# Nível 3: Back-end Sem Banco Não Tem

Rodrigo de O. Alarcon - 202204482321

Vila Mariana
Vamos Manter as Informações? – 2022.2 – 3º Semestre

## Objetivo da Prática

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

A função principal do middleware é a comunicação de dados e a conexão de aplicações, facilitando o desenvolvimento e elaboração de tarefas. JDBC é um Driver de conexão da Aplicação com o SGBD.

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

Um Statement irá sempre passar pelos quatro passos abaixo para cada consulta SQL enviada para o banco.

- ➔ Iterpretar a consulta SQL.
- ➔ Compilar a consulta SQL.
- ➔ Planejar e otimizar o caminho de busca dos dados.
- ➔ Executar a consulta otimizada, buscando e retornando os dados.

Um PreparedStatement executa de imediato:

- ➔ Interptear
- ➔ Planejar e otimizar o caminho de busca dos dados

## Como o padrão DAO melhora a manutenibilidade do software?

Anteriormente os códigos eram todos agrupados e com dificil manutenibilidade dos códigos. Com o padrão aplicado, cada pack e classe possui sua tarefa única e exclusiva. Sendo assim, facilitando no momento de uma alteração e ou manutenção.

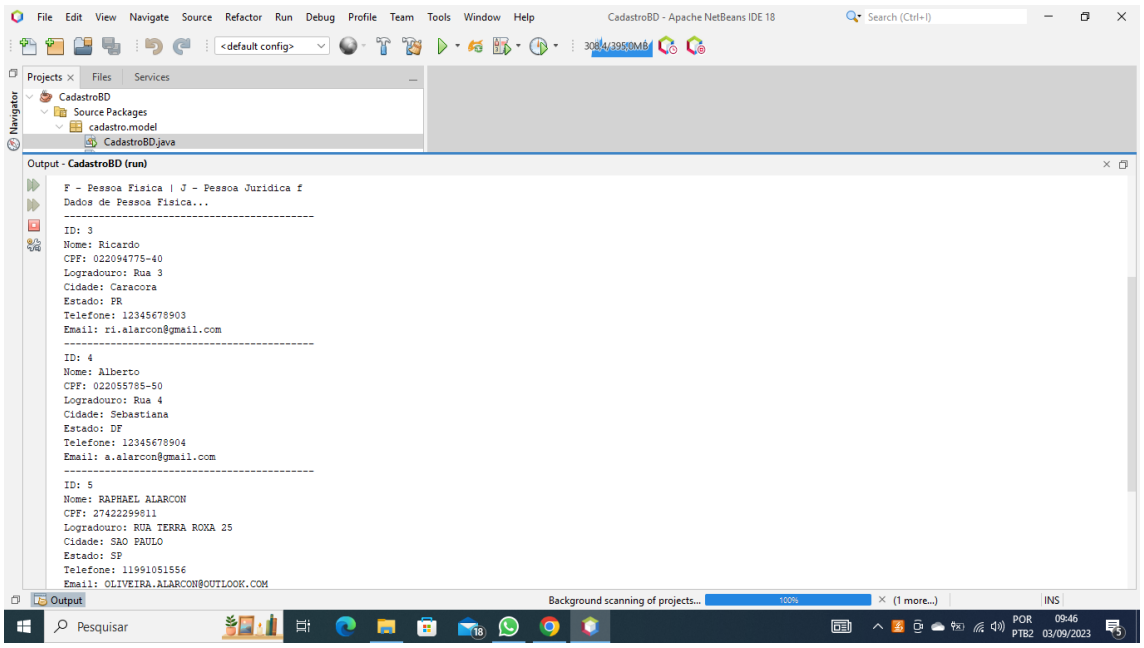
## Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?

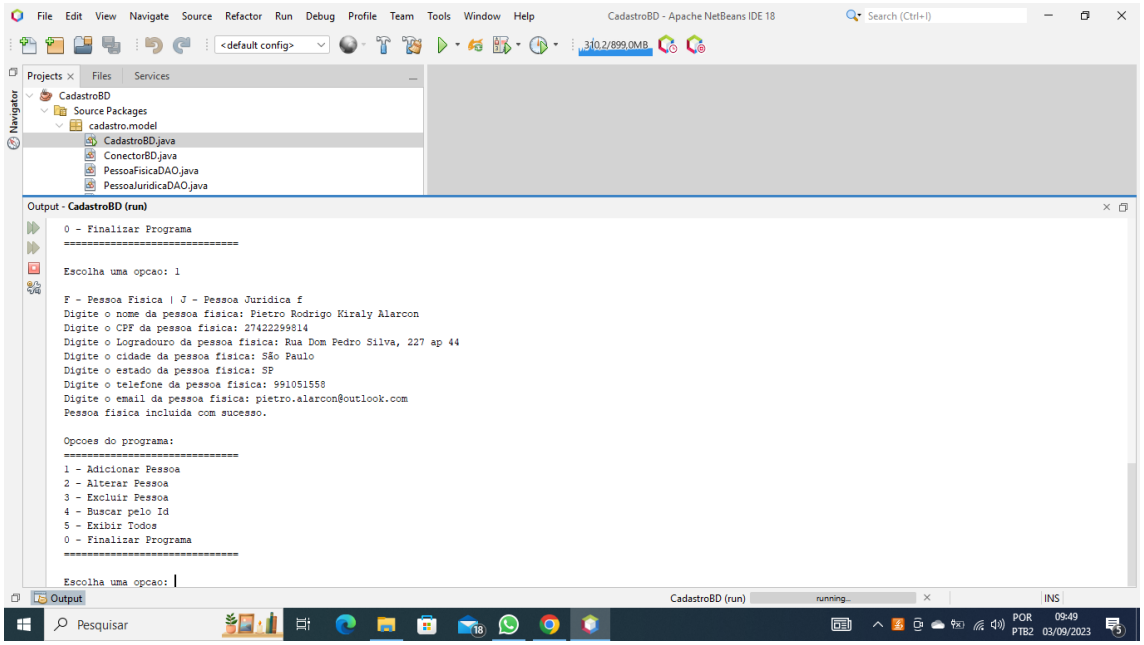
Persistência em arquivo serve para pequenas aplicações desktop onde a gestão e integridade dos dados não são tão rígidas.

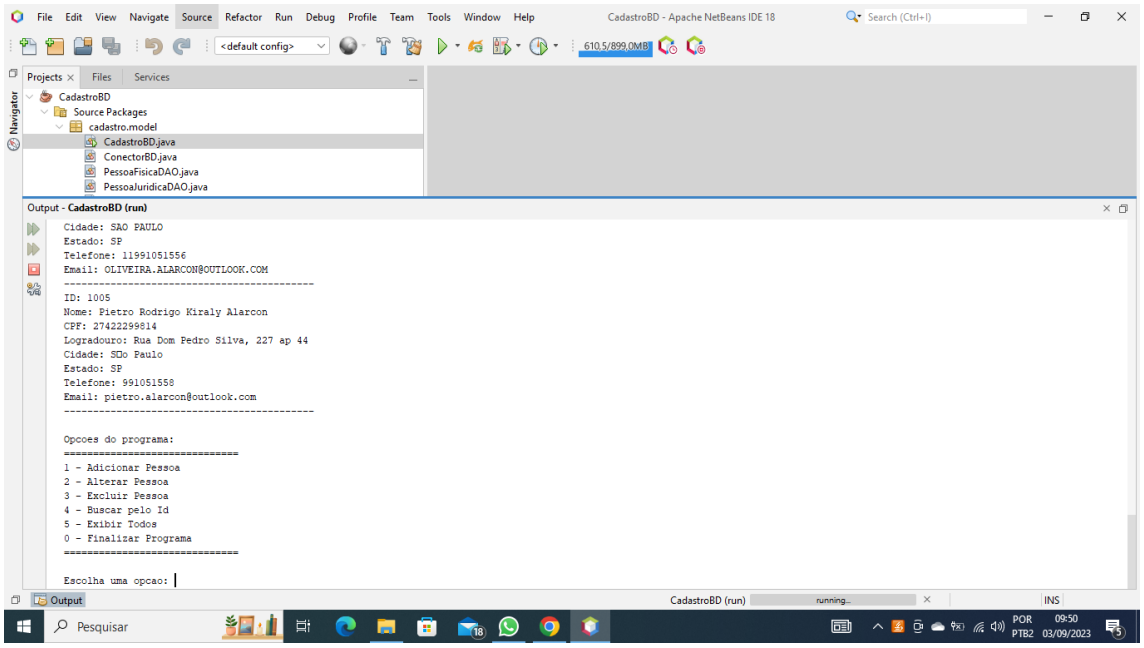
A persistência em SGBD exige uma gestão mais rígida e preocupação com a integridade dos dados.

