "JOIN PessoaFisica ON Pessoa.idPessoa = PessoaFisica.idPessoa " +
28              "WHERE Pessoa.idPessoa = ?";
29          try (Connection conn = ConectorBD.getConnection();
30              PreparedStatement stmt = ConectorBD.getPrepared(sql)) {
31              stmt.setInt(1, id);
32              try (ResultSet rs = ConectorBD.getSelect(stmt)) {
33                  if (rs.next()) {
34                      PessoaFisica pessoa = new PessoaFisica();
35                      pessoa.setId(rs.getInt("idPessoa"));
36                      pessoa.setNome(rs.getString("nome"));
37                      pessoa.setLogradouro(rs.getString("logradouro"));
38                      pessoa.setCidade(rs.getString("cidade"));
39                      pessoa.setEstado(rs.getString("estado"));
40                      pessoa.setTelefone(rs.getString("telefone"));
41                      pessoa.setEmail(rs.getString("email"));
42                      pessoa.setCpf(rs.getString("cpf"));
43                      return pessoa;
44                  }
45              }
46          }
47          return null;
48      }

```

```
CadastroBD.java x CadastroBDTeste.java x PessoaFisica.java x PessoaJuridica.java x ConectorBD.java x SequenceManager.java x PessoaFisicaDAO.java x
Source History
45
46 public List<PessoaFisica> getPessoas() throws SQLException {
47     List<PessoaFisica> pessoas = new ArrayList<>();
48     String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
49         "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaFisica.cpf " +
50         "FROM Pessoa " +
51         "JOIN PessoaFisica ON Pessoa.idPessoa = PessoaFisica.idPessoa";
52     try (Connection conn = ConectorBD.getConnection();
53         PreparedStatement stmt = ConectorBD.getPrepared(sql);
54         ResultSet rs = ConectorBD.getSelect(stmt)) {
55         while (rs.next()) {
56             PessoaFisica pessoa = new PessoaFisica();
57             pessoa.setId(rs.getInt("idPessoa"));
58             pessoa.setNome(rs.getString("nome"));
59             pessoa.setLogradouro(rs.getString("logradouro"));
60             pessoa.setCidade(rs.getString("cidade"));
61             pessoa.setEstado(rs.getString("estado"));
62             pessoa.setTelefone(rs.getString("telefone"));
63             pessoa.setEmail(rs.getString("email"));
64             pessoa.setCpf(rs.getString("cpf"));
65             pessoas.add(pessoa);
66         }
67     }
68     return pessoas;
69 }
70
71 public void incluir(PessoaFisica pessoa) throws SQLException {
72     String sqlInsertPessoa = "INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade,"
73     + " estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
74     String sqlInsertPessoaFisica = "INSERT INTO PessoaFisica (idPessoa, cpf)"
75     + " VALUES (?, ?)";
76     try (Connection conn = ConectorBD.getConnection();
77         PreparedStatement stmtInsertPessoa = conn.prepareStatement(sqlInsertPessoa);
78         PreparedStatement stmtInsertPessoaFisica = conn.prepareStatement(sqlInsertPessoaFisica)) {
79         // Inserir na tabela Pessoa
80         stmtInsertPessoa.setInt(1, pessoa.getId());
81         stmtInsertPessoa.setString(2, pessoa.getNome());
82         stmtInsertPessoa.setString(3, pessoa.getLogradouro());
83         stmtInsertPessoa.setString(4, pessoa.getCidade());
84         stmtInsertPessoa.setString(5, pessoa.getEstado());
85         stmtInsertPessoa.setString(6, pessoa.getTelefone());
86         stmtInsertPessoa.setString(7, pessoa.getEmail());
87         stmtInsertPessoa.executeUpdate();
88     }
89 }
```

