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

Leonardo Schaffer Mota – 2022.05.09098-1

Polo centro – Santo André – SP
BackEnd sem banco não tem– 2023.2 – 2023.4

Objetivo da Prática

Implementar persistência com base no middleware JDBC.

Utilizar o padrão DAO (Data Access Object) no manuseio de dados.

Implementar o mapeamento objeto-relacional em sistemas Java.

Criar sistemas cadastrais com persistência em banco relacional.

1º Procedimento – Mapeamento Objeto-Relacional e DAO

Classe Pessoa:

```java
package cadastrobd.model;

/**
 *
 * @author leosc
 */
public class Pessoa {
    private int id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

    public Pessoa(){
    }

    public Pessoa(int id, String nome, String logradouro, String cidade, String
        this.id = id;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }

        public int getId() {
        return id;
    }
```

```java
    public void setId(int 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;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public void exibir() {
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
        System.out.println("Logradouro: " + logradouro);
        System.out.println("Cidade: " + cidade);
        System.out.println("Estado: " + estado);
        System.out.println("Telefone: " + telefone);
        System.out.println("Email: " + email);
    }
}
```

Classe Pessoa fisica:

```java
package cadastrobd.model;

/**
 *
 * @author leosc
 */
public class PessoaFisica extends Pessoa {
    private String cpf;

    public PessoaFisica(){
    }

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

    public String getCpf() {
        return cpf;
    }

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

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

Classe Pessoa juridica:

```java
public class PessoaJuridica extends Pessoa {
    private String cnpj;

    public PessoaJuridica() {
    }

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

    public String getCnpj() {
        return cnpj;
    }

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

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

Classe ConectorBD:

```java
public class ConectorBD {
    private String url;
    private String usuario;
    private String senha;

    public ConectorBD(String url, String usuario, String senha) {
        this.url = url;
        this.usuario = usuario;
        this.senha = senha;
    }

    public Connection getConnection() throws SQLException {
        return DriverManager.getConnection(url, user:usuario, password: senha);
    }

    public PreparedStatement getPrepared(String sql) throws SQLException {
        Connection conn = getConnection();
        return conn.prepareStatement(string: sql);
    }

    public ResultSet getSelect(String sql) throws SQLException {
        Connection conn = getConnection();
        PreparedStatement stmt = conn.prepareStatement(string: sql);
        return stmt.executeQuery();
    }

    public void close(Connection conn) {
        if (conn != null) {
            try {
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    public void close(Statement stmt) {
        if (stmt != null) {
            try {
                stmt.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    public void close(ResultSet rs) {
        if (rs != null) {
            try {
                rs.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    public void close(Connection conn, Statement stmt, ResultSet rs) {
        close(rs);
        close(stmt);
        close(conn);
    }
}
```

Classe SequenceManager:

```java
public class SequenceManager {
    private ConectorBD conector;

    public SequenceManager(ConectorBD conector) {
        this.conector = conector;
    }

    public int getValue(String sequenceName) {
        int nextValue = -1;
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;

        try {
            conn = conector.getConnection();
            String query = "SELECT NEXT VALUE FOR " + sequenceName + " AS NextVa
            stmt = conn.prepareStatement(string: query);
            rs = stmt.executeQuery();

            if (rs.next()) {
                nextValue = rs.getInt(string: "NextValue");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            conector.close(rs);
            conector.close(stmt);
            conector.close(conn);
        }

        return nextValue;
    }
}
```

