



Estácio

Missão Prática | Nível 3 | Mundo 3

RPG0016 - BackEnd sem banco não tem.

Aluna: Simone Ramos de Jesus.

Matricula: 202208290965.

Campus: Polo Prado – Belo Horizonte – MG.

Curso: Desenvolvimento Full Stack

Turma: 2023.3

Objetivos da prática:

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

Pessoa.java:

```
src > cadastrobd.model > Pessoa.java > Pessoa
1  package src.cadastrobd.model;
2
3  public class Pessoa {
4      private Integer id;
5      private String nome;
6      private String logradouro;
7      private String cidade;
8      private String estado;
9      private String telefone;
10     private String email;
11     private char tPessoa;
12
13     public Pessoa(Integer id, String nome, String logradouro, String cidade, String estado, String telefone,
14                  String email, char tPessoa) {
15         this.id = id;
16         this.nome = nome;
17         this.logradouro = logradouro;
18         this.cidade = cidade;
19         this.estado = estado;
20         this.telefone = telefone;
21         this.email = email;
22         this.tPessoa = tPessoa;
23     }
24
25     public void exibir() {
26         System.out.println("ID: " + id);
27         System.out.println("Nome: " + nome);
28         System.out.println("Logradouro: " + logradouro);
29         System.out.println("Cidade: " + cidade);
30         System.out.println("Estado: " + estado);
31         System.out.println("Telefone: " + telefone);
32         System.out.println("E-mail: " + email);
33         System.out.println("Tipo: " + tPessoa);
34         System.out.println("Tipo: " + tPessoa);
35     }
36 }
```

```

    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getLogradouro() {
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getTelefone() {
        return Telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

```

```

69     }
70     public void setTelefone(String telefone) {
71         this.telefone = telefone;
72     }
73     public String getEmail() {
74         return email;
75     }
76     public void setEmail(String email) {
77         this.email = email;
78     }
79     public char getTPessoa() {
80         return tPessoa;
81     }
82     public void setTPessoa(char tPessoa) {
83         this.tPessoa = tPessoa;
84     }
85 }
86

```

- PessoaFisica.java

```
src > cadastrobd.model > PessoaFisica.java > ...
1  package src.cadastrobd.model;
2
3  public class PessoaFisica extends Pessoa {
4      private String cpf;
5
6      public PessoaFisica(Integer id, String nome, String logradouro, String cidade, String estado, String tel
7          String email, String cpf) {
8          super(id, nome, logradouro, cidade, estado, telefone, email, "F");
9          this.cpf = cpf;
10     }
11
12     public String getCpf() {
13         return cpf;
14     }
15
16     public void setCpf(String cpf) {
17         this.cpf = cpf;
18     }
19
20     @Override
21     public void toString() {
22         return super.toString() + "CPF: " + cpf;
23     }
24 }
25
```

- PessoaJuridica.java

```
src > cadastrobd.model > PessoaJuridica.java > ...
1  package src.cadastrobd.model;
2
3  public class PessoaJuridica extends Pessoa {
4      public static void main(String[] args) {
5          private String cnpj;
6
7
8      public PessoaJuridica(Integer id, String nome, String logradouro, String cidade, String estado, Stri
9          String email, String cpf, String cnpj) {
10         super(id, nome, logradouro, cidade, estado, telefone, email);
11         this.cnpj = cnpj;
12     }
13
14     public String getCnpj() {
15         return cnpj;
16     }
17
18     public void setCnpj(String cnpj) {
19         this.cnpj = cnpj;
20     }
21
22     @Override
23     public String to
24
25     String(){
26         return super.toString() + "CNPJ: " + cnpj;
27     }
28 }
29
```