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

Pelo fato de não possuirem necessiade de se criar uma instância.




File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
CadastroBD - Apache NetBeans IDE 18
Search (Ctrl+I)
<default config>
310.2/899.0MB
Projects
Files
Services
CadastroBD
Source Packages
cadastro.model
CadastroBD.java
ConectorBD.java
PessoaFisicaDAO.java
PessoaJuridicaDAO.java
Output - CadastroBD (run)
0 - Finalizar Programa
==========
Escolha uma opcao: 1
F - Pessoa Fisica | J - Pessoa Juridica f
Digite o nome da pessoa fisica: Pietro Rodrigo Kiraly Alarcon
Digite o CPF da pessoa fisica: 27422299814
Digite o Logradouro da pessoa fisica: Rua Dom Pedro Silva, 227 ap 44
Digite o cidade da pessoa fisica: São Paulo
Digite o estado da pessoa fisica: SP
Digite o telefone da pessoa fisica: 991051558
Digite o email da pessoa fisica: pietro.alarcon@outlook.com
Pessoa fisica incluida com sucesso.
Opcoes do programa:
==========
1 - Adicionar Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==========
Escolha uma opcao:
Output
CadastroBD (run)
running...
INS
Pesquisar
POR
PTB2
09:49
03/09/2023


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
CadastroBD - Apache NetBeans IDE 18
Search (Ctrl+I)
<default config>
610.5/899.0MB
Projects
Files
Services
CadastroBD
Source Packages
cadastro.model
CadastroBD.java
ConectorBD.java
PessoaFisicaDAO.java
PessoaJuridicaDAO.java
Output - CadastroBD (run)
Cidade: SAO PAULO
Estado: SP
Telefone: 11991051556
Email: OLIVEIRA.ALARCON@OUTLOOK.COM
ID: 1005
Nome: Pietro Rodrigo Kiraly Alarcon
CPF: 27422299814
Logradouro: Rua Dom Pedro Silva, 227 ap 44
Cidade: S□o Paulo
Estado: SP
Telefone: 991051558
Email: pietro.alarcon@outlook.com
Opcoes do programa:
==========
1 - Adicionar Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==========
Escolha uma opcao:
Output
CadastroBD (run)
running...
INS
Pesquisar
POR
PTB2
09:50
03/09/2023

```
/* Alarcon ABAP
 */
package cadastro.model;
import cadastrointerface.InterfaceCadastro;
import java.util.Scanner;

public class CadastroBD {

    public static void main(String[] args) {
        try (Scanner scanner = new Scanner(System.in)) {
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
            PessoaJuridicaDAO pessoaJuridicaDAO = new
PessoaJuridicaDAO();

            boolean continuar = true;
            while (continuar) {
                InterfaceCadastro.exibirOpcoes();

                int opcao = InterfaceCadastro.lerOpcao(scanner);

                switch (opcao) {
                    case 1->    {
                                InterfaceCadastro.realizarInclusao
                                (scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                    }
                    case 2 -> {
                                InterfaceCadastro.realizarAlteracao
                                (scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                    break;
                    }
                    case 3 -> {
                                InterfaceCadastro.realizarExclusao
                                (scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                    break;
                    }
                    case 4 -> {

InterfaceCadastro.realizarObtencaoPorID
                                (scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                    break;
                    }
                    case 5 -> {
                                InterfaceCadastro.realizarListagem
                                (scanner, pessoaFisicaDAO,
pessoaJuridicaDAO);
                    break;
                    }
                    case 0 -> continuar = false;
                    default -> System.out.println("OpÃ§Ã£o
invÃ¡lida.");
                }
            }
        }
    }
}
```

```
/*
    Autor: Alarcon ABAP
*/

package cadastro.model;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ConectorBD {

    // ObtÃ©m uma conexÃ£o com o banco de dados
    public Connection getConnection() throws SQLException {
        String url =
"jdbc:sqlserver://localhost:1433;databaseName=Loja;encrypt=true;trustServerCertificate=t
rue";
        String user = "sa";
        String password = "sarava";
        return DriverManager.getConnection(url, user, password);
    }

    // Cria um PreparedStatement para executar uma consulta parametrizada
    public PreparedStatement getPrepared(String sql) throws SQLException {
        Connection connection = getConnection();
        return connection.prepareStatement(sql);
    }

    // Executa uma consulta SELECT e retorna o resultado como um ResultSet
    public ResultSet getSelect(String sql) throws SQLException {
        Statement statement = getConnection().createStatement();
        return statement.executeQuery(sql);
    }

    // Fecha um objeto Statement
    public void close(Statement statement) {
        try {
            if (statement != null) {
                statement.close();
            }
        } catch (SQLException e) {
        }
    }

    // Fecha um objeto ResultSet
    public void close(ResultSet resultSet) {
        try {
            if (resultSet != null) {
                resultSet.close();
            }
        } catch (SQLException e) {
        }
    }

    // Fecha uma conexÃ£o com o banco de dados
    public void close(Connection connection) {
        try {
            if (connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
        }
    }

        Statement getStatement() {
         return null; // ou return new Statement();
     }

}
```

```
/*
    Autor: Alarcon ABAP
*/

package cadastro.model;
import java.sql.Statement;
import cadastrobd.model.PessoaFisica;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaDAO {

    private final ConectorBD conectorBD;

    public PessoaFisicaDAO() {
        this.conectorBD = new ConectorBD();
    }

    // MÃ©todo para obter uma PessoaFisica pelo ID
    public PessoaFisica getPessoa(int id) {
        PessoaFisica pessoaFisica = null;
        Connection connection = null;
        PreparedStatement statementPessoa = null;
        PreparedStatement statementPessoaFisica = null;
        ResultSet resultSetPessoa = null;
        ResultSet resultSetPessoaFisica = null;

        try {
            connection = conectorBD.getConnection();

            // Consultar a tabela Pessoa
            String sqlPessoa = "SELECT * FROM Pessoa WHERE idPessoa = ?";
            statementPessoa = connection.prepareStatement(sqlPessoa);
            statementPessoa.setInt(1, id);
            resultSetPessoa = statementPessoa.executeQuery();

            if (resultSetPessoa.next()) {
                pessoaFisica = new PessoaFisica();
                pessoaFisica.setId(resultSetPessoa.getInt("idPessoa"));
                pessoaFisica.setNome(resultSetPessoa.getString("nome"));
                pessoaFisica.setLogradouro(resultSetPessoa.getString("logradouro"));
                pessoaFisica.setCidade(resultSetPessoa.getString("cidade"));
                pessoaFisica.setEstado(resultSetPessoa.getString("estado"));
                pessoaFisica.setTelefone(resultSetPessoa.getString("telefone"));
                pessoaFisica.setEmail(resultSetPessoa.getString("email"));
            }

            // Consultar a tabela PessoaFisica
            String sqlPessoaFisica = "SELECT * FROM PessoaFisica WHERE idPessoa = ?";
            statementPessoaFisica = connection.prepareStatement(sqlPessoaFisica);
            statementPessoaFisica.setInt(1, id);
            resultSetPessoaFisica = statementPessoaFisica.executeQuery();

            if (resultSetPessoaFisica.next()) {
                pessoaFisica.setCpf(resultSetPessoaFisica.getString("CPF"));
                // Definir outros atributos especÃficos da tabela PessoaFisica, se
houver
            }

        } catch (SQLException e) {
            // Tratar exceÃ§Ã£o, se necessÃ¡rio
        } finally {
            conectorBD.close(resultSetPessoaFisica);
            conectorBD.close(statementPessoaFisica);
            conectorBD.close(resultSetPessoa);
            conectorBD.close(statementPessoa);
            conectorBD.close(connection);
        }

        return pessoaFisica;
    }
```

```
    // MÃ©todo para listar todas as pessoas fÃsicas
    public List<PessoaFisica> listarTodasPessoasFisicas() {
        List<PessoaFisica> pessoasFisicas = new ArrayList<>();
        Connection connection = null;
        PreparedStatement statementPessoa = null;
        PreparedStatement statementPessoaFisica = null;
        ResultSet resultSetPessoa = null;
        ResultSet resultSetPessoaFisica = null;

        try {
            connection = conectorBD.getConnection();

            // Consultar a tabela Pessoa
            String sqlPessoa = "SELECT * FROM PessoaFisica";
            statementPessoa = connection.prepareStatement(sqlPessoa);
            resultSetPessoa = statementPessoa.executeQuery();

            while (resultSetPessoa.next()) {
                PessoaFisica pessoaFisica = new PessoaFisica();
                pessoaFisica.setId(resultSetPessoa.getInt("idPessoa"));
                pessoaFisica.setNome(resultSetPessoa.getString("nome"));
                pessoaFisica.setLogradouro(resultSetPessoa.getString("logradouro"));
                pessoaFisica.setCidade(resultSetPessoa.getString("cidade"));
                pessoaFisica.setEstado(resultSetPessoa.getString("estado"));
                pessoaFisica.setTelefone(resultSetPessoa.getString("telefone"));
                pessoaFisica.setEmail(resultSetPessoa.getString("email"));

                // Consultar a tabela PessoaFisica para buscar o CPF, se houver
                String sqlPessoaFisica = "SELECT * FROM PessoaFisica WHERE idPessoa =