```
93
94 // Inserir na tabela PessoaFisica
95 stmtInsertPessoaFisica.setInt(1, pessoa.getId());
96 stmtInsertPessoaFisica.setString(2, pessoa.getCpf());
97 stmtInsertPessoaFisica.executeUpdate();
98 }
99
100
101 public void alterar(PessoaFisica pessoa, String novoNome, String novoLogradouro, String novaCidade,
102 String novoEstado, String novoTelefone, String novoEmail, String novoCpf) throws SQLException {
103     String sql = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, "
104     + "estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
105     String sqlFisica = "UPDATE PessoaFisica SET cpf = ? WHERE idPessoa = ?";
106     try (Connection conn = ConectorBD.getConnection();
107         PreparedStatement stmt = conn.prepareStatement(sql);
108         PreparedStatement stmtFisica = conn.prepareStatement(sqlFisica)) {
109         // Atualizar dados na tabela Pessoa
110         stmt.setString(1, novoNome);
111         stmt.setString(2, novoLogradouro);
112         stmt.setString(3, novaCidade);
113         stmt.setString(4, novoEstado);
114         stmt.setString(5, novoTelefone);
115         stmt.setString(6, novoEmail);
116         stmt.setInt(7, pessoa.getId());
117         stmt.executeUpdate();
118         // Atualizar CPF na tabela PessoaFisica
119         stmtFisica.setString(1, novoCpf);
120         stmtFisica.setInt(2, pessoa.getId());
121         stmtFisica.executeUpdate();
122     }
123 }
124
125 public void excluir(int id) throws SQLException {
126     String sql = "DELETE FROM PessoaFisica WHERE idPessoa = ?";
127     String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
128     try (Connection conn = ConectorBD.getConnection();
129         PreparedStatement stmt = conn.prepareStatement(sql);
130         PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
131         // Excluir da tabela PessoaFisica
132         stmt.setInt(1, id);
133         stmt.executeUpdate();
134     }
135 }
```

```
136 // Excluir da tabela Pessoa
137 stmtPessoa.setInt(1, id);
138 stmtPessoa.executeUpdate();
139
140 System.out.println("Pessoa fisica excluida com ID: " + id);
141 }
142 }
143
144 }
```

```
CadastroBD.java x CadastroBDTeste.java x PessoaFisica.java x PessoaJuridica.java x ConectorBD.java x SequenceManager.java x PessoaFisicaDAO.java x PessoaJuridicaDAO.java x
Source History
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5   package cadastrobd.model;
6
7   import cadastrobd.model.util.ConectorBD;
8   import java.sql.Connection;
9   import java.sql.PreparedStatement;
10  import java.sql.ResultSet;
11  import java.sql.SQLException;
12  import java.util.ArrayList;
13  import java.util.List;
14
15  /**
16   *
17   * @author Camr
18   */
19  public class PessoaJuridicaDAO {
20
21      public PessoaJuridica getPessoa(int id) throws SQLException {
22          String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
23                      "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaJuridica.cnpj " +
24                      "FROM Pessoa " +
25                      "JOIN PessoaJuridica ON Pessoa.idPessoa = PessoaJuridica.idPessoa " +
26                      "WHERE Pessoa.idPessoa = ?";
27
28          try (Connection conn = ConectorBD.getConnection();
29               PreparedStatement stmt = ConectorBD.getPreparedStatement(sql)) {
30              stmt.setInt(1, id);
31              try (ResultSet rs = ConectorBD.getSelect(stmt)) {
32                  if (rs.next()) {
33                      PessoaJuridica pessoa = new PessoaJuridica();
34                      pessoa.setId(rs.getInt("idPessoa"));
35                      pessoa.setNome(rs.getString("nome"));
36                      pessoa.setLogradouro(rs.getString("logradouro"));
37                      pessoa.setCidade(rs.getString("cidade"));
38                      pessoa.setEstado(rs.getString("estado"));
39                      pessoa.setTelefone(rs.getString("telefone"));
40                      pessoa.setEmail(rs.getString("email"));
41                      pessoa.setCnpj(rs.getString("cnpj"));
42                      return pessoa;
43                  }
44              }
45          }
46          return null;
47      }
48  }
```