Classe PessoaFisicaDAO:

```java
public class PessoaFisicaDAO {
    private ConectorBD conector;

    public PessoaFisicaDAO(ConectorBD conector) {
        this.conector = conector;
    }

    public PessoaFisicaDAO(){
    }

    public PessoaFisica getPessoa(int id) {
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;
        PessoaFisica pessoa = null;

        try {
            conn = conector.getConnection();
            String query = "SELECT * FROM Pessoas P JOIN PessoasFisicas PF ON P.
            stmt = conn.prepareStatement(string: query);
            stmt.setInt(i: 1, i1: id);
            rs = stmt.executeQuery();
```

```
            if (rs.next()) {
                pessoa = new PessoaFisica();
                pessoa.setId(id: rs.getInt(string: "idPessoa"));
                pessoa.setNome(nome:rs.getString(string: "nome"));
                pessoa.setLogradouro(logradouro:rs.getString(string: "logradouro"));
                pessoa.setCidade(cidade: rs.getString(string: "cidade"));
                pessoa.setEstado(estado: rs.getString(string: "estado"));
                pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
                pessoa.setEmail(email:rs.getString(string: "email"));
                pessoa.setCpf(cpf: rs.getString(string: "cpf"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            conector.close(rs);
            conector.close(stmt);
            conector.close(conn);
        }

        return pessoa;
    }

    public List<PessoaFisica> getPessoas() {
        List<PessoaFisica> pessoas = new ArrayList<>();
        Connection conn = null;
        PreparedStatement stmt = null;
        ResultSet rs = null;

        try {
            conn = conector.getConnection();
            String query = "SELECT * FROM Pessoas P JOIN PessoasFisicas PF ON P.
            stmt = conn.prepareStatement(string: query);
            rs = stmt.executeQuery();

            while (rs.next()) {
                PessoaFisica pessoa = new PessoaFisica();
                pessoa.setId(id: rs.getInt(string: "idPessoa"));
                pessoa.setNome(nome:rs.getString(string: "nome"));
                pessoa.setLogradouro(logradouro:rs.getString(string: "logradouro"));
                pessoa.setCidade(cidade: rs.getString(string: "cidade"));
                pessoa.setEstado(estado: rs.getString(string: "estado"));
                pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
                pessoa.setEmail(email:rs.getString(string: "email"));
                pessoa.setCpf(cpf: rs.getString(string: "cpf"));
                pessoas.add(e: pessoa);
            }
        } catch (SQLException e) {
        } finally {
            conector.close(rs);
            conector.close(stmt);
            conector.close(conn);
        }

        return pessoas;
    }

    public boolean incluir(PessoaFisica pessoa) {
        Connection conn = null;
        PreparedStatement stmtPessoa = null;
        PreparedStatement stmtPessoaFisica = null;
        boolean sucesso = false;

```

```
104         try {
105             conn = conector.getConnection();
106             conn.setAutoCommit(bln: false);
107 
108             // Inserir na tabela Pessoas
109             String queryPessoa = "INSERT INTO Pessoas ("
110                     + "idPessoa, nome, logradouro, cidade, estado, telefone,"
111                     + " email) VALUES (?, ?, ?, ?, ?, ?, ?)";
112             stmtPessoa = conn.prepareStatement(string: queryPessoa,
113                     i: PreparedStatement.RETURN_GENERATED_KEYS);
114             stmtPessoa.setInt(i: 1, i1: pessoa.getId());
115             stmtPessoa.setString(i: 2, string: pessoa.getNome());
116             stmtPessoa.setString(i: 3, string: pessoa.getLogradouro());
117             stmtPessoa.setString(i: 4, string: pessoa.getCidade());
118             stmtPessoa.setString(i: 5, string: pessoa.getEstado());
119             stmtPessoa.setString(i: 6, string: pessoa.getTelefone());
120             stmtPessoa.setString(i: 7, string: pessoa.getEmail());
121             stmtPessoa.executeUpdate();
122 
123             if (pessoa.getId() != -1) {
124                 // Inserir na tabela PessoasFisicas
125                 String queryPessoaFisica = "INSERT INTO PessoasFisicas ("
126                         + "idPFisica, cpf) VALUES (?, ?)";
127                 stmtPessoaFisica = conn.prepareStatement(string: queryPessoaFisica
128                 stmtPessoaFisica.setInt(i: 1, i1: pessoa.getId());
129                 stmtPessoaFisica.setString(i: 2, string: pessoa.getCpf());
130                 stmtPessoaFisica.executeUpdate();
131                 conn.commit();
132                 sucesso = true;
133             } else {
134                 conn.rollback();
135             }
136         } catch (SQLException e) {
137             if (conn != null) {
138                 try {
139                     conn.rollback();
140                 } catch (SQLException ex) {
141                     ex.printStackTrace();
142                 }
143             }
144         } finally {
145             conector.close(stmt:stmtPessoaFisica);
146             conector.close(stmt:stmtPessoa);
147             conector.close(conn);
148         }
149 
150         return sucesso;
151     }
152 
153     public boolean alterar(PessoaFisica pessoa) {
154         Connection conn = null;
155         PreparedStatement stmtPessoa = null;
156         PreparedStatement stmtPessoaFisica = null;
157         boolean sucesso = false;
158 
159         try {
160             conn = conector.getConnection();
161             conn.setAutoCommit(bln: false);
162 
163             // Atualizar tabela Pessoas
164             String queryPessoa = "UPDATE Pessoas SET nome = ?, logradouro = ?,"
165                                 + "cidade = ?, estado = ?, telefone = ?,"
166                                 + " email = ? WHERE idPessoa = ?";
```

```
            stmtPessoa = conn.prepareStatement(string: queryPessoa);
            stmtPessoa.setString(i: 1, string: pessoa.getNome());
            stmtPessoa.setString(i: 2, string: pessoa.getLogradouro());
            stmtPessoa.setString(i: 3, string: pessoa.getCidade());
            stmtPessoa.setString(i: 4, string: pessoa.getEstado());
            stmtPessoa.setString(i: 5, string: pessoa.getTelefone());
            stmtPessoa.setString(i: 6, string: pessoa.getEmail());
            stmtPessoa.setInt(i: 7, i1: pessoa.getId());
            stmtPessoa.executeUpdate();

            // Atualizar tabela PessoasFisicas
            String queryPessoaFisica = "UPDATE PessoasFisicas SET cpf = ? WHERE
            stmtPessoaFisica = conn.prepareStatement(string: queryPessoaFisica);
            stmtPessoaFisica.setString(i: 1, string: pessoa.getCpf());
            stmtPessoaFisica.setInt(i: 2, i1: pessoa.getId());
            stmtPessoaFisica.executeUpdate();

            conn.commit();
            sucesso = true;
        } catch (SQLException e) {
            e.printStackTrace();
            if (conn != null) {
                try {
                    conn.rollback();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
        } finally {
            conector.close(stmt:stmtPessoaFisica);
            conector.close(stmt:stmtPessoa);
            conector.close(conn);
        }

        return sucesso;
    }

    public boolean excluir(int id) {
        Connection conn = null;
        PreparedStatement stmt = null;
        boolean sucesso = false;

        try {
            conn = conector.getConnection();
            conn.setAutoCommit(bln: false);

            // Excluir da tabela PessoasFisicas
            String queryPessoaFisica = "DELETE FROM PessoasFisicas WHERE idPFisi
            stmt = conn.prepareStatement(string: queryPessoaFisica);
            stmt.setInt(i: 1, i1: id);
            stmt.executeUpdate();

            // Excluir da tabela Pessoas
            String queryPessoa = "DELETE FROM Pessoas WHERE idPessoa = ?";
            stmt = conn.prepareStatement(string: queryPessoa);
            stmt.setInt(i: 1, i1: id);
            stmt.executeUpdate();

            conn.commit();
            sucesso = true;

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