?";
                statementPessoaFisica = connection.prepareStatement(sqlPessoaFisica);
                statementPessoaFisica.setInt(1, pessoaFisica.getId());
                resultSetPessoaFisica = statementPessoaFisica.executeQuery();

                if (resultSetPessoaFisica.next()) {
                    pessoaFisica.setCpf(resultSetPessoaFisica.getString("CPF"));
                    // Definir outros atributos especÃficos da tabela PessoaFisica, se
houver
                }

                pessoasFisicas.add(pessoaFisica);
            }
        } catch (SQLException e) {
            // Tratar exceÃ§Ã£o, se necessÃ¡rio
        } finally {
            conectorBD.close(resultSetPessoaFisica);
            conectorBD.close(statementPessoaFisica);
            conectorBD.close(resultSetPessoa);
            conectorBD.close(statementPessoa);
            conectorBD.close(connection);
        }

        return pessoasFisicas;
    }

    // Método para incluir uma PessoaFisica no banco de dados
   public void incluir(PessoaFisica pessoaFisica) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        connection = conectorBD.getConnection();
        connection.setAutoCommit(false); // Inicia uma transaÃ§Ã£o

        // Inserir na tabela Pessoa
        String sqlPessoa = "INSERT INTO Pessoa (nome, logradouro, cidade, estado,
telefone, email) VALUES (?, ?, ?, ?, ?, ?)";
        statement = connection.prepareStatement(sqlPessoa,
Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, pessoaFisica.getNome());
        statement.setString(2, pessoaFisica.getLogradouro());
        statement.setString(3, pessoaFisica.getCidade());
        statement.setString(4, pessoaFisica.getEstado());
        statement.setString(5, pessoaFisica.getTelefone());
```

```
        statement.setString(6, pessoaFisica.getEmail());
        statement.executeUpdate();

        // Obter o ID gerado para a Pessoa
        ResultSet generatedKeys = statement.getGeneratedKeys();
        int idPessoa = 0;
        if (generatedKeys.next()) {
            idPessoa = generatedKeys.getInt(1);
        }

        // Inserir na tabela PessoaFisica
        String sqlPessoaFisica = "INSERT INTO PessoaFisica (idPessoa, nome, CPF,
logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
        statement = connection.prepareStatement(sqlPessoaFisica);
        statement.setInt(1, idPessoa);
        statement.setString(2, pessoaFisica.getNome());
        statement.setString(3, pessoaFisica.getCpf());
        statement.setString(4, pessoaFisica.getLogradouro());
        statement.setString(5, pessoaFisica.getCidade());
        statement.setString(6, pessoaFisica.getEstado());
        statement.setString(7, pessoaFisica.getTelefone());
        statement.setString(8, pessoaFisica.getEmail());
        statement.executeUpdate();

        connection.commit();
    } catch (SQLException e) {
        if (connection != null) {
            try {
                connection.rollback();
            } catch (SQLException ex) {
            }
        }
    } finally {
        conectorBD.close(statement);
        conectorBD.close(connection);
    }
}



    public void alterar(PessoaFisica pessoaFisica) {
       Connection connection = null;
       PreparedStatement statement = null;

       try {
           connection = conectorBD.getConnection();
           connection.setAutoCommit(false); // Inicia uma transaÃ§Ã£o

           // Verifica se a PessoaFisica existe no banco de dados
           String sqlVerificarExistencia = "SELECT idPessoa FROM PessoaFisica WHERE
idPessoa = ?";
           statement = connection.prepareStatement(sqlVerificarExistencia);
           statement.setInt(1, pessoaFisica.getId());
           ResultSet resultSet = statement.executeQuery();

           if (resultSet.next()) {
               // Atualizar tabela PessoaFisica
               String sqlPessoaFisica = "UPDATE PessoaFisica SET nome = ?, cpf = ?,
logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
               statement = connection.prepareStatement(sqlPessoaFisica);
               statement.setString(1, pessoaFisica.getNome());
               statement.setString(2, pessoaFisica.getCpf());
               statement.setString(3, pessoaFisica.getLogradouro());
               statement.setString(4, pessoaFisica.getCidade());
               statement.setString(5, pessoaFisica.getEstado());
               statement.setString(6, pessoaFisica.getTelefone());
               statement.setString(7, pessoaFisica.getEmail());
               statement.setInt(8, pessoaFisica.getId());

               int rowsAffected = statement.executeUpdate();

               if (rowsAffected == 0) {
                   throw new SQLException("Nenhuma linha foi alterada.");
               }
```

```
                connection.commit(); // Confirma a transaÃ§Ã£o
            } else {
                throw new SQLException("A PessoaFisica com o ID especificado nÃ£o existe
no banco de dados.");
            }
        } catch (SQLException e) {
            if (connection != null) {
                try {
                    connection.rollback(); // Desfaz a transaÃ§Ã£o em caso de erro
                } catch (SQLException ex) {
                }
            }
        } finally {
            conectorBD.close(statement);
            conectorBD.close(connection);
        }
    }


    public void excluir(int id) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        connection = conectorBD.getConnection();
        connection.setAutoCommit(false); // Inicia uma transaÃ§Ã£o

        // Verificar se a PessoaFisica existe no banco de dados
        String sqlVerificarExistencia = "SELECT idPessoa FROM PessoaFisica WHERE
idPessoa = ?";
        statement = connection.prepareStatement(sqlVerificarExistencia);
        statement.setInt(1, id);
        ResultSet resultSet = statement.executeQuery();

        if (resultSet.next()) {
            // Excluir da tabela PessoaFisica
            String sqlExcluirPessoaFisica = "DELETE FROM PessoaFisica WHERE idPessoa =
?";
            statement = connection.prepareStatement(sqlExcluirPessoaFisica);
            statement.setInt(1, id);
            statement.executeUpdate();

            // Excluir da tabela Pessoa
            String sqlExcluirPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
            statement = connection.prepareStatement(sqlExcluirPessoa);
            statement.setInt(1, id);
            statement.executeUpdate();

            connection.commit(); // Confirma a transaÃ§Ã£o
            System.out.println("Pessoa fÃsica excluÃda com sucesso.");
        } else {
            System.out.println("A pessoa com o ID especificado nÃ£o foi encontrada.");
        }
    } catch (SQLException e) {
        if (connection != null) {
            try {
                connection.rollback(); // Desfaz a transaÃ§Ã£o em caso de erro
            } catch (SQLException ex) {
            }
        }
    } finally {
        conectorBD.close(statement);
        conectorBD.close(connection);
    }
}
```

```
    // Método para buscar uma PessoaFisica por ID
    public PessoaFisica buscarPorId(int idPessoa) {
        Connection connection = null;
        PreparedStatement statement = null;
        ResultSet resultSet = null;
        PessoaFisica pessoaFisica = null;

        try {
            connection = conectorBD.getConnection();

            String sql = "SELECT * FROM PessoaFisica WHERE idPessoa = ?";
            statement = connection.prepareStatement(sql);
            statement.setInt(1, idPessoa);

            resultSet = statement.executeQuery();

            if (resultSet.next()) {
                // Obter os dados da pessoa fÃsica do ResultSet
                int id = resultSet.getInt("idPessoaFisica");
                String nome = resultSet.getString("nome");
                String cpf = resultSet.getString("CPF");
                String logradouro = resultSet.getString("logradouro");
                String cidade = resultSet.getString("cidade");
                String estado = resultSet.getString("estado");
                String telefone = resultSet.getString("telefone");
                String email = resultSet.getString("email");

                // Outros atributos da pessoa fÃsica...

                // Criar um objeto PessoaFisica com os dados obtidos
                pessoaFisica = new PessoaFisica();
                pessoaFisica.setId(id);
                pessoaFisica.setNome(nome);
                pessoaFisica.setCpf(cpf);
                pessoaFisica.setLogradouro(logradouro);
                pessoaFisica.setCidade(cidade);
                pessoaFisica.setEstado(estado);
                pessoaFisica.setTelefone(telefone);
                pessoaFisica.setEmail(email);
                // Definir os outros atributos da pessoa fÃsica no objeto
            }

        } catch (SQLException e) {
            // Tratar exceÃ§Ã£o, se necessÃ¡rio
        } finally {
            // Fechar recursos (ResultSet, PreparedStatement, Connection), se
necessÃ¡rio
        }

        return pessoaFisica;
    }

    // MÃ©todo para listar todas as pessoas fÃsicas
   public List<PessoaFisica> getPessoasFisicasPorId(int id) {
    List<PessoaFisica> pessoasFisicas = new ArrayList<>();
    Connection connection = null;
    PreparedStatement statementPessoaFisica = null;
    ResultSet resultSetPessoaFisica = null;

    try {
        connection = conectorBD.getConnection();

        // Consultar a tabela PessoaFisica
        String sqlPessoaFisica = "SELECT * FROM PessoaFisica"; // Verifique se o nome da
tabela estÃ¡ correto
        statementPessoaFisica = connection.prepareStatement(sqlPessoaFisica);