```
47
48  public List<PessoaJuridica> getPessoas() throws SQLException {
49      List<PessoaJuridica> pessoas = new ArrayList<>();
50      String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
51                  "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaJuridica.cnpj " +
52                  "FROM Pessoa " +
53                  "JOIN PessoaJuridica ON Pessoa.idPessoa = PessoaJuridica.idPessoa";
54
55      try (Connection conn = ConectorBD.getConnection();
56           PreparedStatement stmt = ConectorBD.getPreparedStatement(sql);
57           ResultSet rs = ConectorBD.getSelect(stmt)) {
58          while (rs.next()) {
59              PessoaJuridica pessoa = new PessoaJuridica();
60              pessoa.setId(rs.getInt("idPessoa"));
61              pessoa.setNome(rs.getString("nome"));
62              pessoa.setLogradouro(rs.getString("logradouro"));
63              pessoa.setCidade(rs.getString("cidade"));
64              pessoa.setEstado(rs.getString("estado"));
65              pessoa.setTelefone(rs.getString("telefone"));
66              pessoa.setEmail(rs.getString("email"));
67              pessoa.setCnpj(rs.getString("cnpj"));
68              pessoas.add(pessoa);
69          }
70      }
71      return pessoas;
72  }
73
74  public void inclui(PessoaJuridica pessoa) throws SQLException {
75      String sqlInsertPessoa = "INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade, " +
76                              "estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
77      String sqlInsertPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoa, cnpj) " +
78                                      "VALUES (?, ?)";
79
80      try (Connection conn = ConectorBD.getConnection();
81           PreparedStatement stmtInsertPessoa = conn.prepareStatement(sqlInsertPessoa);
82           PreparedStatement stmtInsertPessoaJuridica = conn.prepareStatement(sqlInsertPessoaJuridica)) {
83
84          // Inserir na tabela Pessoa
85          stmtInsertPessoa.setInt(1, pessoa.getId());
86          stmtInsertPessoa.setString(2, pessoa.getNome());
87          stmtInsertPessoa.setString(3, pessoa.getLogradouro());
88          stmtInsertPessoa.setString(4, pessoa.getCidade());
89          stmtInsertPessoa.setString(5, pessoa.getEstado());
90          stmtInsertPessoa.setString(6, pessoa.getTelefone());
91          stmtInsertPessoa.setString(7, pessoa.getEmail());
92          stmtInsertPessoa.executeUpdate();
93      }
```

```

92 // Inserir na tabela PessoaJuridica
93 stmtInsertPessoaJuridica.setInt(1, pessoa.getId());
94 stmtInsertPessoaJuridica.setString(2, pessoa.getCnpj());
95 stmtInsertPessoaJuridica.executeUpdate();
96 }
97 }
98
99 public void alterar(PessoaJuridica pessoa, String novoNome, String novoLogradouro, String novaCidade,
100 String novoEstado, String novoTelefone, String novoEmail, String novoCnpj) throws SQLException {
101 String sql = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, "
102 + "estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
103 String sqlJuridica = "UPDATE PessoaJuridica SET cnpj = ? WHERE idPessoa = ?";
104
105 try (Connection conn = ConectorBD.getConnection();
106 PreparedStatement stmt = conn.prepareStatement(sql);
107 PreparedStatement stmtJuridica = conn.prepareStatement(sqlJuridica)) {
108
109 // Atualizar dados na tabela Pessoa
110 stmt.setString(1, novoNome);
111 stmt.setString(2, novoLogradouro);
112 stmt.setString(3, novaCidade);
113 stmt.setString(4, novoEstado);
114 stmt.setString(5, novoTelefone);
115 stmt.setString(6, novoEmail);
116 stmt.setInt(7, pessoa.getId());
117 stmt.executeUpdate();
118
119 // Atualizar CNPJ na tabela PessoaJuridica
120 stmtJuridica.setString(1, novoCnpj);
121 stmtJuridica.setInt(2, pessoa.getId());
122 stmtJuridica.executeUpdate();
123 }
124 }
125
126 public void excluir(int id) throws SQLException {
127 String sql = "DELETE FROM PessoaJuridica WHERE idPessoa = ?";
128 String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
129
130 try (Connection conn = ConectorBD.getConnection();
131 PreparedStatement stmt = conn.prepareStatement(sql);
132 PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
133 // Excluir da tabela PessoaJuridica
134 stmt.setInt(1, id);
135 stmt.executeUpdate();
136 }

```

```

137 // Excluir da tabela Pessoa
138 stmtPessoa.setInt(1, id);
139 stmtPessoa.executeUpdate();
140
141 System.out.println("Pessoa juridica excluida com ID: " + id);
142 }
143 }
144 }

```