PessoaFisicaDAO.java

```
src > cadastrdao.model > PessoaFisicaDAO.java > PessoaFisicaDAO > inserirPessoaFisica(PessoaFisica)
1 package src.cadastrdao.model;
2
3 import cadastrdao.model.PessoaFisica;
4 import java.sql.*;
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class PessoaFisicaDAO {
9     private Connection conn;
10
11     public PessoaFisicaDAO() {
12         private Connection conn;
13     }
14
15     public PessoaFisicaDAO(Connection connectionX) {
16         this.conn = connectionX;
17     }
18
19     public void inserirPessoaFisica(PessoaFisica pf) throws SQLException {
20         String sqlPessoa = "INSERT INTO pessoa ( nome, logradouro, cidade, estado, telefone, email, tipoPessoa) VALUES(?,?,?,?,?,?)";
21         String sqlPessoaFisica = "INSERT INTO PessoaFisica (idPessoaFisica, cpf) VALUES(?,?)";
22     }
23
24     try {
25         (conn.setAutoCommit(false));
26         int pessoaId = 0;
27         try (PreparedStatement staPessoa = conn.prepareStatement(sqlPessoa, Statement.RETURN_GENERATED_KEYS)) {
28             staPessoa.setString(1, PessoaFisica.getNome());
29             staPessoa.setString(2, PessoaFisica.getLogradouro());
30             staPessoa.setString(3, PessoaFisica.getCidade());
31             staPessoa.setString(4, PessoaFisica.getEstado());
32             staPessoa.setString(5, PessoaFisica.getTelefone());
33             staPessoa.setString(6, PessoaFisica.getEmail());
34             staPessoa.executeUpdate();
35
36             try (ResultSet rSet = staPessoa.getGeneratedKeys()) {
37                 if (rSet.next()) {
38                     pessoaId = rSet.getInt(1);
39                 }
40             }
41         }
42     } catch (SQLException e) {
43         e.printStackTrace();
44     } finally {
45         conn.setAutoCommit(true);
46     }
47 }
```

```
src > cadastrdao.model > PessoaFisicaDAO.java > PessoaFisicaDAO > inserirPessoaFisica(PessoaFisica)
39
40     try (PreparedStatement staPessoaFisica = conn.prepareStatement(sqlPessoaFisica)) {
41         staPessoaFisica.setInt(1, pessoaId);
42         staPessoaFisica.setString(2, PessoaFisica.getCpf());
43         staPessoaFisica.executeUpdate();
44     }
45
46     conn.commit();
47 } catch (
48 SQLException exception)
49 {
50     conn.rollback();
51     throw exception;
52 } finally {
53     conn.setAutoCommit(true);
54 }
55
56 public void alterar(PessoaFisica pf) throws SQLException {
57     String sqlPessoa = "UPDATE Pessoa Set nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE idPessoa = ? ";
58     String sqlPessoaFisica = "UPDATE PessoaFisica SET cpf = ? WHERE idPessoaFisica = ? ";
59
60     try {
61         conn.setAutoCommit(false);
62         try (PreparedStatement staPessoa = conn.prepareStatement(sqlPessoa)) {
63             staPessoa.setString(1, PessoaFisica.getNome());
64             staPessoa.setString(2, PessoaFisica.getLogradouro());
65             staPessoa.setString(3, PessoaFisica.getCidade());
66             staPessoa.setString(4, PessoaFisica.getEstado());
67             staPessoa.setString(5, PessoaFisica.getTelefone());
68             staPessoa.setString(6, PessoaFisica.getEmail());
69             staPessoa.setInt(7, PessoaFisica.getId());
70             staPessoa.executeUpdate();
71         }
72
73         try (PreparedStatement staPessoaFisica = conn.prepareStatement(sqlPessoaFisica)) {
74             staPessoaFisica.setString(1, PessoaFisica.getCpf());
75             staPessoaFisica.setInt(2, PessoaFisica.getId());
76         }
77     } catch (SQLException e) {
78         e.printStackTrace();
79     } finally {
80         conn.setAutoCommit(true);
81     }
82 }
```

src > cadastrodao.model > PessoaFisicaDAO.java > PessoaFisicaDAO > inserirPessoaFisica(PessoaFisica)

```
76         staPessoaFisica.executeUpdate();
77     }
78
79     conn.commit();
80 } catch(SQLException exception) {
81     conn.rollback();
82     throw exception;
83 } finally { conn.setAutoCommit(true);}
84 }
85
86 public void excluir(Integer id) throws SQLException{
87     String sqlPessoaFisica = "Delete FROM PessoaFisica WHERE idPessoaFisica = ?";
88     String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
89
90     try{
91         conn.setAutoCommit(false);
92         try(PreparedStatement staPessoaFisica = conn.prepareStatement(sqlPessoaFisica)){
93             staPessoaFisica.setInt(1,id);
94             staPessoaFisica.executeUpdate();
95         }
96
97         try(PreparedStatement staPessoa = conn.prepareStatement(sqlPessoa)){
98             staPessoa.setInt(1, id);
99             staPessoa.executeUpdate();
100        }
101
102        conn.commit();
103    } catch(SQLException exception) {
104        conn.rollback();
105        throw exception;
106    } finally { conn.setAutoCommit(true);}
107 }
108
109 public PessoaFisica getPessoa(Integer id) throws SQLException {
110     String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, Pessoa.estado,
111     try(PreparedStatement sta = conn.prepareStatement(sql)){
112         sta.setInt(1, id);
```