        resultSetPessoaFisica = statementPessoaFisica.executeQuery();

        while (resultSetPessoaFisica.next()) {
            PessoaFisica pessoaFisica = new PessoaFisica();
            pessoaFisica.setId(resultSetPessoaFisica.getInt("idPessoaFisica")); //
Verifique se o nome da coluna estÃ¡ correto
            pessoaFisica.setNome(resultSetPessoaFisica.getString("nome")); // Verifique
se o nome da coluna estÃ¡ correto
```

```
            pessoaFisica.setCpf(resultSetPessoaFisica.getString("CPF")); // Verifique se
o nome da coluna estÃ¡ correto
            pessoaFisica.setLogradouro(resultSetPessoaFisica.getString("logradouro"));
// Verifique se o nome da coluna estÃ¡ correto
            pessoaFisica.setCidade(resultSetPessoaFisica.getString("cidade")); //
Verifique se o nome da coluna estÃ¡ correto
            pessoaFisica.setEstado(resultSetPessoaFisica.getString("estado")); //
Verifique se o nome da coluna estÃ¡ correto
            pessoaFisica.setTelefone(resultSetPessoaFisica.getString("telefone")); //
Verifique se o nome da coluna estÃ¡ correto
            pessoaFisica.setEmail(resultSetPessoaFisica.getString("email")); //
Verifique se o nome da coluna estÃ¡ correto
            // Definir outros atributos da pessoa fÃsica, se houver

            pessoasFisicas.add(pessoaFisica);
        }
    } catch (SQLException e) {
        // Tratar exceÃ§Ã£o, se necessÃ¡rio
        e.printStackTrace(); // Imprime a stack trace do erro para facilitar a
depuraÃ§Ã£o
    } finally {
        conectorBD.close(resultSetPessoaFisica);
        conectorBD.close(statementPessoaFisica);
        conectorBD.close(connection);
    }

    return pessoasFisicas;
}
}
```

```
/*
    Autor: Alarcon ABAP
*/

package cadastro.model;
import java.sql.Statement;
import cadastrobd.model.PessoaJuridica;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class PessoaJuridicaDAO {

    private final ConectorBD conectorBD;

    public PessoaJuridicaDAO() {
        this.conectorBD = new ConectorBD();
    }

    // MÃ©todo para obter uma PessoaJuridica pelo ID
    public PessoaJuridica getPessoa(int id) {
        PessoaJuridica pessoaJuridica = null;
        Connection connection = null;
        PreparedStatement statementPessoa = null;
        PreparedStatement statementPessoaJuridica = null;
        ResultSet resultSetPessoa = null;
        ResultSet resultSetPessoaJuridica = null;

        try {
            connection = conectorBD.getConnection();

            // Consultar a tabela Pessoa
            String sqlPessoa = "SELECT * FROM PessoaJuridica WHERE idPessoa = ?";
            statementPessoa = connection.prepareStatement(sqlPessoa);
            statementPessoa.setInt(1, id);
            resultSetPessoa = statementPessoa.executeQuery();

            if (resultSetPessoa.next()) {
                pessoaJuridica = new PessoaJuridica();
                pessoaJuridica.setId(resultSetPessoa.getInt("idPessoa"));
                pessoaJuridica.setNome(resultSetPessoa.getString("nome"));
                pessoaJuridica.setLogradouro(resultSetPessoa.getString("logradouro"));
                pessoaJuridica.setCidade(resultSetPessoa.getString("cidade"));
                pessoaJuridica.setEstado(resultSetPessoa.getString("estado"));
                pessoaJuridica.setTelefone(resultSetPessoa.getString("telefone"));
                pessoaJuridica.setEmail(resultSetPessoa.getString("email"));
            }

            // Consultar a tabela PessoaJuridica
            String sqlPessoaJuridica = "SELECT * FROM PessoaJuridica WHERE idPessoa =
?";
            statementPessoaJuridica = connection.prepareStatement(sqlPessoaJuridica);
            statementPessoaJuridica.setInt(1, id);
            resultSetPessoaJuridica = statementPessoaJuridica.executeQuery();

            if (resultSetPessoaJuridica.next()) {
                pessoaJuridica.setCnpj(resultSetPessoaJuridica.getString("CNPJ"));
                // Definir outros atributos especÃficos da tabela PessoaJuridica, se
houver
            }

        } catch (SQLException e) {
            // Tratar exceÃ§Ã£o, se necessÃ¡rio
        } finally {
            conectorBD.close(resultSetPessoaJuridica);
            conectorBD.close(statementPessoaJuridica);
            conectorBD.close(resultSetPessoa);
            conectorBD.close(statementPessoa);
            conectorBD.close(connection);
        }

        return pessoaJuridica;
```

```
    }

    // Método para listar todas as pessoas jurÃdicas
    public List<PessoaJuridica> listarTodasPessoasJuridicas() {
        List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
        Connection connection = null;
        PreparedStatement statementPessoa = null;
        PreparedStatement statementPessoaJuridica = null;
        ResultSet resultSetPessoa = null;
        ResultSet resultSetPessoaJuridica = null;

        try {
            connection = conectorBD.getConnection();

            // Consultar a tabela Pessoa
            String sqlPessoa = "SELECT * FROM PessoaJuridica";
            statementPessoa = connection.prepareStatement(sqlPessoa);
            resultSetPessoa = statementPessoa.executeQuery();

            while (resultSetPessoa.next()) {
                PessoaJuridica pessoaJuridica = new PessoaJuridica();
                pessoaJuridica.setId(resultSetPessoa.getInt("idPessoa"));
                pessoaJuridica.setNome(resultSetPessoa.getString("nome"));
                pessoaJuridica.setLogradouro(resultSetPessoa.getString("logradouro"));
                pessoaJuridica.setCidade(resultSetPessoa.getString("cidade"));
                pessoaJuridica.setEstado(resultSetPessoa.getString("estado"));
                pessoaJuridica.setTelefone(resultSetPessoa.getString("telefone"));
                pessoaJuridica.setEmail(resultSetPessoa.getString("email"));

                // Consultar a tabela PessoaJuridica para buscar o CNPJ, se houver
                String sqlPessoaJuridica = "SELECT * FROM PessoaJuridica WHERE idPessoa
= ?";
                statementPessoaJuridica =
connection.prepareStatement(sqlPessoaJuridica);
                statementPessoaJuridica.setInt(1, pessoaJuridica.getId());
                resultSetPessoaJuridica = statementPessoaJuridica.executeQuery();

                if (resultSetPessoaJuridica.next()) {
                    pessoaJuridica.setCnpj(resultSetPessoaJuridica.getString("CNPJ"));
                    // Definir outros atributos especÃficos da tabela PessoaJuridica, se
houver
                }

                pessoasJuridicas.add(pessoaJuridica);
            }
        } catch (SQLException e) {
            // Tratar exceÃ§Ã£o, se necessÃ¡rio
        } finally {
            conectorBD.close(resultSetPessoaJuridica);
            conectorBD.close(statementPessoaJuridica);
            conectorBD.close(resultSetPessoa);
            conectorBD.close(statementPessoa);
            conectorBD.close(connection);
        }

        return pessoasJuridicas;
    }

    // MÃ©todo para incluir uma PessoaJuridica no banco de dados
   public void incluir(PessoaJuridica pessoaJuridica) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        connection = conectorBD.getConnection();
        connection.setAutoCommit(false); // Inicia uma transaÃ§Ã£o

        // Inserir na tabela Pessoa
        String sqlPessoa = "INSERT INTO Pessoa (nome, logradouro, cidade, estado,
telefone, email) VALUES (?, ?, ?, ?, ?, ?)";
        statement = connection.prepareStatement(sqlPessoa,
Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, pessoaJuridica.getNome());
        statement.setString(2, pessoaJuridica.getLogradouro());
        statement.setString(3, pessoaJuridica.getCidade());
```

```
        statement.setString(4, pessoaJuridica.getEstado());
        statement.setString(5, pessoaJuridica.getTelefone());
        statement.setString(6, pessoaJuridica.getEmail());
        statement.executeUpdate();

        // Obter o ID gerado para a Pessoa
        ResultSet generatedKeys = statement.getGeneratedKeys();
        int idPessoa = 0;
        if (generatedKeys.next()) {
            idPessoa = generatedKeys.getInt(1);
        }

        // Inserir na tabela PessoaJuridica
        String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoa, nome, CNPJ,
logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
        statement = connection.prepareStatement(sqlPessoaJuridica);
        statement.setInt(1, idPessoa);
        statement.setString(2, pessoaJuridica.getNome());
        statement.setString(3, pessoaJuridica.getCnpj());
        statement.setString(4, pessoaJuridica        .getLogradouro());
        statement.setString(5, pessoaJuridica.getCidade());
        statement.setString(6, pessoaJuridica.getEstado());
        statement.setString(7, pessoaJuridica.getTelefone());
        statement.setString(8, pessoaJuridica.getEmail());
        statement.executeUpdate();

        connection.commit(); // Confirma a transaÃ§Ã£o
    } catch (SQLException e) {
        if (connection != null) {
            try {
                connection.rollback(); // Desfaz a transaÃ§Ã£o em caso de erro
            } catch (SQLException ex) {
            }
        }
    } finally {
        conectorBD.close(statement);
        conectorBD.close(connection);
    }
}

public void alterar(PessoaJuridica pessoaJuridica) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        connection = conectorBD.getConnection();
        connection.setAutoCommit(false); // Inicia uma transaÃ§Ã£o

        // Verifica se a PessoaJuridica existe no banco de dados
        String sqlVerificarExistencia = "SELECT idPessoa FROM PessoaJuridica WHERE
idPessoa = ?";