```

CadastroBD.java x CadastroBDTeste.java x PessoaFisica.java x PessoaJuridica.java x ConectorBD.java x SequenceManager.java x PessoaFisicaDAO.java x PessoaJuridicaDAO.java x
Source History
1 package cadastrobd;
2
3 /*
4  * Click https://nhost/SystemFileSystem/Templates/licenses/license-default.txt to change this license
5  * Click https://nhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
6  */
7
8
9 import java.sql.SQLException;
10 import java.util.List;
11 import cadastrobd.model.PessoaFisica;
12 import cadastrobd.model.PessoaFisicaDAO;
13 import cadastrobd.model.PessoaJuridica;
14 import cadastrobd.model.PessoaJuridicaDAO;
15 import cadastrobd.model.util.ConectorBD;
16 import java.sql.Connection;
17
18 /**
19  *
20  * @author Camz
21  */
22 public class CadastroBDTeste {
23
24     public static void main(String[] args) {
25         try {
26             Connection conn = ConectorBD.getConnection();
27             PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
28             PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();
29
30             // Criar uma pessoa fisica
31             PessoaFisica pf = new PessoaFisica(6, "Pedro", "Rua A, 10", "Atibaia", "SP",
32                 "1234-5678", "pedro@gmail.com", "12345678900");
33
34             // Persistir a pessoa fisica no banco de dados
35             pfDAO.incluir(pf);
36             System.out.println("Pessoa fisica criada:");
37             pf.exibir();
38             System.out.println();
39
40             // Alterar os dados da pessoa fisica no banco
41             pfDAO.alterar(pf, "Pedro Alves", "Rua B, 11", "Atibaia", "SP",
42                 "9999-8888", "pedro.alves@email.com", "12345678900");
43             System.out.println("Dados da pessoa fisica alterados.");
44             System.out.println("-----");
45             System.out.println();
46             System.out.println();

```

```

47 // Consultar todas as pessoas fisicas do banco de dados e listar no console
48 List<PessoaFisica> pessoasFisicas = pfDAO.getPessoas();
49 System.out.println("Todas as pessoas fisicas:");
50 for (PessoaFisica pessoaFisica : pessoasFisicas) {
51     pessoaFisica.exibir();
52 }
53 System.out.println();
54
55 // Excluir a pessoa fisica criada anteriormente no banco
56 System.out.println("-----");
57 pfDAO.excluir(pf.getId());
58 System.out.println("-----");
59 System.out.println();
60
61 // Criar uma pessoa juridica
62 PessoaJuridica pj = new PessoaJuridica(7, "Empresa ABC", "Av. Principal, 100",
63     "Sao Paulo", "SP", "1234-5678", "empresa@abc.com", "12345678901234");
64
65 // Persistir a pessoa juridica no banco de dados
66 pjDAO.incluir(pj);
67 System.out.println("Pessoa juridica criada:");
68 pj.exibir();
69 System.out.println();
70
71 // Alterar os dados da pessoa juridica no banco
72 pjDAO.alterar(pj, "Companhia ABC", "Av. Nova, 200", "Rio de Janeiro", "RJ",
73     "9876-5432", "companhia@abc.com", "98765432109876");
74 System.out.println("-----");
75 System.out.println("Dados da pessoa juridica alterados.");
76 System.out.println("-----");
77 System.out.println();
78
79 // Consultar todas as pessoas juridicas do banco de dados e listar no console
80 List<PessoaJuridica> pessoasJuridicas = pjDAO.getPessoas();
81 System.out.println("Todas as pessoas juridicas:");
82 for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
83     pessoaJuridica.exibir();
84 }
85 System.out.println();
86
87 // Excluir a pessoa juridica criada anteriormente no banco
88 System.out.println("-----");
89 pjDAO.excluir(pj.getId());
90 System.out.println("-----");
91 System.out.println();
92
93
94
95
96 } catch (SQLException e) {
97     System.out.println("Ocorreu um erro: " + e.getMessage());
98 }
99 }
100 }

```

Resultados:

```

Pessoa juridica criada:
=====
ID: 7
Nome: Empresa ABC
Logradouro: Av. Principal, 100
Cidade: Sao Paulo
Estado: SP
Telefone: 1234-5678
E-mail: empresa@abc.com
CPF: 12345678901234