```
                if (conn != null) {
                    try {
                        conn.rollback();
                    } catch (SQLException ex) {
                        ex.printStackTrace();
                    }
                }
            } finally {
                conector.close(stmt);
                conector.close(conn);
                if (sucesso) {
                    System.out.println(x: "Pessoa excluída com sucesso!");
                } else {
                    System.out.println(x: "Erro ao excluir pessoa.");
                }
            }

            return sucesso;
        }
    }
```

Classe PessoaJuridicaDAO:

```
public class PessoaJuridicaDAO {
    private ConectorBD conector;


    public PessoaJuridicaDAO(){
    }

    public PessoaJuridicaDAO(ConectorBD conector) {
        this.conector = conector;

    }

    public PessoaJuridica getPessoa(int id) {
        String sql = "SELECT * FROM Pessoas P INNER JOIN PessoasJuridicas PJ ON"
                        + " P.idPessoa = PJ.idPJuridica WHERE P.idPessoa = ?";
        try (Connection conn = conector.getConnection();
             PreparedStatement stmt = conn.prepareStatement(string: sql)) {
            stmt.setInt(i: 1, i1: id);
            try (ResultSet rs = stmt.executeQuery()) {
                if (rs.next()) {
                    PessoaJuridica pessoaJuridica =
                            new PessoaJuridica(id: rs.getInt(string: "idPessoa"),
                            nome:rs.getString(string: "nome"),
                            logradouro:rs.getString(string: "logradouro"),
                            cidade: rs.getString(string: "cidade"),
                            estado: rs.getString(string: "estado"),
                            telefone: rs.getString(string: "telefone"),
                            email:rs.getString(string: "email"),
                            cnpj:rs.getString(string: "cnpj"));
                    return pessoaJuridica;
                }
            }
        } catch (SQLException e) {
        }
        return null;
    }

```

```java
public List<PessoaJuridica> getPessoas() {
    List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
    String sql = "SELECT * FROM Pessoas P INNER JOIN PessoasJuridicas PJ ON
    try (Connection conn = conector.getConnection();
         PreparedStatement stmt = conn.prepareStatement(string: sql);
         ResultSet rs = stmt.executeQuery()) {
        while (rs.next()) {
            PessoaJuridica pessoaJuridica =
                    new PessoaJuridica(id: rs.getInt(string: "idPessoa"),
                    nome:rs.getString(string: "nome"),
                    logradouro:rs.getString(string: "logradouro"),
                    cidade: rs.getString(string: "cidade"),
                    estado: rs.getString(string: "estado"),
                    telefone: rs.getString(string: "telefone"),
                    email:rs.getString(string: "email"),
                    cnpj:rs.getString(string: "cnpj"));
            pessoasJuridicas.add(e: pessoaJuridica);
        }
    } catch (SQLException e) {
        return null;
    }
    return pessoasJuridicas;
}