        statement = connection.prepareStatement(sqlVerificarExistencia);
        statement.setInt(1, pessoaJuridica.getId());
        ResultSet resultSet = statement.executeQuery();

        if (resultSet.next()) {
            // Atualizar tabela PessoaJuridica
            String sqlPessoaJuridica = "UPDATE PessoaJuridica SET nome = ?, CNPJ = ?,
logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
            statement = connection.prepareStatement(sqlPessoaJuridica);
            statement.setString(1, pessoaJuridica.getNome());
            statement.setString(2, pessoaJuridica.getCnpj());
            statement.setString(3, pessoaJuridica.getLogradouro());
            statement.setString(4, pessoaJuridica.getCidade());
            statement.setString(5, pessoaJuridica.getEstado());
            statement.setString(6, pessoaJuridica.getTelefone());
            statement.setString(7, pessoaJuridica.getEmail());
            statement.setInt(8, pessoaJuridica.getId());

            int rowsAffected = statement.executeUpdate();

            if (rowsAffected == 0) {
                throw new SQLException("Nenhuma linha foi atualizada. Verifique se o ID
da Pessoa Juridica estÃ¡ correto.");
            }
```

```
            connection.commit(); // Confirma a transaÃ§Ã£o
        } else {
            throw new SQLException("A Pessoa Juridica com o ID especificado nÃ£o existe
no banco de dados.");
        }
    } catch (SQLException e) {
        if (connection != null) {
            try {
                connection.rollback(); // Desfaz a transaÃ§Ã£o em caso de erro
            } catch (SQLException ex) {
            }
        }
    } finally {
        conectorBD.close(statement);
        conectorBD.close(connection);
    }
}

public void excluir(int id) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        connection = conectorBD.getConnection();
        connection.setAutoCommit(false); // Inicia uma transaÃ§Ã£o

        // Verificar se a PessoaJuridica existe no banco de dados
        String sqlVerificarExistencia = "SELECT idPessoa FROM PessoaJuridica WHERE
idPessoa = ?";
        statement = connection.prepareStatement(sqlVerificarExistencia);
        statement.setInt(1, id);
        ResultSet resultSet = statement.executeQuery();

        if (resultSet.next()) {
            // Excluir da tabela PessoaJuridica
            String sqlExcluirPessoaJuridica = "DELETE FROM PessoaJuridica WHERE idPessoa
= ?";
            statement = connection.prepareStatement(sqlExcluirPessoaJuridica);
            statement.setInt(1, id);
            statement.executeUpdate();

            // Excluir da tabela Pessoa
            String sqlExcluirPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
            statement = connection.prepareStatement(sqlExcluirPessoa);
            statement.setInt(1, id);
            statement.executeUpdate();

            connection.commit(); // Confirma a transaÃ§Ã£o
            System.out.println("Pessoa JurÃdica excluÃda com sucesso.");
        } else {
            System.out.println("A pessoa com o ID especificado nÃ£o foi encontrada.");
        }
    } catch (SQLException e) {
        if (connection != null) {
            try {
                connection.rollback(); // Desfaz a transaÃ§Ã£o em caso de erro
            } catch (SQLException ex) {
            }
        }
    } finally {
        conectorBD.close(statement);
        conectorBD.close(connection);
    }
}

// MÃ©todo para buscar uma PessoaJuridica por ID
public PessoaJuridica buscarPorId(int idPessoa) {
    Connection connection = null;
    PreparedStatement statement = null;
    ResultSet resultSet = null;
    PessoaJuridica pessoaJuridica = null;

    try {
        connection = conectorBD.getConnection();
```

```
        String sql = "SELECT * FROM PessoaJuridica WHERE idPessoa = ?";
        statement = connection.prepareStatement(sql);
        statement.setInt(1, idPessoa);

        resultSet = statement.executeQuery();

        if (resultSet.next()) {
            // Obter os dados da pessoa jurÃdica do ResultSet
            int id = resultSet.getInt("idPessoa");
            String nome = resultSet.getString("nome");
            String cnpj = resultSet.getString("CNPJ");
            String logradouro = resultSet.getString("logradouro");
            String cidade = resultSet.getString("cidade");
            String estado = resultSet.getString("estado");
            String telefone = resultSet.getString("telefone");
            String email = resultSet.getString("email");

            // Outros atributos da pessoa jurÃdica...

            // Criar um objeto PessoaJuridica com os dados obtidos
            pessoaJuridica = new PessoaJuridica();
            pessoaJuridica.setId(id);
            pessoaJuridica.setNome(nome);
            pessoaJuridica.setCnpj(cnpj);
            pessoaJuridica.setLogradouro(logradouro);
            pessoaJuridica.setCidade(cidade);
            pessoaJuridica.setEstado(estado);
            pessoaJuridica.setTelefone(telefone);
            pessoaJuridica.setEmail(email);
            // Definir os outros atributos da pessoa jurÃdica no objeto
        }

    } catch (SQLException e) {
        // Tratar exceÃ§Ã£o, se necessÃ¡rio
    } finally {
        // Fechar recursos (ResultSet, PreparedStatement, Connection), se necessÃ¡rio
    }

    return pessoaJuridica;
}

// MÃ©todo para listar todas as pessoas jurÃdicas
public List<PessoaJuridica> getPessoasJuridicasPorId(int id) {
    List<PessoaJuridica> pessoasJuridicas = new ArrayList<>();
    Connection connection = null;
    PreparedStatement statementPessoaJuridica = null;
    ResultSet resultSetPessoaJuridica = null;

    try {
        connection = conectorBD.getConnection();

        // Consultar a tabela PessoaJuridica
        String sqlPessoaJuridica = "SELECT * FROM PessoaJuridica"; // Verifique se o
nome da tabela estÃ¡ correto
        statementPessoaJuridica = connection.prepareStatement(sqlPessoaJuridica);
        resultSetPessoaJuridica = statementPessoaJuridica.executeQuery();

        while (resultSetPessoaJuridica.next()) {
            PessoaJuridica pessoaJuridica = new PessoaJuridica();
            pessoaJuridica.setId(resultSetPessoaJuridica.getInt("idPessoaJuridica")); //
Verifique se o nome da coluna estÃ¡ correto
            pessoaJuridica.setNome(resultSetPessoaJuridica.getString("nome")); //
Verifique se o nome da coluna estÃ¡ correto
            pessoaJuridica.setCnpj(resultSetPessoaJuridica.getString("CNPJ")); //
Verifique se o nome da coluna estÃ¡ correto

pessoaJuridica.setLogradouro(resultSetPessoaJuridica.getString("logradouro")); //
Verifique se o nome da coluna estÃ¡ correto
            pessoaJuridica.setCidade(resultSetPessoaJuridica.getString("cidade")); //
Verifique se o nome da coluna estÃ¡ correto
            pessoaJuridica.setEstado(resultSetPessoaJuridica.getString("estado")); //
Verifique se o nome da coluna estÃ¡ correto
            pessoaJuridica.setTelefone(resultSetPessoaJuridica.getString("telefone"));
// Verifique se o nome da coluna estÃ¡ correto
            pessoaJuridica.setEmail(resultSetPessoaJuridica.getString("email"));
```

```
            // Definir outros atributos da pessoa jurÃdica, se houver

            pessoasJuridicas.add(pessoaJuridica);
        }
    } catch (SQLException e) {
        // Tratar exceÃ§Ã£o, se necessÃ¡rio
        e.printStackTrace(); // Imprime a stack trace do erro para facilitar a
depuraÃ§Ã£o
    } finally {
        conectorBD.close(resultSetPessoaJuridica);
        conectorBD.close(statementPessoaJuridica);
        conectorBD.close(connection);
    }

    return pessoasJuridicas;
}
}
```

```
/*
```

```
    Autor: Alarcon ABAP
*/

package cadastro.model;
import java.sql.ResultSet;
import java.sql.Statement;

public class SequenceManager {

    private final ConectorBD conectorBD; // DeclaraÃ§Ã£o de variÃ¡vel

    public SequenceManager() {
        this.conectorBD = new ConectorBD(); // InicializaÃ§Ã£o do objeto ConectorBD no
construtor
    }

    public int getValue(String sequenceName) throws java.sql.SQLException {
        int nextValue = 0; // VariÃ¡vel para armazenar o prÃ³ximo valor da sequÃªncia
        String sql = "SELECT nextval('" + sequenceName + "')"; // Consulta SQL para
obter o prÃ³ximo valor da sequÃªncia
        ResultSet rs = null; // DeclaraÃ§Ã£o de objeto ResultSet para armazenar o
resultado da consulta
        Statement statement = null; // DeclaraÃ§Ã£o de objeto Statement para executar a
consulta

        statement = conectorBD.getStatement(); // ObtÃ©m um objeto Statement do
ConectorBD
        rs = statement.executeQuery(sql); // Executa a consulta SQL e armazena o
resultado no ResultSet
        if (rs.next()) { // Verifica se hÃ¡ um prÃ³ximo resultado no ResultSet
            nextValue = rs.getInt(1); // ObtÃ©m o valor do primeiro coluna do resultado
e atribui Ã  variÃ¡vel nextValue
        }
        conectorBD.close(rs); // Fecha o ResultSet
        conectorBD.close(statement); // Fecha o Statement

        return nextValue; // Retorna o prÃ³ximo valor da sequÃªncia
    }
}
```

```
/*
    Autor: Alarcon ABAP
*/

package cadastrobd.model;
public class Pessoa {
    private int id;
    String nome;
    String logradouro;
    String cidade;
    String estado;
    String telefone;
    String email;

    public Pessoa() {

    }

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


 public String getEmail() {
        return email;
    }

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

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

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

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

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