src > cadastrodao.model > PessoaFisicaDAO.java > PessoaFisicaDAO > inserirPessoaFisica(PessoaFisica)

```
112         sta.setInt(1, id);
113     try (ResultSet rSet = sta.executeQuery()){
114         if (rSet.next()){
115             return new PessoaFisica(
116                 rSet.getInt("idPessoa");
117                 rSet.getString("nome");
118                 rSet.getString("logradouro");
119                 rSet.getString("cidade");
120                 rSet.getString("estado");
121                 rSet.getString("telefone");
122                 rSet.getString("email");
123                 rSet.getString("cpf");
124             );
125         }
126     }
127 }
128 return null;
129 }
130
131 public List<PessoaFisica> getPessoas() throws SQLException{
132     List<PessoaFisica> list = new ArrayList<>();
133     String sql = "SELECT p.*, pf.cpf FROM Pessoa AS p INNER JOIN PessoaFisica AS pf ON p.idPessoa = pf.idPes
134     try(PreparedStatement sta = conn.prepareStatement(sql));
135     ResultSet rSet = (st.executeQuery()) {
136         while (rSet.next()) {
137             list.add(new PessoaFisica(
138                 rSet.getInt("idPessoa");
139                 rSet.getString("nome");
140                 rSet.getString("logradouro");
141                 rSet.getString("cidade");
142                 rSet.getString("estado");
143                 rSet.getString("telefone");
144                 rSet.getString("email");
145                 rSet.getString("cpf");
146             ));
147         }
148     }
```

```

128         return null;
129     }
130
131     public List<PessoaFisica> getPessoas() throws SQLException{
132         List<PessoaFisica> list = new ArrayList<>();
133         String sql = "SELECT p.*, pf.cpf FROM Pessoa AS p INNER JOIN PessoaFisica AS pf ON p.idPessoa = pf.idPes";
134         try (PreparedStatement sta = conn.prepareStatement(sql));
135             ResultSet rSet = (st.executeQuery()) {
136             while (rSet.next()) {
137                 list.add(new PessoaFisica(
138                     rSet.getInt("idPessoa");
139                     rSet.getString("nome");
140                     rSet.getString("logradouro");
141                     rSet.getString("cidade");
142                     rSet.getString("estado");
143                     rSet.getString("telefone");
144                     rSet.getString("email");
145                     rSet.getString("cpf");
146                 ));
147             }
148         }
149         return list;
150     }
151
152

```

PessoaJuridicaDAO.java:

```

src > cadastradoao.model > PessoaJuridicaDAO.java > PessoaJuridicaDAO > conn
1  package src.cadastradoao.model;
2
3  import cadastradoao.model.PessoaJuridicaDAO;
4  import java.sql.*;
5  import java.util.ArrayList;
6  import java.util.List;
7
8  public class PessoaJuridicaDAO {
9      private Connection conn;
10     public PessoaJuridicaDAO(Connection conn) {
11         this.conn = conn;
12     }
13     private PessoaJuridica extrairPessoaJuridica(ResultSet rSet) throws SQLException{
14         return new PessoaJuridica(
15             rSet.getInt("idPessoa");
16             rSet.getString("nome");
17             rSet.getString("logradouro");
18             rSet.getString("cidade");
19             rSet.getString("estado");
20             rSet.getString("telefone");
21             rSet.getString("email");
22             rSet.getString("cnpj");
23         );
24     }
25     public PessoaJuridica getPessoa(int id) throws SQLException ();
26     final String sql = "SELECT P.*, PJ.cnpj FROM Pessoa P INNER JOIN PessoaJuridica PJ ON P.idPessoa = P";
27     try (PreparedStatement st = conn.prepareStatement(sql)){
28         st.setInt(1, id);
29         try (ResultSet rSet = st.executeQuery()){
30             if (rSet.next()){
31                 return extrairPessoaJuridica(rSet);
32             }
33         }
34     }
35     return null;

```



```

src > cadastrodao.model > PessoaJuridicaDAO.java > PessoaJuridicaDAO > conn
38     final String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoaJuridica, cnpj) VALUES(?,?)";
39     try {
40         conn.setAutoCommit(false);
41         int pessoaid = 0;
42         try (PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa, Statement.RETURN_GENERATED_KEYS)) {
43             stPessoa.setString(1, pessoa.getNome());
44             stPessoa.setString(2, pessoa.getLogradouro());
45             stPessoa.setString(3, pessoa.getCidade());
46             stPessoa.setString(4, pessoa.getEstado());
47             stPessoa.setString(5, pessoa.getTelefone());
48             stPessoa.setString(6, pessoa.getEmail());
49             stPessoa.executeUpdate();
50             try (ResultSet generatedKeySet = stPessoa.getGeneratedKeys()) {
51                 if (generatedKeySet.next()) {
52                     pessoaid = generatedKeySet.getInt(1);
53                 }
54             }
55             if (pessoaid == 0) {
56                 throw new SQLException("Ops! Falha ao inserir pessoa, ID não foi gerado ou não foi encontrado");
57             }
58             try (PreparedStatement stPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica)) {
59                 stPessoaJuridica.setInt(1, pessoaid);
60                 stPessoaJuridica.setString(2, pessoa.getCnpj());
61                 stPessoaJuridica.executeUpdate();
62             }
63             conn.commit();
64         }
65         catch (SQLException exception) {conn.setAutoCommit(true);}
66         finally {conn.setAutoCommit(true);}
67     }
68 }
69 public void alterar(PessoaJuridica pessoa) throws SQLException{
70     final String sqlPessoa = "UPDATE Pessoa Set nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?";
71     final String sqlPessoaJuridica = "UPDATE PessoaJuridica SET cnpj =? WHERE idPessoaJuridica = ?";
72     try{
73         conn.setAutoCommit(false);
74         try (PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa)){
75             stPessoa.setString(1, pessoa.getNome());
76             stPessoa.setString(2, pessoa.getLogradouro());
77             stPessoa.setString(3, pessoa.getCidade());
78             stPessoa.setString(4, pessoa.getEstado());
79             stPessoa.setString(5, pessoa.getTelefone());
80             stPessoa.setString(6, pessoa.getEmail());
81             stPessoa.setInt(7, PessoaFisicapate());
82             stPessoa.executeUpdate();
83         }
84         try (PreparedStatement stPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica)){
85             stPessoaJuridica.setString(1, PessoaFisica.getCpf());
86             stPessoaJuridica.setInt(2, PessoaFisica.getId());
87             stPessoaJuridica.executeUpdate();
88         }
89         conn.commit();
90     } catch (SQLException exception) {
91         conn.rollback();
92         throw exception;
93     } finally { conn.setAutoCommit(true);}
94 }
95 public void excluir(Integer id) throws SQLException{
96     String sqlPessoaJuridica = "Delete FROM PessoaJuridica WHERE idPessoaJuridica = ?";
97     String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
98     try{
99         conn.setAutoCommit(false);
100         try (PreparedStatement stPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica)){
101             stPessoaJuridica.setInt(1, id);
102             stPessoaJuridica.executeUpdate();
103         }
104         try (PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa)){
105             stPessoa.setInt(1, id);
106             stPessoa.executeUpdate();
107         }
108         conn.commit();
109     } catch (SQLException exception) {
110         conn.rollback();