=====
Dados da pessoa juridica alterados.
=====

Todas as pessoas juridicas
=====
ID: 4
Nome: Distribuidora Diamante
Logradouro: Avenida A, 40
Cidade: Curitiba
Estado: PR
Telefone: 4444-4444
E-mail: diamante@gmail.com
CPF: 444444444444
=====
ID: 5
Nome: Empresa Estrela
Logradouro: Avenida B, 50
Cidade: Recife
Estado: PE
Telefone: 5555-5555
E-mail: estrela@gmail.com
CPF: 555555555555
=====
ID: 7
Nome: Companhia ABC
Logradouro: Av. Nova, 200
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 9876-5432
E-mail: companhia@abc.com
CPF: 98765432109876
=====
Pessoa juridica excluida com ID: 7

```



```

run:
Pessoa fisica criada:
=====
ID: 6
Nome: Pedro
Logradouro: Rua A, 10
Cidade: Atibaia
Estado: SP
Telefone: 1234-5678
E-mail: pedro@gmail.com
CPF: 12345678910
=====
Dados da pessoa fisica alterados.
=====

Todas as pessoas fisicas
=====
ID: 4
Nome: Alana
Logradouro: Rua X, 10
Cidade: Manaus
Estado: AM
Telefone: 1111-1111
E-mail: alana@gmail.com
CPF: 11111111111
=====
ID: 2
Nome: Bruno
Logradouro: Rua Y, 20
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 2222-2222
E-mail: bruno@gmail.com
CPF: 22222222222
=====
ID: 3
Nome: Caio
Logradouro: Rua Z, 30
Cidade: Porto Alegre
Estado: RS
Telefone: 3333-3333
E-mail: caio@gmail.com
CPF: 33333333333

```