public boolean incluir(PessoaJuridica pessoaJuridica) {
    String sqlPessoa = "INSERT INTO Pessoas (idPessoa, nome, logradouro,"
            + " cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?
    String sqlPessoaJuridica = "INSERT INTO PessoasJuridicas (idPJuridica,"
            + " cnpj) VALUES (?, ?)";

    try (Connection conn = conector.getConnection();
        PreparedStatement stmtPessoa = conn.prepareStatement(string: sqlPessoa
        PreparedStatement stmtPessoaJuridica = conn.prepareStatement(string: s

        stmtPessoa.setInt(i: 1, i1: pessoaJuridica.getId());
        stmtPessoa.setString(i: 2, string: pessoaJuridica.getNome());
        stmtPessoa.setString(i: 3, string: pessoaJuridica.getLogradouro());
        stmtPessoa.setString(i: 4, string: pessoaJuridica.getCidade());
        stmtPessoa.setString(i: 5, string: pessoaJuridica.getEstado());
        stmtPessoa.setString(i: 6, string: pessoaJuridica.getTelefone());
        stmtPessoa.setString(i: 7, string: pessoaJuridica.getEmail());
        stmtPessoa.executeUpdate();

        stmtPessoaJuridica.setInt(i: 1, i1: pessoaJuridica.getId());
        stmtPessoaJuridica.setString(i: 2, string: pessoaJuridica.getCnpj());
        stmtPessoaJuridica.executeUpdate();

        return true;
    } catch (SQLException e) {
        return false;
    }
}

public boolean alterar(PessoaJuridica pessoaJuridica) {
    String sqlPessoa = "UPDATE Pessoas SET nome=?, logradouro=?, cidade=?,"
                     + " estado=?, telefone=?, email=? WHERE idPessoa=?";
    String sqlPessoaJuridica = "UPDATE PessoasJuridicas SET cnpj=? WHERE idP

    try (Connection conn = conector.getConnection();
         PreparedStatement stmtPessoa = conn.prepareStatement(string: sqlPesso
         PreparedStatement stmtPessoaJuridica = conn.prepareStatement(string:

```

```
                stmtPessoa.setString(i: 1, string: pessoaJuridica.getNome());
                stmtPessoa.setString(i: 2, string: pessoaJuridica.getLogradouro());
                stmtPessoa.setString(i: 3, string: pessoaJuridica.getCidade());
                stmtPessoa.setString(i: 4, string: pessoaJuridica.getEstado());
                stmtPessoa.setString(i: 5, string: pessoaJuridica.getTelefone());
                stmtPessoa.setString(i: 6, string: pessoaJuridica.getEmail());
                stmtPessoa.setInt(i: 7, i1: pessoaJuridica.getId());
                stmtPessoa.executeUpdate();

                stmtPessoaJuridica.setString(i: 1, string: pessoaJuridica.getCnpj());
                stmtPessoaJuridica.setInt(i: 2, i1: pessoaJuridica.getId());
                stmtPessoaJuridica.executeUpdate();

                return true;
            } catch (SQLException e) {
                return false;
            }
        }

        public boolean excluir(int id) {
            String sqlPessoa = "DELETE FROM Pessoas WHERE idPessoa=?";
            String sqlPessoaJuridica = "DELETE FROM PessoasJuridicas WHERE idPJuridi
             try (Connection conn = conector.getConnection();
             PreparedStatement stmtPessoaJuridica = conn.prepareStatement(string: sqlP
             PreparedStatement stmtPessoa = conn.prepareStatement(string: sqlPessoa);)

            // Excluir da tabela PessoasJuridicas primeiro
            stmtPessoaJuridica.setInt(i: 1, i1: id);
            stmtPessoaJuridica.executeUpdate();