```

```

95 public void excluir(Integer id) throws SQLException{
96     String sqlPessoaJuridica = "Delete FROM PessoaJuridica WHERE idPessoaJuridica = ?";
97     String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
98     try{
99         conn.setAutoCommit(false);
100         try(PreparedStatement stPessoaJuridica = conn.prepareStatement(sqlPessoaJuridica)){
101             stPessoaJuridica.setInt(1, id);
102             stPessoaJuridica.executeUpdate();
103         }
104         try(PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa)){
105             stPessoa.setInt(1, id);
106             stPessoa.executeUpdate();
107         }
108         conn.commit();
109     } catch(SQLException exception) {
110         conn.rollback();
111         throw exception;
112     } finally { conn.setAutoCommit(true);}
113 }
114
115

```

ConectoBD.java:

```

src > cadastro.model.util > ConectorBD.java > ConectorBD > loadProperties()
1 package src.cadastro.model.util;
2
3 import java.io.*;
4 import java.util.Properties;
5
6 import javax.management.RuntimeErrorException;
7
8 import java.util.ArrayList;
9 import java.sql.SQLException;
10 import java.sql.DriverManager;
11 import java.sql.Statement;
12 import java.sql.ResultSet;
13 import java.sql.Connection;
14
15 public class ConectorBD {
16     private static Connection conn = null;
17
18     public static Connection getConnection() {
19         if (conn == null) {
20             try {
21                 Properties propt = loadProperties();
22                 String url = propt.getProperty("url");
23                 String login = propt.getProperty("login");
24                 String senha = propt.getProperty("senha");
25                 conn = DriverManager.getConnection(url, login, senha);
26             } catch (SQLException exception) {
27                 throw new RuntimeException(
28                     "Ops! erro ao obter conexão com o banco de dados: " + exception.getMessage(), except
29                 );
30             }
31         }
32         return conn;
33     }
34 }

```

```

src > cadastro.model.util > ConectorBD.java > ConectorBD
17
18 public static Connection gConnection() {
19     if (conn == null) {
20         try {
21             Properties propt = loadProperties();
22             String url = propt.getProperty("url");
23             String login = propt.getProperty("login");
24             String senha = propt.getProperty("url");
25             conn = DriverManager.getConnection(url, login, senha);
26
27         } catch (SQLException exception) {
28             throw new RuntimeException(
29                 "Ops! erro ao obter conexão com o banco de dados: " + exception.getMessage(), exception)
30         }
31     }
32     return conn;
33 }
34
35 public static void closeStatement(Statement sta){
36     try{
37         if(sta != null){ sta.close();}
38
39         catch(SQLException exception){
40             throw new RuntimeException("Ops! Erro ao fechar o Statement: " + exception.getMessage(),exception)
41         }
42     }
43 }
44
45 public static void closeResultSet(ResultSet rset){
46     try{
47         if(rset != null){rset.close();}
48
49         catch(SQLException exception){
50             throw new RuntimeException("Ops! Erro ao fechar o ResultSet: " + exception.getMessage(),exception)
51         }
52     }
53 }