```

=====
ID: 6
Nome: Pedro Alves
Logradouro: Rua B, 11
Cidade: Atibaia
Estado: SP
Telefone: 9999-8888
E-mail: pedro.alves@email.com
CPF: 12345678900
=====
Pessoa fisica excluida com ID: 6
=====

```

Análise e conclusão:

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

O JDBC é importante para a comunicação entre a aplicação JAVA e o banco de dados .

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

O preparedStatement é mais seguro e eficiente que statement, pois ele evita a injeção de SQL e permite a definição de parâmetros.

Como o padrão DAO melhora a manutenibilidade do software?

Melhora separando a lógica de acesso da lógica de negócios.

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

Pode ser feita de diversas formas, todas as classes compartilhando uma tabela, cada classe ter sua própria tabela, tendo apenas campos exclusivos e uma chave estrangeira para a tabela da superclasse, ou cada subclasse ter sua tabela o que impacta diretamente o sistema, cada um tendo as suas vantagens.

Segundo procedimento:

Códigos solicitados:

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
4  */
5  package cadastrobd;
6
7  import cadastrobd.model.PessoaFisica;
8  import cadastrobd.model.PessoaFisicaDAO;
9  import cadastrobd.model.PessoaJuridica;
10 import cadastrobd.model.PessoaJuridicaDAO;
11 import java.util.Scanner;
12 import java.sql.SQLException;
13 import java.util.ArrayList;
14
15 /**
16 *
17 * @author Camz
18 */
19
20 public class CadastroBD {
21
22     private static final Scanner sc = new Scanner(System.in);
23     private static final PessoaFisicaDAO pfDao = new PessoaFisicaDAO();
24     private static final PessoaJuridicaDAO pjDao = new PessoaJuridicaDAO();
25
26     public static void main(String[] args) {
27         int opcao = -1;
28         while (opcao != 0) {
29             printMenu();
30             opcao = inputInt("ESCOLHA: ");
31             switch (opcao) {
32                 case 1 -> incluir();
33                 case 2 -> alterar();
34                 case 3 -> excluir();
35                 case 4 -> buscarPeloId();
36                 case 5 -> exibirTodos();
37                 case 0 -> System.out.println("Finalizando...");
38                 default -> System.out.println("Escolha invalida!");
39             }
40         }
41     }
42 }
```

```
43 private static void printMenu() {
44     System.out.println("\n=====");
45     System.out.println("1 - Incluir");
46     System.out.println("2 - Alterar");
47     System.out.println("3 - Excluir");
48     System.out.println("4 - Buscar pelo ID");
49     System.out.println("5 - Exibir todos");
50     System.out.println("0 - Sair");
51     System.out.println("=====");
52 }
53
54 private static String input(String prompt) {
55     System.out.print(prompt);
56     return sc.nextLine();
57 }
58
59 private static int inputInt(String prompt) {
60     System.out.print(prompt);
61     try {
62         return Integer.parseInt(sc.nextLine());
63     } catch (NumberFormatException e) {
64         System.out.println("Erro: Entrada invalida. Tente novamente.");
65         return inputInt(prompt);
66     }
67 }
68
69 private static void incluir() {
70     System.out.println("\nIncluindo pessoa...");
71     System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
72     String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
73     Integer id = inputInt("Informe o ID: ");
74     switch (tipoPessoa) {
75         case "F" -> {
76             try {
77                 pfDao.incluir(criarPessoaFisica(id));
78                 System.out.println("Pessoa fisica incluida com sucesso!");
79             } catch (SQLException e) {
80                 System.out.println("Erro ao incluir pessoa fisica: " + e.getMessage());
81             }
82         }
83     }
```

```
83     case "J" -> {
84         try {
85             pjDao.incluir(criarPessoaJuridica(id));
86             System.out.println("Pessoa juridica incluida com sucesso!");
87         } catch (SQLException e) {
88             System.out.println("Erro ao incluir pessoa juridica: " + e.getMessage());
89         }
90     }
91     default -> System.out.println("Tipo de pessoa invalido!");
92 }
93
94 private static PessoaFisica criarPessoaFisica(Integer id) {
95     System.out.println("Criando Pessoa Fisica...");
96     String nome = input("Informe o nome: ");
97     String logradouro = input("Informe o logradouro: ");
98     String cidade = input("Informe a cidade: ");
99     String estado = input("Informe o estado: ");
100    String telefone = input("Informe o telefone: ");
101    String email = input("Informe o email: ");
102    String cpf = input("Informe o CPF: ");
103    return new PessoaFisica(id, nome, logradouro, cidade, estado, telefone, email, cpf);
104 }
105
106 private static PessoaJuridica criarPessoaJuridica(Integer id) {
107     System.out.println("Criando Pessoa Juridica...");
108     String nome = input("Informe o nome: ");
109     String logradouro = input("Informe o logradouro: ");
110     String cidade = input("Informe a cidade: ");
111     String estado = input("Informe o estado: ");
112     String telefone = input("Informe o telefone: ");
113     String email = input("Informe o email: ");
114     String cnpj = input("Informe o CNPJ: ");
115     return new PessoaJuridica(id, nome, logradouro, cidade, estado, telefone, email, cnpj);
116 }
117
118 }
```

```

119 private static void alterar() {
120     System.out.println("\nAlterando pessoa...");
121     System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
122     String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
123     if (tipoPessoa.equals("F")) {
124         try {
125             Integer id = inputInt("Informe o ID da Pessoa Fisica: ");
126             PessoaFisica pf = pfDao.getPessoa(id);
127             if (pf != null) {
128                 System.out.println("Dados atuais da Pessoa Fisica:");
129                 pf.exibir();
130
131                 String novoNome = input("Informe o novo nome: ");
132                 String novoLogradouro = input("Informe o novo logradouro: ");
133                 String novaCidade = input("Informe a nova cidade: ");
134                 String novoEstado = input("Informe o novo estado: ");
135                 String novoTelefone = input("Informe o novo telefone: ");
136                 String novoEmail = input("Informe o novo email: ");
137                 String novoCpf = input("Informe o novo CPF: ");
138
139                 pfDao.alterar(pf, novoNome, novoLogradouro, novaCidade,
140                             novoEstado, novoTelefone, novoEmail, novoCpf);
141                 System.out.println("Pessoa fisica alterada com sucesso!");
142             } else {
143                 System.out.println("ID errado!");
144             }
145         } catch (NullPointerException | SQLException e) {
146             System.out.println("Erro ao alterar pessoa fisica: " + e.getMessage());
147         }
148     } else if (tipoPessoa.equals("J")) {
149         try {
150             Integer id = inputInt("Informe o ID da Pessoa Juridica: ");
151             PessoaJuridica pj = pjDao.getPessoa(id);
152             if (pj != null) {
153                 System.out.println("Dados atuais da Pessoa Juridica:");
154                 pj.exibir();
155
156                 String novoNome = input("Informe o novo nome: ");
157                 String novoLogradouro = input("Informe o novo logradouro: ");
158                 String novaCidade = input("Informe a nova cidade: ");
159                 String novoEstado = input("Informe o novo estado: ");
160                 String novoTelefone = input("Informe o novo telefone: ");
161                 String novoEmail = input("Informe o novo email: ");
162                 String novoCnpj = input("Informe o novo CNPJ: ");

