



Backend sem banco não tem

Inserir aqui seu Nome Completo e Matrícula

Jequié-BA

Backend sem banco não tem – 2023.1 – 2024.1

Objetivo da Prática

O objetivo desta missão prática foi realizar um simples sistema de cadastro usando um banco de dados (Microsoft SQL Server) e Java usando o JDBC (Java Database Connectivity) usando o modelo DAO (Data Access Object)

Link do repositório: <https://github.com/IsraelHamdan/CadastroDB.git>

1º Procedimento | Mapeamento Objeto-Relacional e DAO

- **Classe Pessoa:** Define todos os atributos bases da pessoa, e se relaciona diretamente com a tabela pessoa fazendo com que ela tenha sua tabela exclusiva no banco de dados

```
1  public class Pessoa {
2
3      private int id;
4      private String nome;
5      private String logradouro;
6      private String cidade;
7      private String estado;
8      private String telefone;
9      private String email;
10
11     public Pessoa() {
12
13     }
14     public Pessoa(int id, String nome, String logradouro, String cidade,
15         String estado, String telefone, String email) {
16         this.id = id;
17         this.nome = nome;
18         this.logradouro = logradouro;
19         this.cidade = cidade;
20         this.estado = estado;
21         this.telefone = telefone;
22         this.email = email;
23     }
24 }
25
26
```

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

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

public String getNome() {
    return nome;
}

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

public String getLogradouro() {
    return logradouro;
}

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

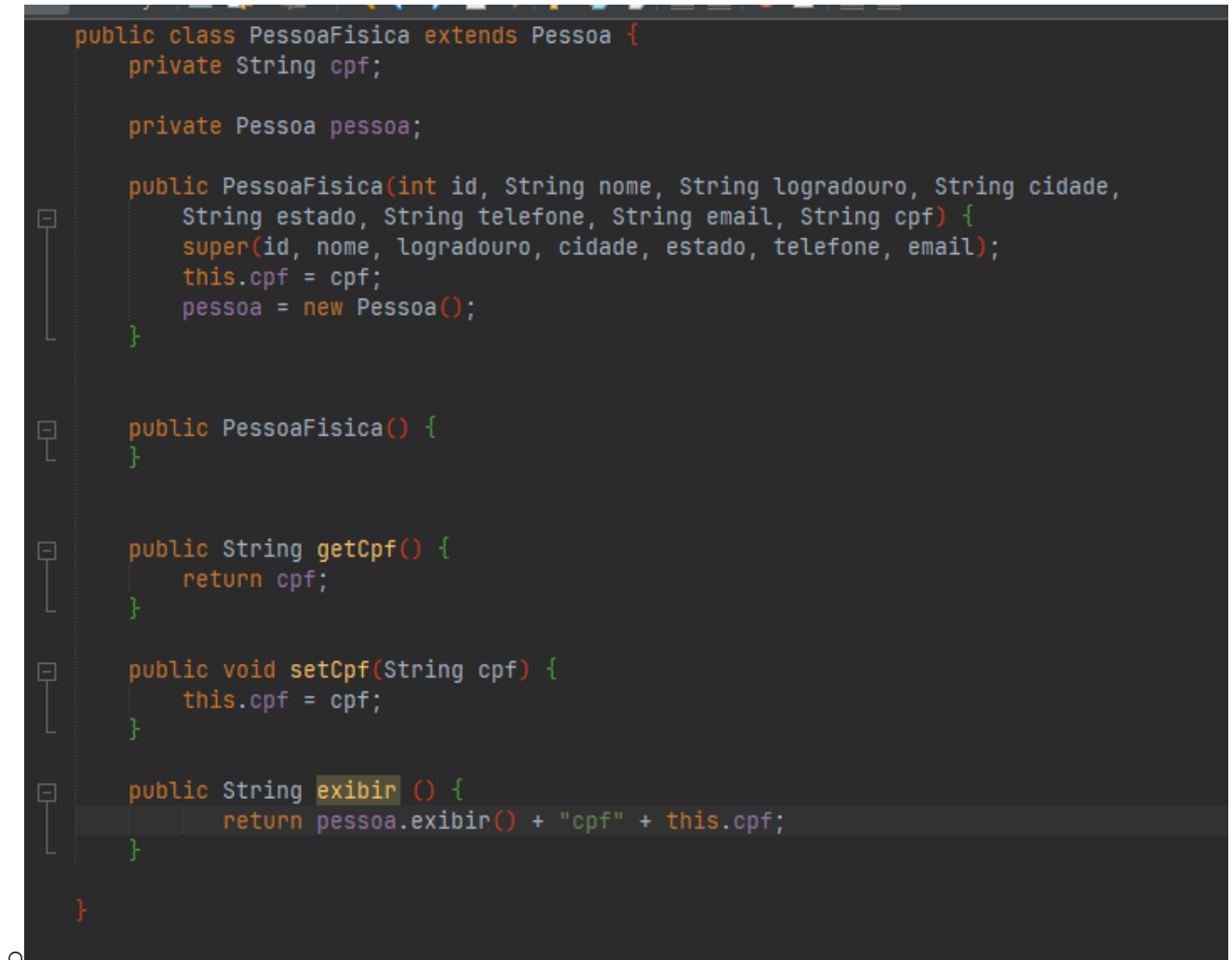
public String getCidade() {
    return cidade;
}
```

```
public String exibir() {
    return "Pessoa{" +
        "id=" + id +
        ", nome=" + nome +
        ", logradouro=" + logradouro +
        ", cidade=" + cidade +
        ", estado=" + estado +
        ", telefone=" + telefone +
        ", email=" + email +
        '}';
}

}
```

- O método `exibir`, na verdade ele é uma coisa que poderia ser substituída pelo `toString` do próprio Java, os dois fazem a mesma função, porém o `toString` é nativo da linguagem

- **Classe PessoaFisica**

A screenshot of a code editor showing the implementation of the `PessoaFisica` class. The class extends `Pessoa` and has a private attribute `cpf` and a private attribute `Pessoa` of type `Pessoa`. It has two constructors: one with multiple parameters including `cpf` and another without. It also has `getCpf()` and `setCpf()` methods. The `exibir()` method is highlighted, showing it returns the result of `Pessoa.exibir()` concatenated with `"cpf"` and `this.cpf`.

```
public class PessoaFisica extends Pessoa {
    private String cpf;

    private Pessoa pessoa;

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

    public PessoaFisica() {
    }

    public String getCpf() {
        return cpf;
    }

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

    public String exibir () {
        return pessoa.exibir() + "cpf" + this.cpf;
    }
}
```

- Por ser uma classe que é apenas uma extensão da `Pessoa` ela acaba requerendo menos atributos, já que a única característica própria da `PessoaFisica` é o CPF, então todos os parâmetros de criação da `Pessoa` são passados juntos, para que a `Pessoa` também seja criada toda vez que criarmos uma nova `PessoaFisica`, e o método `exibir`, como já falei, poderia ser substituído pelo `toString`

- **Classe PessoaJuridica:** a única característica diferente da `PessoaFisica` é o CNPJ, pois de resto, ambas são idênticas, e por isso o raciocínio de criação de ambas foi o mesmo

○

```

public class PessoaJuridica extends Pessoa {
    private String cnpj;

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

    public PessoaJuridica() {
    }

    public String getCnpj() {
        return cnpj;
    }

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

    public String exibir() {
        super.exibir();
        return "CNPJ" + this.cnpj;
    }
}

```

- **Classe ConectorDB:** esta classe é responsável por cuidar da conexão com o banco de dados, eu sei que tem uma melhor maneira de fazer essa conexão, escondendo os dados sensíveis da conexão em uma parte extra, mas não foi implementado neste projeto!