```

```

src > cadastro.model.util > ConectorBD.java > ConectorBD
17
18 > public static Connection gConnection() {...
34
35 > public static void closeStatement(Statement sta){...
44
45 > public static void closeResultSet(ResultSet rset){...
54
55 public static void closeConnection(){
56     try{
57         if(conn != null){conn.close();}
58
59         catch(SQLException exception){
60             throw new RuntimeException("Ops! Erro ao fechar a conexão com o Banco de Dados: " + exception.get
61         }
62     }
63 }
64
65 public static Properties loadProperties(){
66     try(InputStream is = ConectorBD.class.getClassLoader().getResourceAsStream("properties.db")){
67         Properties propt = new Properties();
68         if(is != null){propt.load(is);}
69         else{ throw new FileNotFoundException("Ops! Arquivo properties.db não encontrado no classpath.");}
70
71         return propt;
72
73         catch(SQLException exception){
74             throw new RuntimeException("Ops! Erro ao carregar as propriedades a conexão do Banco de Dados: "
75         }
76     }
77 }
78
79

```

SequenceManager.java:

src > cadastro.model.util > SequenceManager.java > ...

```
1 package src.cadastro.model.util;
2
3 public class BDEException
4 extends RuntimeException{
5
6     public ExceptionDB(String Mens){
7         super(mens);
8     }
9
10 }
11
```

CadastroBDTeste.java:

src > cadastrobdteste.model > cadastroBDTeste.java > cadastroBDTeste > exibirPessoasFisicas(PessoaFisicaDAO)

```
1 package src.cadastrobdteste.model;
2
3 import cadastrobd.modelbd.model.Pessoa;
4 import cadastrobd.modelbd.model.PessoaFisica;
5 import cadastrobd.modelbd.model.PessoaJuridica;
6 import cadastrodao.model.PessoaFisicaDAO;
7 import cadastrodao.model.PessoaJuridicaDAO;
8 import cadastro.model.util.ConectorBD;
9 import cadastro.model.util.SequenceManager;
10
11 import java.sql.Connection;
12 import java.sql.SQLException;
13 import java.util.List;
14 import java.util.Scanner;
15
16 public class cadastroBDTeste {
17     Run | Debug
18     public static void main(String[] args){
19         Scanner scanner = new Scanner(System.in);
20         Connection connection = ConectorBD.getConnection();
21         PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(connection);
22         PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(connection);
23
24         int opcao;
25         do{
26             System.out.println("=====");
27             System.out.println("Selecione a opção desejada: ");
28             System.out.println("1 - Incluir Pessoa");
29             System.out.println("2 - Alterar Pessoa");
30             System.out.println("3 - Excluir Pessoa");
31             System.out.println("4 - Buscar pelo Id");
32             System.out.println("5 - Exibir Todos");
33             System.out.println("6 - Exibir Pessoa Física");
34             System.out.println("7 - Exibir Pessoa Jurídica");
35             System.out.println("0 - Finalizar Programa");
36             System.out.println("=====");
37             opcao = scanner.nextInt();
38         } while (opcao != 0);
39     }
40 }
```

```

36 opcao = scanner.nextInt();
37 scanner.nextLine();
38 try{
39     switch(opcao){
40         case 1:
41             system.out.println("F - Pessoa Física | J - Pessoa Jurídica");
42             char tInclusao = scanner.next().charAt(opcao);
43             scanner.nextLine();
44             if (tInclusao == "F" | tInclusao == "fi"){cadastrarPessoaFisica(pessoaFisicaDAO, scanner);}
45             else if (tInclusao == "J" | tInclusao == "ju"){cadastrarPessoaJuridica(pessoaFisicaJuridicaDAO, scanner);}
46             break;
47         case 2:
48             alterarPessoa(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
49             break;
50         case 3:
51             excluirPessoa(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
52             break;
53         case 4:
54             buscarPeloId(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
55             break;
56         case 5:
57             exibirTodos(pessoaFisicaDAO, pessoaJuridicaDAO);
58             break;
59         case 6:
60             exibirPessoaFisica(pessoaFisicaDAO);
61             break;
62         case 7:
63             exibirPessoaJuridica(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
64             break;
65         case 0:
66             System.out.println("Encerrando o programa. Volte sempre que precisar!!!");
67             break;
68         default: System.out.println("Ops... Opção inválida. Tente novamente!");
69     } catch(SQLException exception){
70         scanner.nextLine();
71         try{
72             switch(opcao){
73                 case 1: ...
74                 case 2: ...
75                 case 3: ...
76                 case 4: ...
77                 case 5: ...
78                 case 6: ...
79                 case 7: ...
80                 case 0: ...
81                 default: System.out.println("Ops... Opção inválida. Tente novamente!");
82             } catch(SQLException exception){
83                 System.out.println("Ops... Erro no Banco de Dados: " + exception.getMessage());
84                 exception.printStackTrace();
85             }
86         } while (opcao != 0);
87     }
88 }