```

```

164         pjDao.alterar(pj, novoNome, novoLogradouro, novaCidade,
165                     novoEstado, novoTelefone, novoEmail, novoCnpj);
166         System.out.println("Pessoa juridica alterada com sucesso!");
167     } else {
168         System.out.println("ID errado!");
169     }
170 } catch (NullPointerException | SQLException e) {
171     System.out.println("Erro ao alterar pessoa juridica: " + e.getMessage());
172 }
173 } else {
174     System.out.println("Tipo de pessoa invalido!");
175 }
176 }
177
178 private static void excluir() {
179     System.out.println("\nExcluindo pessoa...");
180     System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
181     String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
182     switch (tipoPessoa) {
183         case "F" -> {
184             try {
185                 Integer id = inputInt("Informe o ID da Pessoa Fisica: ");
186                 PessoaFisica pf = pfDao.getPessoa(id);
187                 if (pf != null) {
188                     pfDao.excluir(pf.getId());
189                     System.out.println("Sucesso ao excluir!");
190                 } else {
191                     System.out.println("ID errado!");
192                 }
193             } catch (NullPointerException | SQLException e) {
194                 System.out.println("Erro ao excluir pessoa fisica: " + e.getMessage());
195             }
196         }
197         case "J" -> {
198             try {
199                 Integer id = inputInt("Informe o ID da Pessoa Juridica: ");
200                 PessoaJuridica pj = pjDao.getPessoa(id);
201                 if (pj != null) {
202                     pjDao.excluir(pj.getId());
203                     System.out.println("Sucesso ao excluir!");
204                 } else {
205                     System.out.println("ID errado!");
206                 }
207             } catch (NullPointerException | SQLException e) {
208                 System.out.println("Erro ao excluir pessoa juridica: " + e.getMessage());
209             }
210         }

```

```

211         default -> System.out.println("Tipo de pessoa invalido!");
212     }
213 }
214
215 private static void buscarPeloId() {
216     System.out.println("\nBuscando pessoa pelo ID...");
217     System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
218     String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
219     switch (tipoPessoa) {
220         case "F" -> {
221             try {
222                 Integer id = inputInt("Informe o ID da Pessoa Fisica: ");
223                 PessoaFisica pf = pfDao.getPessoa(id);
224                 if (pf != null) {
225                     pf.exibir();
226                 } else {
227                     System.err.println("Pessoa fisica com o ID " + id + " nao encontrada!");
228                 }
229             } catch (SQLException e) {
230                 System.err.println("Erro ao buscar pessoa fisica: " + e.getMessage());
231             }
232         }
233         case "J" -> {
234             try {
235                 Integer id = inputInt("Informe o ID da Pessoa Juridica: ");
236                 PessoaJuridica pj = pjDao.getPessoa(id);
237                 if (pj != null) {
238                     pj.exibir();
239                 } else {
240                     System.err.println("Pessoa juridica com o ID " + id + " nao encontrada!");
241                 }
242             } catch (SQLException e) {
243                 System.err.println("Erro ao buscar pessoa juridica: " + e.getMessage());
244             }
245         }
246         default -> System.out.println("Tipo de pessoa invalido!");
247     }
248 }
249

```

```

250 private static void exibirTodos() {
251     System.out.println("\nExibindo todas as pessoas...");
252     System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
253     String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
254     try {
255         switch (tipoPessoa) {
256             case "F" -> {
257                 ArrayList<PessoaFisica> listaPf = (ArrayList<PessoaFisica>) pfDao.getPessoas();
258                 for (PessoaFisica pessoa : listaPf) {
259                     pessoa.exibir();
260                 }
261             }
262             case "J" -> {
263                 ArrayList<PessoaJuridica> listaPj = (ArrayList<PessoaJuridica>) pjDao.getPessoas();
264                 for (PessoaJuridica pessoa : listaPj) {
265                     pessoa.exibir();
266                 }
267             }
268             default -> System.out.println("Tipo de pessoa invalido!");
269         }
270     } catch (SQLException e) {
271         System.out.println("Erro ao exibir pessoas: " + e.getMessage());
272     }
273 }
274