```

public class ConectorDB {
    private static final Logger LOGGER = Logger.getLogger(PessoaFisicaDAO.class.getName());
    private static final String URL = "jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true" ;
    private static final String USERNAME = "loja";
    private static final String SENHA = "loja";
    private Connection connection;
    private Properties props;

    public ConectorDB() {
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            props = new Properties();
            props.setProperty("user", USERNAME);
            props.setProperty("password", SENHA);
            props.setProperty("encrypt", "true");
            props.setProperty("trustServerCertificate", "true");
            connection = DriverManager.getConnection(URL, props);
        } catch (ClassNotFoundException | SQLException e) {
            LOGGER.log(Level.SEVERE, "erro ao conectar com o banco de dados", e);
        }
    }
}

```

```

public static Connection getConnection() throws SQLException {
    try {
        return DriverManager.getConnection(URL, USERNAME, SENHA);
    } catch (SQLException e) {
        LOGGER.log(Level.SEVERE, "error ao realizar a conexão do driver com o banco", e);
    }
    return null;
}

public PreparedStatement getPreparedStatement(String sql) throws SQLException {
    try {
        return connection.prepareStatement(sql);
    } catch (SQLException e) {
        LOGGER.log(Level.SEVERE, "erro ao usar o preparedStatement na classe conectorDB: " , e);
    }
    return null;
}

public ResultSet getSelected(String sql) throws SQLException {
    try {
        Statement statement = connection.createStatement();
        return statement.executeQuery(sql);
    } catch (SQLException e) {
        LOGGER.log(Level.SEVERE, "erro ao definir o ResultSet: ", e);
    }
    return null;
}

```

- Aqui o getConnection tenta fazer a conexão do driver do JDBC para SQL Server, usando os parametros de conexão definidos anteriormente
- O getPreparedStatement recebe uma string que é a consulta SQL que precisa ser preparada
- O getSelected, também recebe uma string de consulta SQL mas o invés de prepara-la, ele já manda direto pro JDBC chama o createStatement para criar um estado e retorna esse estado acionando uma query SQL que foi mandada como parametro

```

// fecha o PreparedStatement
public void close(PreparedStatement ps) {
    try {
        if(ps != null && !ps.isClosed()) {
            ps.close();
        }
    } catch (SQLException e) {
        LOGGER.log(Level.SEVERE, "erro ao fechar o PreparedStatement: ", e);
    }
}

// metodo close para fechar o result
public void close (ResultSet rs) {
    try {
        if(rs != null && !rs.isClosed()) rs.clearWarnings();
    } catch (SQLException e) {
        LOGGER.log(Level.SEVERE, "erro ao fechar o ResultSet: ", e);
    }
}

public void close() {
    try {
        if(connection != null && !connection.isClosed()) {
            connection.close();
        }
    } catch (SQLException e) {
        LOGGER.log(Level.SEVERE, "erro ao fechar o Connection: ", e);
    }
}

```

▪Aqui todas as conexões são fechadas

- **Classe SequenceManager:** Sua única responsabilidade é pegar o valor do Sequence que é usado para definir o valor do id do usuário

```

public class SequenceManager {

    public SequenceManager() {
    }

    public int getValue(String sequenceName) throws SQLException {
        ResultSet rs = null;
        rs = new ConectorDB().getSelected("SELECT NEXT VALUE FOR " + sequenceName);
        if(rs.next()) {
            return rs.getInt(1);
        } else {
            throw new SQLException("erro: Não tem valor de sequencia" + sequenceName);
        }
    }
}

```

- **Classe PessoaFisicaDAO:** Sua responsabilidade é construir os objetos de acesso aos atributos, fazendo consultas no SQL através do PreparedStatement

```

public class PessoaFisicaDAO {
    private PessoaFisica pf;

    private ResultSet rs;
    private final ConectorDB connector;

    private List<PessoaFisica> pessoas;
    private String error;

    public PessoaFisicaDAO() {
        connector = new ConectorDB();
        pf = new PessoaFisica();
        pessoas = new ArrayList<>();
    }

    private static final Logger LOGGER = Logger.getLogger(PessoaFisicaDAO.class.getName());

    private String errorMessage(String verbo, String especificador) {
        String message = "Erro ao %s : " + verbo + "pessoas fisicas %s banco" + especificador;
        return message;
    }
}

```

- Iniciando a classe instanciamos algumas classes dentro do construtor encapsulando eles, além de que logo no inicio, é implementado o Logger para o tratamento de erros, o metodo errorMessage foi explicado posteriormente

```

public PessoaFisica getPessoa(int id) throws SQLException {
    String query = "SELECT PF.idPessoaFisica, PF.CPF, P.Nome, P.Logradouro, P.Cidade, P.Estado, P.Telefone, P.email " +
        "From PessoasFisicas PF JOIN Pessoas P ON PF.idPessoaFisica = P.idPessoa WHERE P.idPessoa = ?";
    try (PreparedStatement ps = connector.getConnection()
        .prepareStatement(query)) {
        ps.setInt(1, id);
        rs = ps.executeQuery();
        if (rs.next()) {
            pf.setNome(rs.getString("Nome"));
            pf.setLogradouro(rs.getString("Logradouro"));
            pf.setId(rs.getInt("idPessoaFisica"));
            pf.setCpf(rs.getString("cpf"));
            pf.setCidade(rs.getString("Logradouro"));
            pf.setEstado(rs.getString("Estado"));
            pf.setTelefone(rs.getString("Telefone"));
            pf.setEmail(rs.getString("Email"));

            return pf;
        }
    } catch (SQLException e) {
        error = errorMessage("buscar", "no");
        LOGGER.log(Level.SEVERE, error, e );
    } finally {
        connector.close();
    }
    return null;
}

```

- Aqui, o método getPessoa, busca uma pessoaFisica pelo id, e monta uma estrutura de categorização os dados sempre que houver um proximo na lista e faz o tratamento de exceções usando o errorMessage e o Logger
- Mas este metodo só serve para uso interno do sistema pois quando ele retorna uma pessoa fisica, ele printa um objeto de memória, por isso eu criei um metodo adicional que pega a pessoa que ele encontrou e monta uma exibição dos dados

```

public void exibirPessoaFisica(int idPessoaFisica) throws SQLException {
    try {
        if(idPessoaFisica != 0) {
            PessoaFisica pf = getPessoa(idPessoaFisica);
            if(pf != null) {
                System.out.println("CPF: " + pf.getCpf());
                System.out.println("Nome: " + pf.getNome());
                System.out.println("Logradouro: " + pf.getLogradouro());
                System.out.println("Cidade: " + pf.getCidade());
                System.out.println("Estado: " + pf.getEstado());
                System.out.println("Telefone " + pf.getTelefone());
                System.out.println("Email: " + pf.getEmail());
            } else {
                System.out.println("Pessoa não encontrada");
            }
        }
    } catch (SQLException e) {
        error = "Erro %s ao exibir pessoas encontradas no banco";
        LOGGER.log(Level.SEVERE, error, e);
    }
}

```

○Método getPessoas, retorna uma lista de pessoas físicas, mas infelizmente, também printa somente um endereço de memória, então foi criado um método auxiliar que exibe de forma legível ao usuário, as pessoas da tabela de pessoas físicas da tabela de pessoas, chamado exibirPessoasFisicas