            // Em seguida, excluir da tabela Pessoas
            stmtPessoa.setInt(i: 1, i1: id);
            stmtPessoa.executeUpdate();

            return true;
        } catch (SQLException e) {
            return false;
        }
    }
```

CadastroBDTeste:

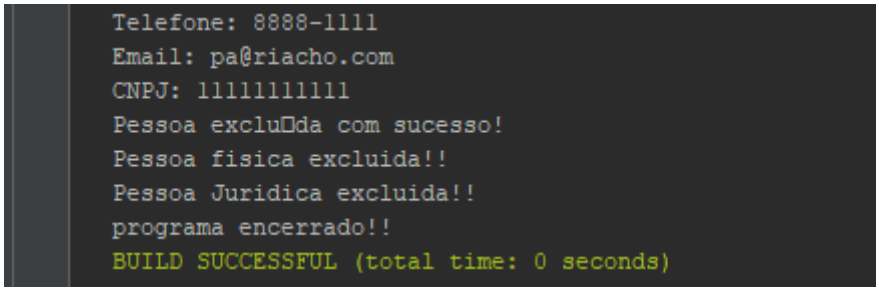
```
public class CadastroBDTeste {

    public static void main(String[] args) {
        ConectorBD conector = new ConectorBD
            (url: "jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true
                    usuario: "loja",
                    senha: "loja");

        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conector);
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(conector);
        PessoaFisica pessoaFisica = new PessoaFisica();
        PessoaJuridica pessoaJuridica = new PessoaJuridica();

        //Instanciar uma pessoa física e persistir no banco de dados.
        pessoaFisica.setId(id: 7);
        pessoaFisica.setNome(nome:"João");
        pessoaFisica.setLogradouro(logradouro: "Rua 12, casa 3, Quitanda");
        pessoaFisica.setCidade(cidade: "Riacho do Sul");
        pessoaFisica.setEstado(estado: "PA");
        pessoaFisica.setTelefone(telefone: "1111-1111");
        pessoaFisica.setEmail(email: "joao@riacho.com");
```

```
        pessoaFisica.setCpf(cpf: "11111111111");

        pessoaFisica.exibir();

        pessoaFisicaDAO.incluir(pessoa: pessoaFisica);
        System.out.println(x: "Pessoa Fisica incluida");

        pessoaJuridica.setId(id: 8);
        pessoaJuridica.setNome(nome:"JJC");
        pessoaJuridica.setLogradouro(logradouro:"Rua 11, Centro");
        pessoaJuridica.setCidade(cidade: "Riacho do Norte");
        pessoaJuridica.setEstado(estado: "PA");
        pessoaJuridica.setTelefone(telefone: "1212-1212");
        pessoaJuridica.setEmail(email:"jjc@riacho.com");
        pessoaJuridica.setCnpj(cnpj:"22222222222");

        pessoaJuridica.exibir();

        pessoaJuridicaDAO.incluir(pessoaJuridica);
        System.out.println(x: "Pessoa Juridica incluida");

        //Alterar os dados da pessoa física no banco.
        pessoaFisica.setId(id: 7);
        pessoaFisica.setNome(nome:"Paulo André");
        pessoaFisica.setLogradouro(logradouro:"Rua 20");
        pessoaFisica.setCidade(cidade: "Riacho do Norte");
        pessoaFisica.setEstado(estado: "SP");
        pessoaFisica.setTelefone(telefone: "7777-8888");
        pessoaFisica.setEmail(email:"pauloandre@riacho.com");
        pessoaFisica.setCpf(cpf: "99999999999");

        pessoaFisicaDAO.alterar(pessoa: pessoaFisica);
        System.out.println(x: "Pessoa Fisica alterada");

        pessoaJuridica.setId(id: 8);
        pessoaJuridica.setNome(nome:"PA-eletric");
        pessoaJuridica.setLogradouro(logradouro:"Rua 70");
        pessoaJuridica.setCidade(cidade: "Riacho");
        pessoaJuridica.setEstado(estado: "TO");
        pessoaJuridica.setTelefone(telefone: "8888-1111");
        pessoaJuridica.setEmail(email:"pa@riacho.com");
        pessoaJuridica.setCnpj(cnpj:"11111111111");

        pessoaJuridicaDAO.alterar(pessoaJuridica);
        System.out.println(x: "Pessoa Juridica alterada");

        //Consultar todas as pessoas físicas do banco de dados e listar no conso
        System.out.println(x: "Exibir todas Pessoa Fisicas");
        List<PessoaFisica> pessoasF = pessoaFisicaDAO.getPessoas();
        for (PessoaFisica pessoa : pessoasF) {
        pessoa.exibir();
        }

        System.out.println(x: "Exibir todas Pessoa Juridicas");
        List<PessoaJuridica> pessoasJ = pessoaJuridicaDAO.getPessoas();
        for (PessoaJuridica pessoa : pessoasJ) {
        pessoa.exibir();
        }

        //Excluir a pessoa física criada anteriormente no banco.
        pessoaFisicaDAO.excluir(id: 7);
        System.out.println(x: "Pessoa fisica excluida!!");
```

```
101
102             pessoaJuridicaDAO.excluir(id: 8 );
103             System.out.println(x: "Pessoa Juridica excluida!!");
104
105             System.out.println(x: "programa encerrado!!");
106         }
107     }
108
```

Resultados exibidos no console:

```
Output - CadastroBD (run)
ID: 1
Nome: Jo□o
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 11111111111
Pessoa Fisica incluida
ID: 2
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CNPJ: 22222222222
Pessoa Juridica incluida
Pessoa Fisica alterada
Pessoa Juridica alterada
Exibir todas Pessoas Fisicas
ID: 1
Nome: Paulo Andr□
Logradouro: Rua 20
Cidade: Riacho do Norte
Estado: SP
Telefone: 7777-8888
Email: pauloandre@riacho.com
CPF: 99999999999
ID: 7
Nome: Paulo Andr□
Logradouro: Rua 20
Cidade: Riacho do Norte
Estado: SP
Telefone: 7777-8888
Email: pauloandre@riacho.com
CPF: 99999999999
Exibir todas Pessoas Juridicas
ID: 2
Nome: PA-eletric
Logradouro: Rua 70
Cidade: Riacho
Estado: TO
```



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

Os componentes de middleware como o JDBC desempenham um papel fundamental na facilitação da comunicação e interação entre aplicativos e sistemas de gerenciamento de banco de dados. Eles proporcionam abstração, portabilidade, segurança e eficiência, contribuindo para a criação de aplicativos robustos, flexíveis e de alto desempenho.

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

O PreparedStatement é uma escolha mais segura e eficiente para manipulação de dados em bancos de dados, especialmente quando envolve a inserção de valores dinâmicos em consultas SQL. O Statement, embora simples, pode expor o aplicativo a vulnerabilidades de segurança se não for usado com cuidado.

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

O padrão DAO melhora a manutenibilidade do software ao promover a modularidade, a separação de responsabilidades, a flexibilidade e a testabilidade. Ele ajuda a criar um código mais organizado, menos propenso a erros e mais fácil de manter e evoluir ao longo do tempo.

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

A herança é refletida no banco de dados por meio da criação de tabelas separadas para cada classe herdeira PessoaFisica e PessoaJuridica, que compartilham campos comuns com a classe base Pessoa. Isso permite armazenar dados específicos de cada tipo de pessoa enquanto mantém uma estrutura organizada e eficiente no banco de dados. As tabelas separadas mantêm a integridade dos dados e possibilitam consultas específicas para cada tipo de pessoa.

2º Procedimento – Alimentando a Base

Classe SetPessoas, criada para reutilizar linhas de código:

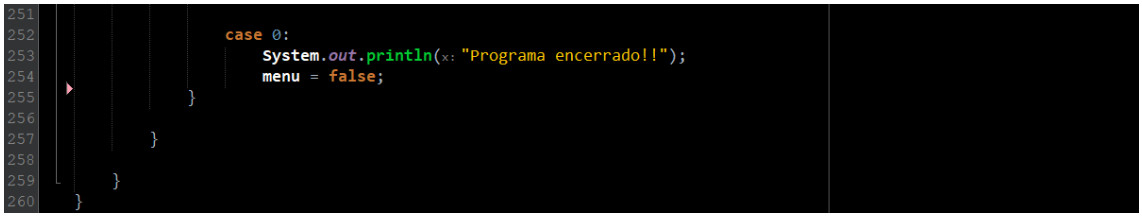
```java
public class SetPessoas {

    public static void setDadosGerais(Pessoa pessoa, Scanner scanner) {
        System.out.println(x: "Digite o nome:");
        String nome = scanner.nextLine();

        System.out.println(x: "Digite a Rua:");
        String logradouro = scanner.nextLine();

        System.out.println(x: "Digite a cidade:");
        String cidade = scanner.nextLine();

        System.out.println(x: "Digite o estado:");
        String estado = scanner.nextLine();

        System.out.println(x: "Digite o telefone:");
        String telefone = scanner.nextLine();

        System.out.println(x: "Digite o E-mail:");
        String email = scanner.nextLine();


        pessoa.setNome(nome);
        pessoa.setLogradouro(logradouro);
        pessoa.setCidade(cidade);
        pessoa.setEstado(estado);
        pessoa.setTelefone(telefone);
        pessoa.setEmail(email);
    }

    public static void setDadosPessoaFisica(PessoaFisica pessoaFisica, Scanner scanner) {
        setDadosGerais(pessoa: pessoaFisica, scanner);

        System.out.println(x: "Digite o CPF:");
        String cpf = scanner.nextLine();
        pessoaFisica.setCpf(cpf);
    }

    public static void setDadosPessoaJuridica(PessoaJuridica pessoaJuridica, Scanner scanner) {
        setDadosGerais(pessoa: pessoaJuridica, scanner);