```

```
Run | Debug
17 > public static void main(String[] args){ ...
77
78 private static void cadastroPessoaFisica(PessoaFisicaDAO pessoaFisicaDAO, Scanner sca) throws SQLException {
79     System.out.println("Digite o nome completo do usuário: ");
80     String nomeCompleto = scanner.nextLine();
81     System.out.println("Digite o CPF: ");
82     String cpf = scanner.nextLine();
83     System.out.println("Digite o telefone: ");
84     String telefone = scanner.nextLine();
85     System.out.println("Digite o email: ");
86     String email = scanner.nextLine();
87     System.out.println("Digite o logradouro: ");
88     String logradouro = scanner.nextLine();
89     System.out.println("Digite a cidade: ");
90     String cidade = scanner.nextLine();
91     System.out.println("Digite o estado: ");
92     String estado = scanner.nextLine();
93
94     PessoaFisica newPessoaFisica = new PessoaFisica(0, nome, cpf, telefone, email, logradouro, cidade, estado);
95     pessoaFisicaDAO.inserirPessoaFisica(newPessoaFisica);
96     System.out.println("Pessoa Fisica cadastrada com sucesso");
97 }
98
99 private static void cadastroPessoaJuridica(PessoaFisicaJuridicaDAO pessoaFisicaJuridicaDAO, Scanner sca)
100     throws SQLException {
101     System.out.println("Digite o nome completo da Empresa: ");
102     String nomeCompleto = scanner.nextLine();
103     System.out.println("Digite o CNPJ: ");
104     String cpf = scanner.nextLine();
105     System.out.println("Digite o telefone: ");
106     String telefone = scanner.nextLine();
107
108     PessoaFisica newPessoaFisicaJuridica = new PessoaFisica(0, nome, cnpj, telefone, email, logradouro, cidade, estado);
109     pessoaFisicaDAO.inserirPessoaFisicaJuridica(newPessoaFisicaJuridica);
110     System.out.println("Empresa cadastrada com sucesso");
111 }
112
113 private static void alterarPessoa(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO, Scanner sca) {
114     System.out.println("F - Alterar Pessoa Fisica | J - Alterar Pessoa Jurídica");
115     char tPessoa = sca.next().charAt(0);
116     sca.nextLine();
117
118     if (tPessoa == "F" | tPessoa == "fi"){
119         System.out.println(" Digite o ID do usuário que deseja alterar: ");
120         int id = sca.nextInt();
121         sca.nextLine();
122
123         PessoaFisica pesExistente = pessoaFisicaDAO.getPessoa(id);
124     }
125 }
```

src > cadastrobdteste.model > cadastroBDTeste.java > cadastroBDTeste > main(String[])

```
133
134 ~ if (pesExistente != null){
135     System.out.println("Digite o novo, nome completo do usuário: ");
136     String novoNomeCompleto = scanner.nextLine();
137     System.out.println("Digite o novo CPF; ");
138     String novoCpf = scanner.nextLine();
139     System.out.println("Digite o novo telefone; ");
140     String novoTelefone = scanner.nextLine();
141     System.out.println("Digite o novo email; ");
142     String novoEmail = scanner.nextLine();
143     System.out.println("Digite o novo logradouro: ");
144     String novoLogradouro = scanner.nextLine();
145     System.out.println("Digite a nova cidade: ");
146     String novaCidade = scanner.nextLine();
147     System.out.println("Digite o novo estado: ");
148     String novoEstado = scanner.nextLine();
149
150     PessoaFisica novaPessoaFisica = new PessoaFisica(id, novoNome, novoCpf, novoTelefone, novoEmail, novo
151     pessoaFisicaDAO.alterarPessoaFisica(novaPessoa);
152     System.out.println("Pessoa Física atualizada com sucesso");
153 } else {System.out.println("Pessoa Física não encontrada.");}
154 }
155
156 ~ else if (tPessoa == "J" | tPessoa == "ju"){
157     System.out.println(" Digite o ID da empresa que deseja alterar: ");
158     int id = sca.nextInt();
159     sca.nextLine();
160
161     PessoaJuridica pesExistente = pessoaJuridicaDAO.getPessoa(id);
162
163 ~ if (pesExistente != null){
164     System.out.println("Digite o novo, nome completo da empresa: ");
165     String novoNomeCompleto = scanner.nextLine();
166     System.out.println("Digite o novo CPF; ");
167     String novoCnpj = scanner.nextLine();
168     System.out.println("Digite o novo telefone; ");
```