```

public List<PessoaFisica> getPessoas() throws SQLException {
    String query = "SELECT PF.idPessoaFisica, PF.CPF, P.Nome, P.Logradouro, P.Cidade, P.Estado, P.Telefone, P.email " +
        "From PessoasFisicas PF JOIN Pessoas P ON PF.idPessoaFisica = P.idPessoa";
    try (PreparedStatement ps = connector.getConnection().prepareStatement(query)) {
        rs = ps.executeQuery();
        while (rs.next()) {
            pf = new PessoaFisica();
            pf.setId(rs.getInt("idPessoaFisica"));
            pf.setNome(rs.getString("Nome"));
            pf.setCpf(rs.getString("CPF"));
            pf.setCidade(rs.getString("Cidade"));
            pf.setLogradouro(rs.getString("Logradouro"));
            pf.setEstado(rs.getString("Estado"));
            pf.setTelefone(rs.getString("Telefone"));
            pf.setEmail(rs.getString("Email"));
            pessoas.add(pf);
        }
    } catch (SQLException e) {
        error = errorMessage("listar", "do");
        LOGGER.log(Level.SEVERE, error, e);
    } finally {
        connector.close();
    }
    return pessoas;
}

```



```

public void exibirPessoasFisicas() throws SQLException{
    try {
        List<PessoaFisica> pessoas = getPessoas();
        if(!pessoas.isEmpty()) {
            for (PessoaFisica pessoa : pessoas) {
                System.out.println("ID: " + pessoa.getId());
                System.out.println("Nome: " + pessoa.getNome());
                System.out.println("Logradouro: " + pessoa.getLogradouro());
                System.out.println("Cidade: " + pessoa.getCidade());
                System.out.println("Estado: " + pessoa.getEstado());
                System.out.println("Telefone: " + pessoa.getTelefone());
                System.out.println("Email: " + pessoa.getEmail());
                System.out.println("-----");
            }
        } else {
            System.out.println("Id não encontrado!");
        }
    } catch (SQLException e) {
        error = errorMessage("exibir","do");
        LOGGER.log(Level.SEVERE, error, e);
    }
}

```

o Método **incluiPessoa**: Faz a inclusão de uma única pessoa no banco de dados, passando todos os parametros que o usuário precisa digitar

```

public void incluiPessoa(PessoaFisica pessoa) throws SQLException {
    String queryPessoa = "INSERT INTO Pessoas(idPessoa, Nome, Logradouro, Cidade, Estado, Telefone, Email) VALUES(?,?,?,?,?,?,?)";
    String queryPf = "INSERT INTO PessoasFisicas(idPessoaFisica, cpf) VALUES(?,?)";

    try (PreparedStatement psPessoa = connector.getConnection().prepareStatement(queryPessoa);
        PreparedStatement psPessoaFisica = connector.getConnection().prepareStatement(queryPf)) {

        psPessoa.setInt(1, pessoa.getId());
        psPessoa.setString(2, pessoa.getNome());
        psPessoa.setString(3, pessoa.getLogradouro());
        psPessoa.setString(4, pessoa.getCidade());
        psPessoa.setString(5, pessoa.getEstado());
        psPessoa.setString(6, pessoa.getTelefone());
        psPessoa.setString(7, pessoa.getEmail());
        psPessoa.executeUpdate();

        psPessoaFisica.setInt(1, pessoa.getId());
        psPessoaFisica.setString(2, pessoa.getCpf());
        psPessoaFisica.executeUpdate();
    } catch (SQLException e) {
        error = errorMessage("incluir", "no");
        LOGGER.log(Level.SEVERE, error, e);
    } finally {
        connector.close();
    }
}

```

o Método **alterar**: faz a alterar se uma pessoa fsisca de dentro do banco de dados

```

public void alterarPessoa(int idPessoaFisica, PessoaFisica pessoa) throws SQLException {
    String query = "UPDATE Pessoas SET Nome = ?, Logradouro = ?, Cidade = ?, Estado = ?, Telefone = ?, Email = ?"
        + "WHERE idPessoa = ?";

    try (PreparedStatement ps = connector.getConnection().prepareStatement(query)) {
        ps.setString(1, pessoa.getNome());
        ps.setString(2, pessoa.getLogradouro());
        ps.setString(3, pessoa.getCidade());
        ps.setString(4, pessoa.getEstado());
        ps.setString(5, pessoa.getTelefone());
        ps.setString(6, pessoa.getEmail());
        ps.setInt(7, pessoa.getId());
        ps.executeUpdate();
    } catch (SQLException e) {
        error = errorMessage("alterar", "no");
        LOGGER.log(Level.SEVERE, error, e);
    }
}

```

- Por regra de negócio, o usuário não pode alterar seu id e seu CPF

○ Método excluir: Exclui uma pessoa física do banco de dados,

```
public void excluirPessoa(int id) throws SQLException {
    String queryDeletePf = "DELETE FROM PessoasFisicas WHERE idPessoaFisica = ?";
    String queryDeletePs = "DELETE FROM Pessoas WHERE idPessoa = ?";
    try(PreparedStatement ps = connector.getConnection().prepareStatement(queryDeletePf);
        PreparedStatement pfDelete = connector.getConnection().prepareStatement(queryDeletePs)) {

        //excluindo da tabela de pessoas
        ps.setInt(1, id);
        ps.executeUpdate();

        //excluindo da tabela de pessoas fisicas;
        pfDelete.setInt(1, id);
        pfDelete.executeUpdate();
    } catch (SQLException e) {
        error = errorMessage("excluir", "do");
        LOGGER.log(Level.SEVERE, error, e);
    } finally {
        connector.close();
    }
}
```

- Classe PessoaJuridicaDAO: foi usado a mesma metodologia de implementação da classe PessoaFisicaDAO, só mudou apenas algumas coisas, pra ficar específico para as pessoas jurídicas

```
public class PessoaJuridicaDAO {
    private static final Logger LOGGER = Logger.getLogger(PessoaFisicaDAO.class.getName());
    private PessoaJuridica pj;

    private ResultSet rs;
    private ConectorDB connector;
    private SequenceManager sequence;

    private List<PessoaJuridica> pessoas;
    private String error;

    public PessoaJuridicaDAO() {
        sequence = new SequenceManager();
        connector = new ConectorDB();
        pj = new PessoaJuridica();
        pessoas = new ArrayList<>();
    }

    private String errorMessage(String verbo, String especificador) {
        String message = "Erro ao %s : " + verbo + "pessoas fisicas %s banco" + especificador;
        return message;
    }
}
```