```

Resultados:

```

=====
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
=====
ESCOLHA: 1

Incluindo pessoa...
F - Pessoa Fisica | J - Pessoa Juridica
TIPO DE PESSOA: F
Informe o ID: 11
Criando pessoa fisica...
Informe o nome: Vera
Informe o logradouro: Rua A, 02
Informe a cidade: Rio de Janeiro
Informe o estado: RJ
Informe o telefone: 2222-1111
Informe o e-mail: vera@gmail.com
Informe o CPF: 11221122112
Pessoa fisica incluida com sucesso!

```

```

=====
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
=====
ESCOLHA: 2

Alterando pessoa...
F - Pessoa Fisica | J - Pessoa Juridica
TIPO DE PESSOA: F
Informe o ID da Pessoa Fisica: 11
Dados atuais da Pessoa Fisica:
=====
ID: 11
Nome: Vera
Logradouro: Rua A, 02
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 2222-1111
E-mail: vera@gmail.com
CPF: 11221122112
Informe o novo nome: Vera Duarte
Informe o novo logradouro: Rua A, 02
Informe a nova cidade: Rio de Janeiro
Informe o novo estado: RJ
Informe o novo telefone: 2222-1111
Informe o novo e-mail: veraduarte@gmail.com
Informe o novo CPF: 11221122112
Pessoa fisica alterada com sucesso!

```

```
=====
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
=====
ESCOLHA: 3

Excluindo pessoa...
P - Pessoa Fisica | J - Pessoa Juridica
TIPO DE PESSOA: J
Informe o ID da Pessoa Juridica: 15
Pessoa Juridica excluida com o ID: 15
Sucesso ao excluir!
```

```
=====
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
=====
ESCOLHA: 4

Buscando pessoa pelo ID...
P - Pessoa Fisica | J - Pessoa Juridica
TIPO DE PESSOA: J
Informe o ID da Pessoa Juridica: 15
```

Pessoa juridica com o ID 15 nao encontrada!

| Resultados | | Mensagens | | | | | | | |
|------------|----------|------------------------|---------------|----------------|--------|-----------|----------------------|-------------|---------------|
| | idPessoa | nome | logradouro | cidade | estado | telefone | email | cpf | cnpj |
| 1 | 1 | Alana | Rua X, 10 | Manaus | AM | 1111-1111 | alana@gmail.com | 11111111111 | NULL |
| 2 | 2 | Breno | Rua Y, 20 | Rio de Janeiro | RJ | 2222-2222 | breno@gmail.com | 22222222222 | NULL |
| 3 | 3 | Caio | Rua Z, 30 | Porto Alegre | RS | 3333-3333 | caio@gmail.com | 33333333333 | NULL |
| 4 | 4 | Distribuidora Diamante | Avenida A, 40 | Curitiba | PR | 4444-4444 | diamante@gmail.com | NULL | 4444444444444 |
| 5 | 5 | Empresa Estrela | Avenida B, 50 | Recife | PE | 5555-5555 | estrela@gmail.com | NULL | 5555555555555 |
| 6 | 10 | Carlos Silva | Rua A, 01 | Osasco | SP | 1111-2222 | carlos2024@gmail.com | 10101010101 | NULL |

Análise e conclusão:

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

A persistência em arquivo consiste em armazenar dados em arquivos, podendo ser em xml, json e até arquivos binários. Já em banco de dados as informações são armazenadas de forma estruturada, oferecendo vários recursos e sendo mais adequada para aplicativos em grande escala.

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

Reduzindo a necessidade de classes anônimas, simplificando a sintaxe para definir comportamentos que podem ser passados como argumentos para métodos ou armazenados em variáveis.

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

Os métodos chamados pelo main precisam ser estáticos porque são chamados no contexto da classe, não de uma instância, podendo ser chamados sem ter que criar um objeto para instanciar a classe.