src > cadastrobdteste.model > cadastroBDTeste.java > cadastroBDTeste > main(String[])

```
168     System.out.println("Digite o novo telefone; ");
169     String novoTelefone = scanner.nextLine();
170     System.out.println("Digite o novo email; ");
171     String novoEmail = scanner.nextLine();
172     System.out.println("Digite o novo logradouro: ");
173     String novoLogradouro = scanner.nextLine();
174     System.out.println("Digite a nova cidade: ");
175     String novaCidade = scanner.nextLine();
176     System.out.println("Digite o novo estado: ");
177     String novoEstado = scanner.nextLine();
178
179     PessoaJuridica novaPessoa = new PessoaJuridica(id, novoNome, novoCnpj, novoTelefone, novoEmail,
180     pessoaJuridicaDAO.alterarPessoaJuridica(novaPessoa);
181     System.out.println("Pessoa Jurídica atualizada com sucesso.");
182 } else {System.out.println("Pessoa Jurídica não encontrada.");}
183 else{System.out.println("Ops.. Opção inválida")}
184 }
185 }
186
187 private static void excluirPessoa(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO, Scanner
188 System.out.println("F - Excluir Pessoa Física | J - Excluir Pessoa Jurídica");
189 char tPessoa = sca.next().charAt(0);
190 sca.nextLine();
191
192 if (tPessoa == "F" | tPessoa == "fi"){
193     System.out.println(" Digite o ID do usuário que deseja excluir: ");
194     int id = sca.nextInt();
195     pessoaFisicaDAO.excluir(id);
196     System.out.println(" Pessoa Física excluída com sucesso.");
197
198     else if (tPessoa == "J" | tPessoa == "ju"){
199     System.out.println(" Digite o ID da empresa que deseja excluir:");
200     int id = sca.nextInt();
201     pessoaJuridicaDAO.excluir(id);
202     System.out.println("Pessoa Jurídica excluída com sucesso.");
203     sca.nextLine();
```

```
src > cadastrobdteste.model > cadastroBDTeste.java > cadastroBDTeste > main(String[])
205     } else { System.out.println("Ops... Opção inválida"); }
206     }
207 }
208
209 private static void buscarPessoaPeloId(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
210     System.out.println("F - buscar Pessoa Física | J - Buscar Pessoa Jurídica");
211     char tPessoa = sca.next().charAt(0);
212     sca.nextLine();
213
214     if (tPessoa == "F" | tPessoa == "fi"){ ...
215
216     else if (tPessoa == "J" | tPessoa == "ju"){
217         System.out.println(" Digite o ID da empresa: ");
218         int id = sca.nextInt();
219         sca.nextLine();
220
221         PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);
222
223         if (pessoaFisica != null){
224             System.out.println("Detalhes do Empresa:");
225             System.out.println("ID: " + pessoaJuridica.getId());
226             System.out.println("Nome: " + pessoaJuridica.getNome());
227             System.out.println("CNPJ: " + pessoaJuridica.getCpf());
228             System.out.println("Telefone: " + pessoaJuridica.getTelefone());
229             System.out.println("Email: " + pessoaJuridica.getEmail());
230             System.out.println("Logradouro: " + pessoaJuridica.getLogradouro());
231             System.out.println("Cidade: " + pessoaJuridica.getCidade());
232             System.out.println("Estado: " + pessoaJuridica.getEstado());
233         }
234         else { System.out.println("Empresa não encontrada."); }
235         else { System.out.println("Opção inválida."); }
236     }
237 }
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256 }
```

```
src > cadastrobdteste.model > cadastroBDTeste.java > cadastroBDTeste > main(String[])
260     System.out.println(" Usuário cadastrado:");
261     List<PessoaFisica> pesFisicas = pessoaFisicaDAO.getPessoas();
262     if (pesFisicas.isEmpty()) {
263         System.out.println("Nenhum usuário cadastrado.");
264     } else {
265         for (PessoaFisica pf : pesFisicas) {
266             System.out.println("ID: " + pf.getId());
267             System.out.println("Nome: " + pf.getNome());
268             System.out.println("CPF: " + pf.getCpf());
269             System.out.println("Telefone: " + pf.getTelefone());
270             System.out.println("Email: " + pf.getEmail());
271             System.out.println("Logradouro: " + pf.getLogradouro());
272             System.out.println("Cidade: " + pf.getCidade());
273             System.out.println("Estado: " + pf.getEstado());
274             System.out.println();
275         }
276     }
277
278     System.out.println(" Empresa cadastrada:");
279     List<PessoaJuridica> pesJuridicas = pessoaJuridicaDAO.getPessoasJuridicas();
280     if (pesJuridicas.isEmpty()) {
281         System.out.println("Nenhuma empresa cadastrada.");
282     } else {
283         for (PessoaJuridica pj : pesJuridicas) {
284             System.out.println("ID: " + pj.getId());
285             System.out.println("Nome: " + pj.getNome());
286             System.out.println("CNPJ: " + pj.getCnpj());
287             System.out.println("Telefone: " + pj.getTelefone());
288             System.out.println("Email: " + pj.getEmail());
289             System.out.println("Logradouro: " + pj.getLogradouro());
290             System.out.println("Cidade: " + pj.getCidade());
291             System.out.println("Estado: " + pj.getEstado());
292             System.out.println();
293         }
294     }
295 }
296 }
```