```

public PessoaJuridica getPessoa(int id) throws SQLException {
    String query = "Select PJ.idPJ, PJ.CNPJ, P.Nome, P.Logradouro, P.Cidade, P.Estado, P.Telefone, P.Email " +
        "From PessoasJuridicas PJ JOIN Pessoas P ON PJ.idPJ = P.idPessoa WHERE idPJ = ?";
    try (PreparedStatement ps = connector.getConnection()
        .prepareStatement(query)) {
        ps.setInt(1, id);
        rs = ps.executeQuery();
        if (rs.next()) {
            pj.setCnpj(rs.getString("cnpj"));
            pj.setNome(rs.getString("Nome"));
            pj.setLogradouro(rs.getString("Logradouro"));
            pj.setCidade(rs.getString("Cidade"));
            pj.setTelefone(rs.getString("Estado"));
            pj.setEmail(rs.getString("Email"));
            return pj;
        }
    } catch (SQLException e) {
        error = errorMessage("buscar", "no");
        LOGGER.log(Level.SEVERE, error, e );
    } finally {
        connector.close();
    }
    return null;
}

```

- Método auxiliar para exibir a pessoa jurídica

```

public void exibirPessoaJuridica(int idPJ) throws SQLException {
    if(idPJ != 0) {
        PessoaJuridica pj = getPessoa(idPJ);
        System.out.println("CNPJ" + pj.getCnpj());
        System.out.println("Nome: " + pj.getNome());
        System.out.println("Logradouro: " + pj.getLogradouro());
        System.out.println("Cidade: " + pj.getCidade());
        System.out.println("Estado: " + pj.getEstado());
        System.out.println("Telefone " + pj.getTelefone());
        System.out.println("Email: " + pj.getEmail());
    } else{
        System.out.println("Pessoa não encontrada!");
    }
}

```

```

public List<PessoaJuridica> getPessoas() throws SQLException {
    String query = "Select PJ.idPJ, PJ.CNPJ, P.Nome, P.Logradouro, P.Cidade, P.Estado, P.Telefone, P.Email " +
        "From PessoasJuridicas PJ JOIN Pessoas P ON PJ.idPJ = P.idPessoa";

    try (PreparedStatement ps = connector.getConnection().prepareStatement(query)) {
        rs = ps.executeQuery();
        while(rs.next()) {
            pj = new PessoaJuridica();
            pj.setId(rs.getInt("idPj"));
            pj.setNome(rs.getString("Nome"));
            pj.setCnpj(rs.getString("CNPJ"));
            pj.setCidade(rs.getString("Cidade"));
            pj.setLogradouro(rs.getString("Logradouro"));
            pj.setEstado(rs.getString("Estado"));
            pj.setTelefone(rs.getString("Telefone"));
            pj.setEmail(rs.getString("Email"));
            pessoas.add(pj);
        }
    } catch (SQLException e) {
        error = errorMessage("buscar todas as pessoas", "do");
    } finally {
        connector.close();
    }
    return pessoas;
}

```

- Método auxiliar para imprimir a lista de pessoas jurídicas

```

public void exibirPessoasJuridicas() throws SQLException {
    List<PessoaJuridica> pessoas = getPessoas();
    for (PessoaJuridica pessoa : pessoas) {
        System.out.println("ID: " + pessoa.getId());
        System.out.println("Nome: " + pessoa.getNome());
        System.out.println("Logradouro: " + pessoa.getLogradouro());
        System.out.println("Cidade: " + pessoa.getCidade());
        System.out.println("Estado: " + pessoa.getEstado());
        System.out.println("Telefone: " + pessoa.getTelefone());
        System.out.println("Email: " + pessoa.getEmail());
        System.out.println("-----");
    }
}

```

```

public void incluirPessoa(PessoaJuridica pessoa) throws SQLException {
    String queryPessoa = "INSERT INTO Pessoas(idPessoa, Nome, Logradouro, Cidade, Estado, Telefone, Email) VALUES(?, ?, ?, ?, ?, ?, ?)";
    String queryCnpj = "INSERT INTO PessoasJuridicas(idPJ, CNPJ) VALUES(?, ?)";

    try (PreparedStatement psPessoa = connector.getConnection().prepareStatement(queryPessoa);
        PreparedStatement psPj = connector.getConnection().prepareStatement(queryCnpj)) {

        psPessoa.setInt(1, pessoa.getId());
        psPessoa.setString(2, pessoa.getNome());
        psPessoa.setString(3, pessoa.getLogradouro());
        psPessoa.setString(4, pessoa.getCidade());
        psPessoa.setString(5, pessoa.getEstado());
        psPessoa.setString(6, pessoa.getTelefone());
        psPessoa.setString(7, pessoa.getEmail());
        psPessoa.executeUpdate();

        psPj.setInt(1, pessoa.getId());
        psPj.setString(2, pessoa.getCnpj());
        psPj.executeUpdate();
    } catch (SQLException e) {
        error = errorMessage("inserir", "no");
    } finally {
        connector.close();
    }
}

```

```

public void alterarPessoa(int idPJ, PessoaJuridica pessoa) throws SQLException {
    String query = "UPDATE Pessoas SET Nome = ?, Logradouro = ?, Cidade = ?, Estado = ?, Telefone = ?, Email = ?"
        + "WHERE idPessoa = ?";

    try (PreparedStatement ps = connector.getConnection().prepareStatement(query)) {
        ps.setString(1, pessoa.getNome());
        ps.setString(2, pessoa.getLogradouro());
        ps.setString(3, pessoa.getCidade());
        ps.setString(4, pessoa.getEstado());
        ps.setString(5, pessoa.getTelefone());
        ps.setString(6, pessoa.getEmail());
        ps.setInt(7, pessoa.getId());
        ps.executeUpdate();
    } catch (SQLException e) {
        error = errorMessage("alterar", "no");
        LOGGER.log(Level.SEVERE, error, e);
    }
}

```

```

public void excluirPessoa(int id) throws SQLException {
    String queryDeletePj = "DELETE FROM PessoasJuridicas WHERE idPj = ?";
    String queryDeleteP = "DELETE FROM Pessoas WHERE idPessoa = ?";

    try (PreparedStatement ps = connector.getConnection().prepareStatement(queryDeletePj);
        PreparedStatement pjDelete = connector.getConnection().prepareStatement(queryDeleteP)) {

        //excluindo da tabela de pessoas
        ps.setInt(1, id);
        ps.executeUpdate();

        //excluindo da tabela de pessoas juridicas
        pjDelete.setInt(1, id);
        pjDelete.executeUpdate();
    } catch (SQLException e) {
        error = errorMessage("excluir", "do");
        LOGGER.log(Level.SEVERE, error, e);
    } finally {
        connector.close();
    }
}

```

- **Classe CadastroDBTeste:** esta classe o usuário não interage com ela, é somente para testar a aplicação antes de começar a construir a interface do terminal, para cada teste são necessários colocar novos valores para o banco de dados

```
public class CadastroDBTeste {

    PessoaFisica pf;
    PessoaJuridica pj;

    private PessoaFisicaDAO pfDAO;
    private PessoaJuridicaDAO pjDAO;
    private SequenceManager seq;

    public CadastroDBTeste() {
        pfDAO = new PessoaFisicaDAO();
        seq = new SequenceManager();
        pjDAO = new PessoaJuridicaDAO();
        pf = new PessoaFisica();
        pj = new PessoaJuridica();
    }

    private void testeDeInclusao () throws SQLException{
        String sequencia = "seq_pessoa";
        int seqM = seq.getValue(sequencia);

        pf = new PessoaFisica(seqM, "Natalia", "condominio", "Maracaes", "BA", "73987654321", "Natalia@gmail.com",
            "41258099500");
        pfDAO.incluiPessoa(pf);
        pfDAO.exibirPessoaFisica(seqM);
    }
}
```