```java
/*
    Autor: Alarcon ABAP
*/

package cadastrobd.model;

public class PessoaFisica extends Pessoa {
    private String cpf;
    private int id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

    public PessoaFisica() {
    }
    public PessoaFisica(int id, String nome, String cpf) {
    this.cpf = cpf;
}
    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
    }

    public String getNome() {
        return nome;
    }

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

    @Override
    public String getLogradouro() {
        return logradouro;
    }

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

    @Override
    public String getCidade() {
         return cidade;
    }

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

    @Override
    public String getEstado() {
        return estado;
    }

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

    @Override
    public String getTelefone() {
        return telefone;
    }


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

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

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

    @Override
    public String getEmail() {
        return email;
    }

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

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }
       public void setIdPessoa(int idPessoa) {
    this.id = idPessoa;
}
    @Override
        public String toString() {
            return
                   "ID: " + id + "\n" +
                   "Nome: " + nome + "\n" +
                   "CPF: " + cpf + "\n" +
                   "Logradouro: " + logradouro + "\n" +
                   "Cidade: " + cidade + "\n" +
                   "Estado: " + estado + "\n" +
                   "Telefone: " + telefone + "\n" +
                   "Email: " + email+ "\n"+
                    "-----------------------------------------";

        }




}
```

```
package cadastrobd.model;

public class PessoaJuridica extends Pessoa {
    private String cnpj;
    private int id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

    public PessoaJuridica() {
    }

    public PessoaJuridica(int id, String nome, String cnpj) {
        this.cnpj = cnpj;
    }