src > cadastrobdteste.model > cadastroBDTeste.java > cadastroBDTeste > main(String[])

```
297 private static void exibirPessoasFisicas(PessoaFisicaDAO pessoaFisicaDAO) throws SQLException {
298     System.out.println(" Usuários cadastrados:");
299     List<PessoaFisica> pesFisicas = pessoaFisicaDAO.getPessoas();
300     if (pesFisicas.isEmpty()) {
301         System.out.println("Nenhum usuário cadastrado.");
302     } else {
303         for (PessoaFisica pf : pesFisicas) {
304             System.out.println("ID: " + pf.getId());
305             System.out.println("Nome: " + pf.getNome());
306             System.out.println("CPF: " + pf.getCpf());
307             System.out.println("Telefone: " + pf.getTelefone());
308             System.out.println("Email: " + pf.getEmail());
309             System.out.println("Logradouro: " + pf.getLogradouro());
310             System.out.println("Cidade: " + pf.getCidade());
311             System.out.println("Estado: " + pf.getEstado());
312             System.out.println();
313         }
314     }
315
316     System.out.println(" Empresas cadastradas:");
317     List<PessoaJuridica> pesJuridicas = pessoaJuridicaDAO.getPessoas();
318     if (pesJuridicas.isEmpty()) {
319         System.out.println("Nenhuma empresa cadastrada.");
320     } else {
321         for (PessoaJuridica jd : pesJuridicas) {
322             System.out.println("ID: " + pj.getId());
323             System.out.println("Nome: " + pj.getNome());
324             System.out.println("CPF: " + pj.getCpf());
325             System.out.println("Telefone: " + pj.getTelefone());
326             System.out.println("Email: " + pj.getEmail());
327             System.out.println("Logradouro: " + pj.getLogradouro());
328             System.out.println("Cidade: " + pj.getCidade());
329             System.out.println("Estado: " + pj.getEstado());
330             System.out.println();
331         }
332     }
333 }
```

Resultado da execução dos códigos:

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
0
Encerrando o programa.
```

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
4
F - Buscar Pessoa Física | J - Buscar Pessoa Jurídica
J
Digite o ID da Pessoa Jurídica:
21
Detalhes da Pessoa Jurídica:
ID: 21
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1111-1111
Email: email@example.com
CNPJ: 111111111111
=====
```

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
3
F - Excluir Pessoa Física | J - Excluir Pessoa Jurídica
F
Digite o ID da Pessoa Física a ser excluída:
35
Pessoa Física excluída com sucesso.
=====
```

```
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Exibir Somente Pessoas Físicas
7 - Exibir Somente Pessoas Jurídicas
0 - Finalizar Programa
=====
5
PessoasJuridicasCadastradas:
ID: 33
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 2222-2222
Email: email@example.com
CNPJ: 11111111111111

ID: 7
Nome: Joao
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: email@example.com
CPF: 11111111111
```

Importância dos componentes de middleware, como o JDBC:

- O JDBC (Java Database Connectivity) é um middleware crucial para a comunicação entre aplicativos Java e bancos de dados. Ele fornece uma interface padronizada para acesso a dados, independentemente do banco de dados subjacente, aumentando a portabilidade e facilitando a manutenção.

Diferença no uso de Statement ou PreparedStatement para a manipulação de dados:

- **Statement** é usado para executar consultas SQL simples.
- **PreparedStatement** é preferido para consultas parametrizadas. A principal diferença está na prevenção de injeção de SQL e no ganho de desempenho e pode ser pré-compilado.

Como o padrão DAO melhora a manutenibilidade do software:

- O padrão DAO (Data Access Object) separa a lógica de acesso a dados do restante do código, facilitando a manutenção. Isso permite alterações na camada de persistência sem afetar outras partes do sistema, promovendo uma melhor organização e modularidade.

Reflexo da herança no banco de dados em um modelo estritamente relacional:

- Em um modelo estritamente relacional, a herança é frequentemente representada usando tabelas separadas para cada classe concreta. Isso resulta em uma estrutura de várias tabelas relacionadas, onde cada tabela representa uma classe específica ou uma subclasse.

Diferenças entre persistência em arquivo e persistência em banco de dados:

- A persistência em arquivo armazena dados em arquivos no sistema de arquivos, sendo mais simples, mas menos eficiente para consultas complexas. A persistência em banco de dados utiliza sistemas gerenciadores de banco de dados, proporcionando maior segurança, concorrência e capacidade de consultas avançadas.

Simplificação da impressão de valores usando operador lambda no Java mais recente:

- O uso de operadores lambda simplifica a expressão de operações em coleções, como listas. Com a introdução de métodos como `forEach` em coleções, a impressão de valores contidos nas entidades pode ser realizada de forma mais concisa e legível.

Razão pela qual métodos acionados diretamente pelo método `main` precisam ser marcados como `static`:

- Métodos acionados diretamente pelo método `main` precisam ser `static` porque o método `main` é estático. Métodos estáticos pertencem à classe, não a instâncias específicas, e podem ser chamados sem criar uma instância da classe. O método `main` é o ponto de entrada do programa e precisa ser estático para ser chamado sem criar um objeto da classe principal.

Conclusão:

Esses conceitos fundamentais, como acesso a dados, design de padrões, herança em bancos de dados, persistência e novos recursos da linguagem, são cruciais para o desenvolvimento eficiente e sustentável de software em Java. Compreender esses elementos contribui para a construção de sistemas robustos, modulares e de fácil manutenção.