        System.out.println(x: "Digite o CNPJ:");
        String cnpj = scanner.nextLine();
        pessoaJuridica.setCnpj(cnpj);
    }

    public static void juridicas(PessoaJuridica pessoaJuridica, Scanner scanner) {
        SetPessoas.setDadosPessoaJuridica(pessoaJuridica, scanner);
    }

    public static void fisicas(PessoaFisica pessoaFisica, Scanner scanner) {
        SetPessoas.setDadosPessoaFisica(pessoaFisica, scanner);
    }

}
```

Classe Main:

```
public class CadastroBD {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        ConectorBD conector = new ConectorBD
            (url: "jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true
                    usuario: "loja",
                    senha: "loja");
        // TODO code application logic here
        SequenceManager sequenceManager = new SequenceManager(conector);
        Scanner scanner = new Scanner(source: System.in);

        PessoaFisica pessoaFisica = new PessoaFisica();
        PessoaJuridica pessoaJuridica = new PessoaJuridica();

        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conector);
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(conector);
        boolean menu = true;
        boolean caso;
        int numMenu;
        String tipoPessoa;
        int id;


        while(menu == true){
            System.out.println(x: "=========================");
            System.out.println(x: "1 - Incluir pessoa");
            System.out.println(x: "2 - Alterar pessoa ");
            System.out.println(x: "3 - Excluir pessoa ");
            System.out.println(x: "4 - Exibir pelo ID ");
            System.out.println(x: "5 - Exibir todos ");
            System.out.println(x: "0 - Finalizar Programa");
            System.out.println(x: "=========================");

            System.out.println(x: "Digite o numero desejado: ");
            numMenu = scanner.nextInt();
            scanner.nextLine();

            switch(numMenu){
                case 1:
                    caso = true;
                    while(caso == true){
                        System.out.println(x: "Selecionado Incluir pessoa");
                        System.out.println(x: "F - Pessoa Fisica | J - Pessoa Jur
                        tipoPessoa = scanner.nextLine().toUpperCase();

                        switch(tipoPessoa){
                            case "F" -> {
                                id = sequenceManager.getValue(sequenceName: "seqId"
                                pessoaFisica.setId(id);
                                SetPessoas.fisicas(pessoaFisica, scanner);
                                pessoaFisicaDAO.incluir(pessoa: pessoaFisica);
                                caso = false;
                                break;
                            }

                            case "J" -> {
                                id = sequenceManager.getValue(sequenceName: "seqId"
                                pessoaJuridica.setId(id);
                                SetPessoas.juridicas(pessoaJuridica, scanner);
                                pessoaJuridicaDAO.incluir(pessoaJuridica);
```

```
                            caso = false;
                            break;
                        }

                        default -> {
                            System.out.println(x: "comando incorreto");
                            caso = true;
                        }
                    }

                }
            break;

            case 2:
                caso = true;
                while(caso == true){
                    System.out.println(x: "Selecionado alterar pessoa");
                    System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica");
                    tipoPessoa = scanner.nextLine().toUpperCase();

                    switch(tipoPessoa){
                        case "F" -> {
                            System.out.println(x: "Digite o ID:");
                            id = scanner.nextInt();
                            scanner.nextLine();
                            PessoaFisica pessoaF = pessoaFisicaDAO.getPessoa(id);
                            pessoaF.exibir();
                            pessoaFisica.setId(id);
                            SetPessoas.fisicas(pessoaFisica, scanner);
                            pessoaFisicaDAO.alterar(pessoa: pessoaFisica);
                            System.out.println(x: "Pessoa alterada com sucesso!");
                            caso = false;
                            break;
                        }

                        case "J" -> {
                            System.out.println(x: "Digite o ID:");
                            id = scanner.nextInt();
                            scanner.nextLine();
                            PessoaJuridica pessoaJ = pessoaJuridicaDAO.getPessoa(id);
                            pessoaJ.exibir();
                            pessoaJuridica.setId(id);
                            SetPessoas.juridicas(pessoaJuridica, scanner);
                            pessoaJuridicaDAO.alterar(pessoaJuridica);
                            System.out.println(x: "Pessoa alterada com sucesso!");
                            caso = false;
                            break;
                        }

                        default -> {
                            System.out.println(x: "comando incorreto");
                            caso = true;
                        }
                    }
                }
            break;

            case 3:
                caso = true;
                while(caso == true){
                    System.out.println(x: "Selecionado excluir pessoa");
                    System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica");
                    tipoPessoa = scanner.nextLine().toUpperCase();

                    switch(tipoPessoa){
                        case "F" -> {
                            System.out.println(x: "Digite o ID:");
                            id = scanner.nextInt();
                            scanner.nextLine();
                            PessoaFisica pessoaF = pessoaFisicaDAO.getPessoa(id);
                            pessoaF.exibir();
                            pessoaFisicaDAO.excluir(id);
                            caso = false;
                            break;
                        }