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

    public String getNome() {
        return nome;
    }

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

    @Override
    public String getLogradouro() {
        return logradouro;
    }

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

    @Override
    public String getCidade() {
        return cidade;
    }

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

    @Override
    public String getEstado() {
        return estado;
    }

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

    @Override
    public String getTelefone() {
        return telefone;
    }

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

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

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

    @Override
    public String getEmail() {
        return email;
    }

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

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }
    public void setIdPessoa(int idPessoa) {
        this.id = idPessoa;
    }

    @Override
    public String toString() {
        return "ID: " + id + "\n" +
               "Nome: " + nome + "\n" +
               "CNPJ: " + cnpj + "\n" +
               "Logradouro: " + logradouro + "\n" +
               "Cidade: " + cidade + "\n" +
               "Estado: " + estado + "\n" +
               "Telefone: " + telefone + "\n" +
               "Email: " + email + "\n" +
               "----------------------------------------";
    }
}
```

```
/*

Author: Alarcon ABAP
*/
package cadastrointerface;


import java.util.List;

import cadastro.model.PessoaFisicaDAO;
import cadastro.model.PessoaJuridicaDAO;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import java.util.InputMismatchException;
import java.util.Scanner;

public class InterfaceCadastro {

    // Exibe as opÃ§Ãµes do programa
    public static void exibirOpcoes() {
        System.out.println("\nOpcoes do programa:");
        System.out.println("============================");
        System.out.println("1 - Adicionar Pessoa");
        System.out.println("2 - Alterar Pessoa");
        System.out.println("3 - Excluir Pessoa");
        System.out.println("4 - Buscar pelo Id");
        System.out.println("5 - Exibir Todos");
        System.out.println("0 - Finalizar Programa");
        System.out.println("============================");
    }

     // LÃª a opÃ§Ã£o escolhida pelo usuÃ¡rio
    public static int lerOpcao(Scanner scanner) {
        int opcao = -1;
        boolean opcaoValida = false;
        while (!opcaoValida) {
            try {
                System.out.print("\nEscolha uma opcao: ");
                opcao = scanner.nextInt();
                opcaoValida = true;
            } catch (InputMismatchException e) {
                System.out.println("Opcao invalida. Digite um numero valido.");
                scanner.nextLine(); // Limpar o buffer do scanner
            }
        }
        return opcao;
    }

    // Solicita ao usuÃ¡rio que selecione o tipo (Pessoa FÃsica ou JurÃdica)
    public static String selecionarTipo(Scanner scanner) {
        String tipo = "";
        boolean tipoValido = false;
        while (!tipoValido) {
            System.out.print("\nF - Pessoa Fisica | J - Pessoa Juridica ");
            tipo = scanner.next();
            if (tipo.equalsIgnoreCase("F") || tipo.equalsIgnoreCase("J")) {
                tipoValido = true;
            } else {
                System.out.println("Tipo invalido. Digite F ou J.");
            }
        }
        return tipo;
    }
        public static void realizarInclusao(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        String tipoInclusao = selecionarTipo(scanner);
        if (tipoInclusao.equalsIgnoreCase("F")) {
            incluirPessoaFisica(scanner, pessoaFisicaDAO);
        } else if (tipoInclusao.equalsIgnoreCase("J")) {
            incluirPessoaJuridica(scanner, pessoaJuridicaDAO);
        }
    }
```

```
    // MÃ©todo para realizar a alteraÃ§Ã£o de uma pessoa fÃsica ou jurÃdica
    public static void realizarAlteracao(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        String tipoAlteracao = selecionarTipo(scanner);
        if (tipoAlteracao.equalsIgnoreCase("F")) {
            alterarPessoaFisica(scanner, pessoaFisicaDAO);
        } else if (tipoAlteracao.equalsIgnoreCase("J")) {
            alterarPessoaJuridica(scanner, pessoaJuridicaDAO);
        }
    }

   // MÃ©todo para realizar a exclusÃ£o de uma pessoa fÃsica ou jurÃdica
    public static void realizarExclusao(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        String tipoExclusao = selecionarTipo(scanner);
        if (tipoExclusao.equalsIgnoreCase("F")) {
        int idPessoa = selecionarIdPessoa(scanner);
        pessoaFisicaDAO.excluir(idPessoa);
        System.out.println("Pessoa Fisica Excluida com Sucesso.");

        // ApÃ³s a exclusÃ£o, listar novamente as pessoas fÃsicas atualizadas
        realizarListagem(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
    } else if (tipoExclusao.equalsIgnoreCase("J")) {
        int idPessoa = selecionarIdPessoa(scanner);
        pessoaJuridicaDAO.excluir(idPessoa);
        System.out.println("Pessoa Juridica Excluida com Sucesso.");

        // ApÃ³s a exclusÃ£o, listar novamente as pessoas jurÃdicas atualizadas
        realizarListagem(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
    }
}


    // MÃ©todo para obter os dados de uma pessoa fÃsica ou jurÃdica por ID
    public static void realizarObtencaoPorID(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
        String tipoObtencao = selecionarTipo(scanner);
        if (tipoObtencao.equalsIgnoreCase("F")) {
            exibirPessoaFisicaPorID(scanner, pessoaFisicaDAO);
        } else if (tipoObtencao.equalsIgnoreCase("J")) {
            exibirPessoaJuridicaPorID(scanner, pessoaJuridicaDAO);
        }
    }

    // MÃ©todo para listar todas as pessoas fÃsicas cadastradas no banco de dados
    public static void realizarListagem(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO) {
    String tipoListagem = selecionarTipo(scanner);
    if (tipoListagem.equalsIgnoreCase("F")) {
        List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.listarTodasPessoasFisicas();
        System.out.println("""
                           Dados de Pessoa Fisica...
                           ----------------------------------------""");
        for (PessoaFisica pessoaFisica : pessoasFisicas) {
            System.out.println(pessoaFisica.toString());
        }
    } else if (tipoListagem.equalsIgnoreCase("J")) {
        List<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.listarTodasPessoasJuridicas();
        System.out.println("""
                           Dados de Pessoa Juridica...
                           ----------------------------------------""");
        for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
            System.out.println(pessoaJuridica.toString());
        }
    }
}



     // LÃª os dados da pessoa fÃsica e realiza a inclusÃ£o no banco de dados
    public static void incluirPessoaFisica(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO) {
        PessoaFisica pessoaFisica = new PessoaFisica();
        scanner.nextLine(); // Limpar o buffer do scanner
```

```
        System.out.print("Digite o nome da pessoa fisica: ");
        String nome = scanner.nextLine();
        pessoaFisica.setNome(nome);

        System.out.print("Digite o CPF da pessoa fisica: ");
        String cpf = scanner.nextLine();
        pessoaFisica.setCpf(cpf);

        System.out.print("Digite o Logradouro da pessoa fisica: ");
        String logradouro = scanner.nextLine();
        pessoaFisica.setLogradouro(logradouro);

        System.out.print("Digite o cidade da pessoa fisica: ");
        String cidade = scanner.nextLine();
        pessoaFisica.setCidade(cidade);

        System.out.print("Digite o estado da pessoa fisica: ");
        String estado = scanner.nextLine();
        pessoaFisica.setEstado(estado);

        System.out.print("Digite o telefone da pessoa fisica: ");
        String telefone = scanner.nextLine();
        pessoaFisica.setTelefone(telefone);

        System.out.print("Digite o email da pessoa fisica: ");
        String email = scanner.nextLine();
        pessoaFisica.setEmail(email);

        pessoaFisicaDAO.incluir(pessoaFisica);
        System.out.println("Pessoa fisica incluida com sucesso.");
    }

     // LÃª os dados da pessoa jurÃdica e realiza a inclusÃ£o no banco de dados
    public static void incluirPessoaJuridica(Scanner scanner, PessoaJuridicaDAO
pessoaJuridicaDAO) {
        PessoaJuridica pessoaJuridica = new PessoaJuridica();
        scanner.nextLine(); // Limpar o buffer do scanner