```
private void testeDeInclusaoDaPessoaJuridica() throws SQLException {
    String sequencia = "seq_pessoa";
    int seqM = seq.getValue(sequencia);

    pj = new PessoaJuridica(seqM, "Bar do João ", "Avenida", "Jequié", "BA", "7335432100",
        "JoaoDa51@gmail.com", "62221978099");
    pjDAO.incluirPessoa(pj);
    pjDAO.exibirPessoaJuridica(seqM);
}

private void buscaPessoaFisica(PessoaFisica pf) throws SQLException {
    pfDAO.exibirPessoaFisica(pf.getId());
}

private void buscaPessoaJuridica()throws SQLException {
    for(PessoaJuridica pj : pjDAO.getPessoas()) {
        System.out.println(pj.exibir());
    }
    pjDAO.exibirPessoasJuridicas();
}
}
```

```
private void testeDeAlteracao() throws SQLException {
    pf.setNome("Lorena");
    pf.setLogradouro("Condominio");
    pf.setCidade("Maracaes");
    pf.setEstado("BA");
    pf.setTelefone("73987654321");
    pf.setEmail("lorena@gmail.com");
    pfDAO.alterarPessoa(pf.getId(), pf);
    pfDAO.exibirPessoaFisica(pf.getId());
}

private void testeDeAlteracaoDaPessoaJuridica() throws SQLException {
    pj.setNome("Bar do Agemiro");
    pj.setLogradouro("rua");
    pj.setCidade("Jequié");
    pj.setEstado("BA");
    pj.setTelefone("7335654321");
    pj.setEmail("bardoAgemiro@gmail.com");
    pjDAO.alterarPessoa(pj.getId(), pj);
}

private void testeDeExclusao() throws SQLException {
    pfDAO.excluirPessoa(pf.getId());
}
```

○

```
private void testeDeExclusao() throws SQLException {
    pfDAO.excluirPessoa(pf.getId());
}

private void testeDeExclusaoDaPessoaJuridica() throws SQLException {
    pjDAO.excluirPessoa(pj.getId());
}

public void run() throws SQLException {
    testeDeInclusao();
    testeDeAlteracao();
    buscaPessoaFisica(pf);
    testeDeExclusao();
    testeDeInclusaoDaPessoaJuridica();
    testeDeAlteracaoDaPessoaJuridica();
    buscaPessoaJuridica();
    testeDeExclusaoDaPessoaJuridica();
}

public static void main(String[] args) throws SQLException {
    new CadastroDBTeste().run();
}
}
```

- Foto do teste realizado, uma observação, como sempre excluimos a pessoa no final dos testes, então ela não fica no banco, então não tem como mostrar ela lá, até porque ela foi excluída!

1. Teste 1: Incluir e alterar uma pessoa física

```
run:
CPF: 75647250017
Nome: Brenda
Logradouro: condominio
Cidade: condominio
Estado: BA
Telefone 7335456789
Email: Brenda@gmail.com
CPF: 75647250017
Nome: Fernanda
Logradouro: Condominio
Cidade: Condominio
Estado: BA
Telefone 7335356789
Email: Fernanda@gmail.com
```

a.

b.Os dados inseridos foram, nome: Brenda, logradouro: condominio, cidade Jequié, estado: Bahia, telefone: 7335456789, email: Brenda@gmail.com

c.Depois os dados foram trocados por nome: Fernanda, logradouro cidade e estado foram mantidos, mas o telefone é 7335356789 e o email é"Fernanda@gmail.com

2.Teste 2: Incluir, alterar e excluir uma pessoa física

a.Dados inicialmente cadastrados

i.(seqM, "Fernanda", "condominio", "Jequié", "BA", "7335356789",
"Fernanda@gmail.", "26850513095"),

1.Obs: seqM é o id que é gerado automaticamente pelo banco

b.Dados alterados:

i.

```
pf.setNome("Ayme");  
pf.setLogradouro("Condominio");  
pf.setCidade("Jequié");  
pf.setEstado("BA");  
pf.setTelefone("7334567890");  
pf.setEmail("Ayme@gmail.com");  
pfDAO.alterarPessoa(pf.getId(),pf);  
pfDAO.exibirPessoaFisica(pf.getId());
```

c.Resultados

```
run:  
CPF: 26850513095  
Nome: Fernanda  
Logradouro: condominio  
Cidade: condominio  
Estado: BA  
Telefone 7335356789  
Email: Fernanda@gmail.com  
=====  
CPF: 26850513095  
Nome: Ayme  
Logradouro: Condominio  
Cidade: Condominio  
Estado: BA  
Telefone 7334567890  
Email: Ayme@gmail.com  
=====
```

3. Incluindo, alterando e buscando uma pessoa física:

a.Dados iniciais:

i.

```
pf = new PessoaFisica(seqM, "Lucia", "condominio", "Salvador", "BA", "73998765432", "Lucia@gmail.com",  
"59821873057");  
pfDAO.incluiPessoa(pf);  
pfDAO.exibirPessoaFisica(seqM);
```

b.Dados para alteração

```
private void testeDeAlteracao() throws SQLException {  
    pf.setNome("Lucia");  
    pf.setLogradouro("Condominio");  
    pf.setCidade("Salvador");  
    pf.setEstado("BA");  
    pf.setTelefone("73998765432");  
    pf.setEmail("Lucia@gmail.com");  
    pfDAO.alterarPessoa(pf.getId(), pf);  
    pfDAO.exibirPessoaFisica(pf.getId());  
}
```

c.Resultados do teste:

```
run:  
CPF: 59821873057  
Nome: Lucia  
Logradouro: condominio  
Cidade: condominio  
Estado: BA  
Telefone 73998765432  
Email: Lucia@gmail.com  
=====  
CPF: 59821873057  
Nome: Lucia  
Logradouro: Condominio  
Cidade: Condominio  
Estado: BA  
Telefone 73998765432  
Email: Lucia@gmail.com  
=====  
CPF: 59821873057  
Nome: Lucia  
Logradouro: Condominio  
Cidade: Condominio  
Estado: BA  
Telefone 73998765432  
Email: Lucia@gmail.com  
=====
```

4. Inserindo, alterando, buscando, excluindo uma pessoaFisica

```

=====Inserindo=====
CPF: 15458679687
Nome: Sérgio
Logradouro: Apto
Cidade: Apto
Estado: PR
Telefone 41987654321
Email: Sergio@gmail.com
=====Alterando=====
CPF: 15458679687
Nome: Flávio
Logradouro: Apto
Cidade: Apto
Estado: PR
Telefone 41987654321
Email: Favin@gmail.com
=====Buscando=====
CPF: 15458679687
Nome: Flávio
Logradouro: Apto
Cidade: Apto
Estado: PR
Telefone 41987654321
Email: Favin@gmail.com
=====Exluindo=====
BUILD SUCCESSFUL (total time: 2 seconds)

```

a.

5. Testando inclusão da pessoa jurídica

```

pj = new PessoaJuridica(seqM, "Bar do Iran ", "rua", "Jequié", "BA", "7335312345",
    "JoaoDa51@gmail.com", "05840379530");

```

a.

b. Resultado do teste