                        case "J" -> {
                            System.out.println(x: "Digite o ID:");
                            id = scanner.nextInt();
                            scanner.nextLine();
                            PessoaJuridica pessoaj = pessoaJuridicaDAO.getPessoa(id);
                            pessoaj.exibir();
                            pessoaJuridicaDAO.excluir(id);
```

```
                            caso = false;
                            break;
                        }

                        default -> {
                            System.out.println(x: "comando incorreto");
                            caso = true;
                        }

                    }

                }
            break;

            case 4:
                caso = true;
                while(caso == true){
                    System.out.println(x: "Selecionado exibir pessoa");
                    System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica");
                    tipoPessoa = scanner.nextLine().toUpperCase();

                    switch(tipoPessoa){
                        case "F" -> {
                            System.out.println(x: "Digite o ID:");
                            id = scanner.nextInt();
                            scanner.nextLine();
                            PessoaFisica pessoaF = pessoaFisicaDAO.getPessoa(id);
                            pessoaF.exibir();
                            caso = false;
                        }

                        case "J" -> {
                            System.out.println(x: "Digite o ID:");
                            id = scanner.nextInt();
                            scanner.nextLine();
                            PessoaJuridica pessoaj = pessoaJuridicaDAO.getPessoa(id);
                            pessoaj.exibir();
                            caso = false;
                        }

                        default -> {
                            System.out.println(x: "comando incorreto");
                            caso = true;
                        }

                    }

                }
            break;

            case 5:
                caso = true;
                while(caso == true){
                    System.out.println(x: "Selecionado exibir todos");
                    System.out.println(x: "F - Pessoa Fisica | J - Pessoa Juridica");
                    tipoPessoa = scanner.nextLine().toUpperCase();

                    switch(tipoPessoa){
                        case "F" -> {
                            List<PessoaFisica> pessoasF = pessoaFisicaDAO.getPessoas();
                            for (PessoaFisica pessoa : pessoasF) {
                            pessoa.exibir();
                            }
                            caso = false;
                            break;
                        }

                        case "J" -> {
                            List<PessoaJuridica> pessoasJ = pessoaJuridicaDAO.getPessoas();
                            for (PessoaJuridica pessoa : pessoasJ) {
                            pessoa.exibir();
                            }
                            caso = false;
                            break;
                        }

                        default -> {
                            System.out.println(x: "comando incorreto");
                            caso = true;
                        }

                    }

                }
            break;
```



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

A persistência em arquivo e a persistência em banco de dados diferem em como os dados são armazenados. A persistência em arquivo envolve gravar dados diretamente em arquivos, sendo simples e portátil, mas menos escalável e estruturada. Em contraste, a persistência em banco de dados usa sistemas gerenciados para armazenar dados, oferecendo estrutura, consultas avançadas e escalabilidade, embora com maior complexidade e dependência de um sistema de gerenciamento de banco de dados.

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

O uso de operadores lambda no Java permite uma forma mais concisa e legível de iterar e processar coleções de dados. Nas versões mais recentes do Java, os operadores lambda simplificaram a impressão dos valores contidos nas entidades, reduzindo a necessidade de escrever loops tradicionais e permitindo a definição de funções inline para manipulação de dados. Isso resulta em um código mais limpo e eficiente, aumentando a produtividade e a clareza na programação.

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

Métodos acionados diretamente pelo método main em Java precisam ser marcados como static porque o método main é estático e pertence à classe em si, não a uma instância dela. Isso significa que ele pode ser chamado sem a necessidade de criar um objeto da classe. Métodos não estáticos pertencem a instâncias da classe e requerem um objeto para serem invocados, o que não é adequado quando chamados diretamente do método main. Marcando os métodos como static, eles podem ser acessados diretamente sem a criação de instâncias.

Conclusão

Neste trabalho, desenvolvi um sistema de cadastro de pessoas que abrange tanto pessoas físicas quanto jurídicas. Utilizei uma abordagem orientada a objetos, criando

classes como Pessoa, PessoaFisica e PessoaJuridica, cada uma com seus respectivos atributos e métodos. Implementei operações de CRUD (criação, leitura, atualização e exclusão) através de classes DAO, garantindo a interação com o banco de dados de forma eficiente e organizada. A aplicação conta com um menu de fácil utilização, permitindo ao usuário realizar diversas ações, como inclusão, alteração, exclusão e exibição de pessoas.