        System.out.print("Digite o nome da pessoa jurÃdica: ");
        String nome = scanner.nextLine();
        pessoaJuridica.setNome(nome);

        System.out.print("Digite o CNPJ da pessoa jurÃdica: ");
        String cnpj = scanner.nextLine();
        pessoaJuridica.setCnpj(cnpj);

        System.out.print("Digite o Logradouro da pessoa jurÃdica: ");
        String logradouro = scanner.nextLine();
        pessoaJuridica.setLogradouro(logradouro);

        System.out.print("Digite a cidade da pessoa jurÃdica: ");
        String cidade = scanner.nextLine();
        pessoaJuridica.setCidade(cidade);

        System.out.print("Digite o estado da pessoa jurÃdica: ");
        String estado = scanner.nextLine();
        pessoaJuridica.setEstado(estado);

        System.out.print("Digite o telefone da pessoa jurÃdica: ");
        String telefone = scanner.nextLine();
        pessoaJuridica.setTelefone(telefone);

        System.out.print("Digite o email da pessoa jurÃdica: ");
        String email = scanner.nextLine();
        pessoaJuridica.setEmail(email);

        pessoaJuridicaDAO.incluir(pessoaJuridica);
        System.out.println("Pessoa jurÃdica incluÃda com sucesso.");
    }


    // ObtÃ©m o ID da pessoa fÃsica a ser alterada, lÃª os novos dados e realiza a
    //alteraÃ§Ã£o no banco de dados
```

```
    public static void alterarPessoaFisica(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO) {
        System.out.print("Digite o ID da pessoa fÃsica a ser alterada: ");
        int id = scanner.nextInt();

        PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);
        if (pessoaFisica != null) {
            System.out.println("Pessoa fÃsica encontrada:");
            System.out.println(pessoaFisica);

            scanner.nextLine(); // Limpar o buffer do scanner

            System.out.print("Digite o nome da pessoa fÃsica: ");
            String nome = scanner.nextLine();
            pessoaFisica.setNome(nome);

            System.out.print("Digite o CPF da pessoa fÃsica: ");
            String cpf = scanner.nextLine();
            pessoaFisica.setCpf(cpf);

            System.out.print("Digite o Logradouro da pessoa fÃsica: ");
            String logradouro = scanner.nextLine();
            pessoaFisica.setLogradouro(logradouro);

            System.out.print("Digite o cidade da pessoa fÃsica: ");
            String cidade = scanner.nextLine();
            pessoaFisica.setCidade(cidade);

            System.out.print("Digite o estado da pessoa fÃsica: ");
            String estado = scanner.nextLine();
            pessoaFisica.setEstado(estado);

            System.out.print("Digite o telefone da pessoa fÃsica: ");
            String telefone = scanner.nextLine();
            pessoaFisica.setTelefone(telefone);

            System.out.print("Digite o email da pessoa fÃsica: ");
            String email = scanner.nextLine();
            pessoaFisica.setEmail(email);

            pessoaFisicaDAO.incluir(pessoaFisica);
            System.out.println("Pessoa fÃsica incluÃda com sucesso.");

            pessoaFisicaDAO.alterar(pessoaFisica);
            System.out.println("Pessoa fÃsica alterada com sucesso.");
        } else {
            System.out.println("Pessoa fÃsica nÃ£o encontrada com o ID informado.");
        }
    }

    // Lista todas as pessoas jurÃdicas cadastradas no banco de dados
    public static void listarPessoasJuridicas(PessoaJuridicaDAO pessoaJuridicaDAO) {
    List<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.listarTodasPessoasJuridicas();
    System.out.println("""
                       Exibindo dados de Pessoa Juridica...
                       ------------------------------------------""");
    for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
        System.out.println(pessoaJuridica.toString());
    }
}

    // Lista todas as pessoas fÃsicas cadastradas no banco de dados
public static void listarPessoasFisicas(PessoaFisicaDAO pessoaFisicaDAO) {
    List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.listarTodasPessoasFisicas();
    if (pessoasFisicas.isEmpty()) {
        System.out.println("NÃ£o hÃ¡ pessoas fÃsicas cadastradas.");
    } else {
        System.out.println("""Exibindo dados de Pessoa Fisica...
                           ------------------------------------------""");
        for (PessoaFisica pessoaFisica : pessoasFisicas) {
            System.out.println(pessoaFisica.toString());
        }
    }
}
```

```
    public static void exibirPessoaFisicaPorID(Scanner scanner, PessoaFisicaDAO
pessoaFisicaDAO) {
        System.out.println("Digite o ID da pessoa FÃsica a ser exibida:");
        int idPessoa = scanner.nextInt();
        scanner.nextLine(); // Limpar o buffer do scanner

        PessoaFisica pessoaFisica = pessoaFisicaDAO.buscarPorId(idPessoa);
        if (pessoaFisica != null) {
            System.out.println(pessoaFisica); // Exibir os dados da pessoa fÃsica
        } else {
            System.out.println("Pessoa FÃsica nÃ£o encontrada com o ID informado.");
        }
    }

    // Exibe os dados de uma pessoa jurÃdica com base no ID fornecido
    public static void exibirPessoaJuridicaPorID(Scanner scanner, PessoaJuridicaDAO
pessoaJuridicaDAO) {
        System.out.println("Digite o ID da pessoa JurÃdica a ser exibida:");
        int idPessoa = scanner.nextInt();
        scanner.nextLine(); // Limpar o buffer do scanner

        PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.buscarPorId(idPessoa);
        if (pessoaJuridica != null) {
            System.out.println(pessoaJuridica); // Exibir os dados da pessoa jurÃdica
        } else {
            System.out.println("Pessoa JurÃdica nÃ£o encontrada com o ID informado.");
        }
    }


    public static void alterarPessoaJuridica(Scanner scanner, PessoaJuridicaDAO
pessoaJuridicaDAO) {
        System.out.print("Digite o ID da pessoa jurÃdica a ser alterada: ");
        int id = scanner.nextInt();

        PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);
        if (pessoaJuridica != null) {
            System.out.println("Pessoa jurÃdica encontrada:");
            System.out.println(pessoaJuridica);

            scanner.nextLine(); // Limpar o buffer do scanner

            System.out.print("Digite o nome da pessoa Juridica: ");
            String nome = scanner.nextLine();
            pessoaJuridica.setNome(nome);

            System.out.print("Digite o CNPJ da pessoa Juridica: ");
            String cnpj = scanner.nextLine();
            pessoaJuridica.setCnpj(cnpj);

            System.out.print("Digite o Logradouro da pessoa Juridica: ");
            String logradouro = scanner.nextLine();
            pessoaJuridica.setLogradouro(logradouro);

            System.out.print("Digite o cidade da pessoa Juridica: ");
            String cidade = scanner.nextLine();
            pessoaJuridica.setCidade(cidade);

            System.out.print("Digite o estado da pessoa Juridica: ");
            String estado = scanner.nextLine();
            pessoaJuridica.setEstado(estado);

            System.out.print("Digite o telefone da pessoa Juridica: ");
            String telefone = scanner.nextLine();
            pessoaJuridica.setTelefone(telefone);

            System.out.print("Digite o email da pessoa Juridica: ");
            String email = scanner.nextLine();
            pessoaJuridica.setEmail(email);

            pessoaJuridicaDAO.alterar(pessoaJuridica);
            System.out.println("Pessoa jurÃdica alterada com sucesso.");
        } else {
            System.out.println("Pessoa jurÃdica nÃ£o encontrada com o ID informado.");
```

```
        }
    }

        // Solicita ao usuÃ¡rio que digite o ID de uma pessoa fÃsica
        // a ser excluÃda e retorna o ID fornecido
        public static int selecionarIdPessoa(Scanner scanner) {
        System.out.println("Digite o ID da pessoa fÃsica a ser excluÃda:");
        int idPessoa = scanner.nextInt();
        scanner.nextLine(); // Limpar o buffer do scanner

        return idPessoa;
    }
}
```