```

run:
=====Inserindo PJ=====
CNPJ05840379530
Nome: Bar do Iran
Logradouro: rua
Cidade: Jequié
Estado: null
Telefone BA
Email: JoaoDa51@gmail.com
BUILD SUCCESSFUL (total time: 3 seconds)

```

i.

6. Testando inclusão e alteração de uma pessoa jurídica

a. Dados para cadastrar

```

pj = new PessoaJuridica(seqM, "Bar do Falcão", "rua", "Jequié", "BA", "7335156789",
    "PIXDoBarDoFalcão@gmail.com", "55115465000");

```

i.

ii. Dados para alterar

```
private void testeDeAlteracaoDaPessoaJuridica() throws SQLException {
    pj.setNome("Bar do Jocemar");
    pj.setLogradouro("avenida");
    pj.setCidade("Jequié");
    pj.setEstado("BA");
    pj.setTelefone("7335501234");
    pj.setEmail("PIXBarDoJocemar@gmail.com");
    pjDAO.alterarPessoa(pj.getId(), pj);
}
```

iii.Resultados

| | idPJ | CNPJ | Nome | Logradouro | Cidade | Estado | Telefone | Email |
|----|------|-------------|----------------|------------|----------------|--------|-------------|---------------------------|
| 1 | 211 | 05840379530 | Bar do Iran | rua | Jequié | BA | 7335312345 | JoaoDa51@gmail.com |
| 2 | 197 | 06718410098 | Magno Express | Avenida | Jequié | BA | 7335512345 | magnoCafes@gmail.com |
| 3 | 5 | 1232151546 | Ambev | Avenida | Petropolis | RJ | 21999999999 | ambev@email.com |
| 4 | 6 | 1516541351 | Heiniken | Parque | Belo Horizonte | MG | 31999999999 | Heiniken@email.com |
| 5 | 4 | 2151321531 | Budwiser | Rua | São Luis | Ma | 11999999999 | budwiser@email.com |
| 6 | 198 | 52843617090 | Bar do Pedro | Avenida | Jequié | BA | 7335178901 | PedraDa51@gmail.com |
| 7 | 212 | 55115465000 | Bar do Jocemar | avenida | Jequié | BA | 7335501234 | PIXBarDoJocemar@gmail.com |
| 8 | 7 | 5531546535 | Corona | Parque | Belo Horizonte | MG | 31999999999 | Corona@email.com |
| 9 | 8 | 58796413561 | Coca-cola | Fabrica | Uberlandia | MG | 31336541325 | cocacolabr@gmail.com |
| 10 | 199 | 62221978099 | Bar do Agemiro | rua | Jequié | BA | 7335654321 | bardoAgemiro@gmail.com |

1.

2.Obs:mostrei os resultados no s SQL Server pois, esse teste era pra ver somente se conseguimos incluir e alterar uma pessoa jurídica!

7. Incluindo, alterando e buscando uma pessoa juridica

a.Dados para inserir:

```
pj = new PessoaJuridica(seqM, "Café do Bosque", "Shopping", "Jequié", "BA", " 7333337890",
    "contato@cafedobosque.com.br", "61619480018");
```

b.Dados para alterar

```
pj.setNome("Café Recanto Tropical");
pj.setLogradouro("avenida");
pj.setCidade("Jequié");
pj.setEstado("BA");
pj.setTelefone("7332324567");
pj.setEmail("recantotropical@cafetropical.je");
```

c.Resultados do teste:

8.Inserindo, alterando, buscando e excluindo

a.Dados para inserir

```
pj = new PessoaJuridica(seqM, "Café Estação", "Avenida", "Jequié", "BA",
    "7335874321","estacaocafe@estacao.je", "79331345054");
```

b.Dados para alterar

```

pj.setNome("Café Aroma da Terra");
pj.setLogradouro("avenida");
pj.setCidade("Jequié");
pj.setEstado("BA");
pj.setTelefone("7335411212");
pj.setEmail("aromadaterra@cafejequie.br");

```

c.Resultados do teste

```

=====Inserindo PJ=====
CNPJ79331345054
Nome: Café Estação
Logradouro: Avenida
Cidade: Jequié
Estado: null
Telefone BA
Email: estacaocafe@estacao.je
=====Alterando PJ=====
CNPJ79331345054
Nome: Café Aroma da Terra
Logradouro: avenida
Cidade: Jequié
Estado: null
Telefone BA
Email: aromadaterra@cafejequie.br
=====Buscando PJ=====
CNPJ79331345054
Nome: Café Aroma da Terra
Logradouro: avenida
Cidade: Jequié
Estado: null
Telefone BA
Email: aromadaterra@cafejequie.br

```

9.Incluindo, alterando, buscando e excluindo uma pessoa juridica

a.Dados para inserir:

```

i. pj = new PessoaJuridica(seqM, "Cantinho do café", "rua", "Jequié", "BA",
    "7337778888", "cantinhodocafe@jequiecafe.com", "78771426094");

```

b.Dados para alterar

```

i. pj.setNome("Café Avenida");
   pj.setLogradouro("avenida");
   pj.setCidade("Jequié");
   pj.setEstado("BA");
   pj.setTelefone("7336665555");
   pj.setEmail("cafeAvenida@cafejequie.br");

```

c.Resultado do teste

```

=====Inserindo PJ=====
CNPJ78771426094
Nome: Cantinho do café
Logradouro: rua
Cidade: Jequié
Estado: null
Telefone BA
Email: cantinhodocafe@jequiecafe.com
=====Alterando PJ=====
CNPJ78771426094
Nome: Café Avenida
Logradouro: avenida
Cidade: Jequié
Estado: null
Telefone BA
Email: cafeAvenida@cafejequie.br
=====Buscando PJ=====
CNPJ78771426094
Nome: Café Avenida
Logradouro: avenida
Cidade: Jequié
Estado: null
Telefone BA
Email: cafeAvenida@cafejequie.br
=====Exluindo PJ=====
=====TESTES FINALIZADOS=====

```

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

a.Resposta: O JDBC é uma ferramenta de fundamental importância para o desenvolvimento Java, pois abstrai a complexidade de ter que lidar com um banco de dados, tornando-a mais simples para o programador além de permitirem a comunicação entre diferentes tecnologias, no caso do JDBC permite que vários apps java se comuniquem com vários tipos de bancos de dados, nessa aplicação foi usado o SQL Server, mas poderíamos usar o JDBC para se comunicar com mais de um banco por exemplo o SQL Server e o MongoDB. Além de fornecer ferramentas de segurança para melhorar a transação de dados entre o DB e o backend.

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

a.Resposta: A principal diferença é na forma que as consultas SQL são feitas, no *Statement*, as consultas são feitas sem parâmetros, elas são enviadas ao banco de dados diretamente como Strings, mas apresenta uma vulnerabilidade de injeção de SQL se os valores das consultas forem alterados, além de ser mais lento quando se trata de repetidas consultas. Já o *PreparedStatement* precisa de parâmetros pra realizar a conexão e elas são pré-compiladas pelo banco, fazendo com que ele seja mais rápido para consultas simultâneas, os parâmetros são definidos por posição de

consulta, isso ajuda a diminuir a chance se sofrer uma injeção de SQL, pois cada consulta é tratada separadamente!

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

a.Resposta: O padrão DAO ajuda na manutenibilidade do código porque ele separa as responsabilidades da lógica de acesso a uma camada separada, essas operações estão contidas (encapsuladas), fazendo com que fique mais fácil de se compreender os operadores. Além de não necessitar de atenção a detalhes específicos de cada banco de dados, facilita também mudar de um banco relacional como o SQL Server, para um não relacional como o MongoDB. Como o código é encapsulado, é mais fácil de reaproveitar ele dentro do projeto além de que facilita os testes

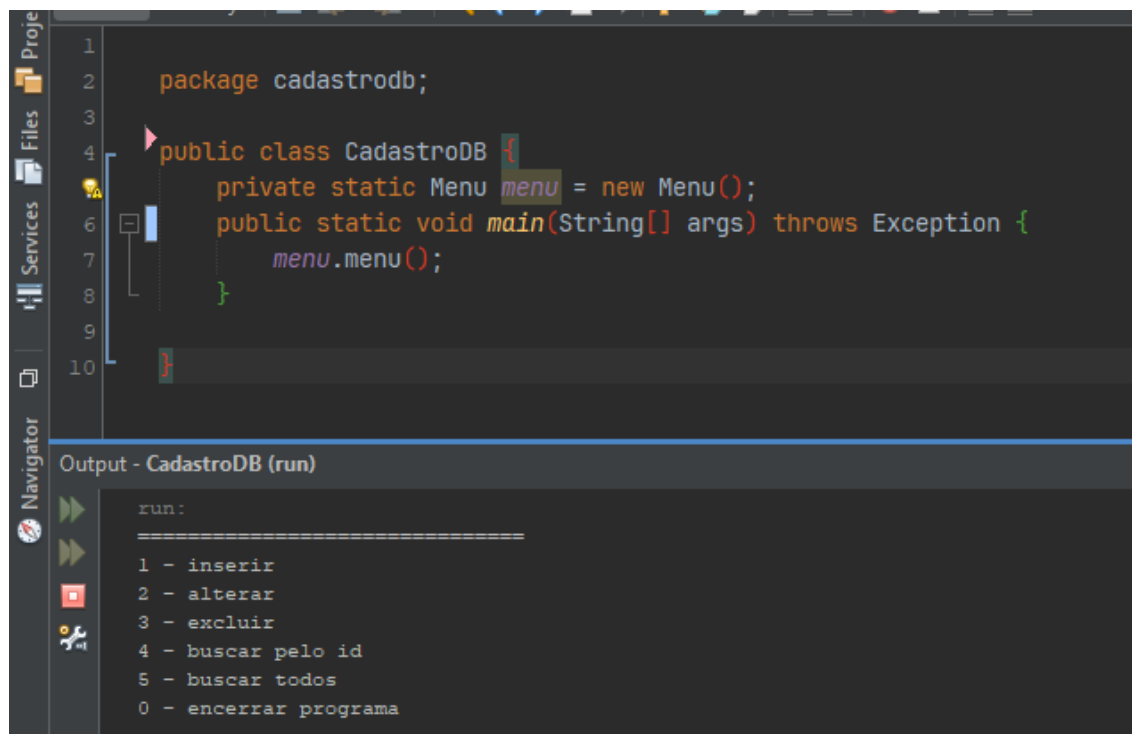
d) Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?

a.Resposta: Quando lidamos com o modelo SQL (bancos relacionais) a herança é um ponto fundamental, pois entrelaça os dados através de chaves estrangeiras (*foreign keys*) e se necessário excluir um dado, ele tem que ser modificado em cadeia, da parte menor até o maior, como uma pirâmide de cabeça pra baixo. No caso desta aplicação, se fossemos excluir uma pessoa diretamente da tabela de pessoas, precisaríamos primeiro excluir da tabela de pessoas físicas se ela fosse uma pessoa física e depois na tabela de pessoas, e se mais tabelas a puxassem, ainda chamassem ela, teríamos que excluir até chegar no pessoas físicas e depois no pessoas. Quando lidamos com esse tipo de banco temos três maneiras de herdar os dados, em tabelas separadas, cada classe do Java puxava sua própria tabela, o modelo a tabela única com coluna do tipo discriminante, onde todos os dados comuns são armazenados em uma única tabela e o dados diferente em uma tabela a parte. E a tabela única com junção de outras tabelas, no caso juntaríamos várias tabelas dentro de uma só pegando apenas as colunas que queremos, por exemplo, mostrar nome, cpf, logradouro, cidade, estado, telefone e id de todas as pessoas físicas do banco, isso seria uma junção da tabela Pessoas contendo o nome, logradouro, cidade, estado, telefone e id e da tabela PessoasFísicas contendo o CPF.

2º Procedimento | Alimentando a Base

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 2º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

Método, main, classe CadastroDB, está é a principal classe do projeto, o main chama o método menu, que é uma classe onde toda a interface de terminal que o usuário acessa é feita, chamando todos os métodos de pessoas físicas e jurídicas dentro do menu



The screenshot shows an IDE with a dark theme. The top pane displays the source code for `CadastroDB.java`. The code is as follows:

```
1 package cadastrodb;
2
3
4 public class CadastroDB {
5     private static Menu menu = new Menu();
6     public static void main(String[] args) throws Exception {
7         menu.menu();
8     }
9 }
10
```

The bottom pane, titled "Output - CadastroDB (run)", shows the output of the program:

```
run:
=====
1 - inserir
2 - alterar
3 - excluir
4 - buscar pelo id
5 - buscar todos
0 - encerrar programa
```

Ao executar o programa o menu logo é exibido

Classe Menu, metodo menu: este é responsavel por mostrar o usuário as opções de consulta que ele pode fazer ao banco de dados:


```
public void menu() throws SQLException {
    System.out.println("=====");
    System.out.println("""
        1 - inserir
        2 - alterar
        3 - excluir
        4 - buscar pelo id
        5 - buscar todos
        0 - encerrar programa
        """);
    int option = Integer.parseInt(sc.nextLine());
    switch (option) {
        case 0: break;
        case 1: inserirPessoa(); break;
        case 2: alterarPessoa(); break;
        case 3: excluirPessoa(); break;
        case 4: buscarPeloId(); break;
        case 5: buscarTodos(); break;
    }
}
```

Método `inserirPessoa` da classe `menu`: este método pergunta ao usuário se ele quer adicionar uma pessoa física ou jurídica e depois pede ao usuário que digite os atributos necessários para esta criação

```

public void inserirPessoa() throws Exception {
    String sequencia;
    int seqM;
    char res = question("inserir");
    switch (res) {
        case 'F' → {
            try {
                sequencia = "seq_pessoa";
                seqM = seq.getValue(sequencia);
                sc.nextLine();

                System.out.println("Insira o nome");
                nome = sc.nextLine();

                System.out.println("Insira o logradouro");
                logradouro = sc.nextLine();

                System.out.println("Insira a cidade");
                cidade = sc.nextLine();

                System.out.println("Insira o estado (SOMENTE A SIGLA)");
                estado = sc.nextLine();

                System.out.println("Insira o telefone (DDD9xxxxxxxx)");
                telefone = sc.nextLine();

                System.out.println("Insira o email");
                email = sc.nextLine();

                System.out.println("Insira o cpf");
            }
        }
    }
}

```

(continua na proxima pagina)

```

        cpf = sc.nextLine();

        sc.close();

        pf = new PessoaFisica(seqM, nome, logradouro, cidade, estado, telefone, email, cpf);
        pfDAO.incluiPessoaFisica(pf);
        pf.exibir();

    } catch (Exception e) {
        String erro = errorMessage("incluir", "juridica");
        LOGGER.log(Level.SEVERE, erro, e);
    }
    break;
}
case 'J' → {
    try {
        sequencia = "seq_pessoa";
        seqM = seq.getValue(sequencia);

        sc.nextLine();

        System.out.println("Insira o nome");
        nome = sc.nextLine();

        System.out.println("Insira o logradouro");
        logradouro = sc.nextLine();

        System.out.println("Insira a cidade");
        cidade = sc.nextLine();
    }
}

```

```

        cidade = sc.nextLine();

        System.out.println("Insira o estado (SOMENTE A SIGLA);");
        estado = sc.nextLine();

        System.out.println("Insira o telefone (DDD9xxxxxxxxx");
        telefone = sc.nextLine();

        System.out.println("Insira o email");
        email = sc.nextLine();

        System.out.println("Insira o CNPJ (xxxxxxxxxxx");
        cnpj = sc.nextLine();

        sc.close();

        pj = new PessoaJuridica (seqM, nome, logradouro, cidade, estado, telefone, email, cnpj);
        pj.exibir();
        pjDAO.incluirPessoaJuridica(pj);

    } catch (Exception e) {
        String erro = errorMessage("incluir", "juridica");
        LOGGER.Log(Level.SEVERE, erro, e);
    }
    break;

```

Metodo alterarPessoa, este metodo pergunta ao usuário se ele quer alterar uma pessoa fisica ou juridica e depois do usuário escolher mostra os atributos para que o uruário digite

```

public void alterarPessoa() throws Exception {
    char res = question("alterar");
    switch (res) {
        case 'F' → {
            pfDAO.exibirPessoasFisicas();
            int idPessoaFisica = questionId("fisica", "alterar ");
            PessoaFisica pf =
                pfDAO.getPessoaFisica(idPessoaFisica);

            sc.nextLine();

            System.out.println("Insira o novo nome");
            nome = sc.nextLine();
            pf.setNome(nome);

            System.out.println("Insira o novo logradouro");
            logradouro = sc.nextLine();
            pf.setLogradouro(logradouro);

            System.out.println("Insira a nova cidade cidade");
            cidade = sc.nextLine();
            pf.setCidade(cidade);

            System.out.println("Insira o novo estado (SOMENTE A SIGLA);");
            estado = sc.nextLine();
            pf.setEstado(estado);

            System.out.println("Insira o novo telefone (DDD9xxxxxxxxx");
            telefone = sc.nextLine();

```

```

05         pf.setTelefone(telefone);
06
07         System.out.println("Insira novo o email");
08         email = sc.nextLine();
09         pf.setEmail(email);
10
11         pfDAO.alterarPessoaFisica(idPessoaFisica, pf);
12         pfDAO.exibirPessoaFisica(idPessoaFisica);
13         sc.close();
14     }
15     case 'J' → {
16         pjDAO.exibirPessoasJuridicas();
17
18         int idPJ = questionId("fisica", "alterar ");
19         PessoaJuridica pj =
20             pjDAO.getPessoaJuridica(idPJ);
21
22         sc.nextLine();
23
24         System.out.println("Insira o novo nome");
25         nome = sc.nextLine();
26         pj.setNome(nome);
27
28         System.out.println("Insira o novo logradouro");
29         logradouro = sc.nextLine();
30         pj.setLogradouro(logradouro);
31
32         System.out.println("Insira a nova cidade cidade");
33         cidade = sc.nextLine();
34         pj.setCidade(cidade);
35

```

Método excluirPessoa da classe Menu, ela é responsável por mostrar ao usuário por exibir as pessoas físicas ou jurídicas do banco (dependendo da escolha do usuário), e perguntar o id da pessoa que ele deseja excluir

```

public void excluirPessoa() throws SQLException {
    char res = question("excluir");
    int idPessoaFisica = 0;
    int idPJ = 0;
    switch (res) {
        case 'F' → {
            try {
                pfDAO.exibirPessoasFisicas();
                idPessoaFisica = questionId("fisica", "excluir");
                pfDAO.excluirPessoa(idPessoaFisica);
            } catch (SQLException e) {
                String erro = errorMessage("excluir", "fisica");
                LOGGER.log(Level.SEVERE, erro + " com id: " + idPessoaFisica, e);
            }
            break;
        }
        case 'J' → {
            try {
                pjDAO.exibirPessoasJuridicas();
                idPJ = questionId("juridica", "excluir");
                pjDAO.excluirPessoa(idPJ);
            } catch (SQLException e) {
                String erro = errorMessage("excluir", "juridica");
                LOGGER.log(Level.SEVERE, erro + " com o id: " + idPJ, e);
            }
            break;
        }
    }
}

```

Método buscarPeloId da classe Menu, ele é responsável por buscar a pessoa do tipo que o usuário escolher e acionar o método exibirPessoa da classe PessoaFisicaDAO ou PessoaJuridicaDAO, este método exibe a pessoa pelo id, mostrando todos os atributos dela

```

public void buscarPeloId() throws SQLException {
    char res = question("buscar");
    int idPessoaFisica = 0;
    int idPJ = 0;
    switch (res) {
        case 'F' → {
            try {
                idPessoaFisica = questionId("física", "buscar");
                pfDAO.exibirPessoaFisica(idPessoaFisica);
            } catch (SQLException e) {
                String erro = errorMessage("buscar", "física");
                LOGGER.log(Level.SEVERE, erro + " com o id: " + idPessoaFisica, e);
            }
            break;
        }
        case 'J' → {
            try {
                idPJ = questionId("jurídica", "buscar");
                pjDAO.exibirPessoaJuridica(idPJ);
            } catch (SQLException e) {
                String erro = errorMessage("buscar", "jurídica");
                LOGGER.log(Level.SEVERE, erro + " com o id: " + idPJ, e);
            }
            break;
        }
    }
}

```

Método buscarTodos, ele pergunta ao usuário qual lista de pessoas ele quer ver e a exibe, usando o método exibirPessoaFisica ou exibirPessoaJuridica, cada uma das suas respectivas classes, PessoaFisicaDAO e PessoaJuridicaDAO

```

public void buscarTodos() throws SQLException {
    System.out.println("Você quer buscar todas as pessoas físicas ou jurídicas? ");
    String res = sc.nextLine();
    try{
        if(res.equalsIgnoreCase("F")) {
            pfDAO.exibirPessoasFisicas();
        } else if (res.equalsIgnoreCase("J")) {
            pjDAO.exibirPessoasJuridicas();
        } else {
            System.out.println("Escolha um tipo válido");
        }
    } catch (SQLException e) {
        LOGGER.log(Level.SEVERE, "Erro ao buscar todas as pessoas físicas do banco", e);
    }
}

```

Testes do programa!

- Testando incluir uma pessoa física
 -

```
=====
1 - inserir
2 - alterar
3 - excluir
4 - buscar pelo id
5 - buscar todos
0 - encerrar programa

1
Você deseja inserir uma pessoa física ou jurídica? F
Insira o nome
Gabriel Rocha Andrade
Insira o logradouro
Rua
Insira a cidade
Betim
Insira o estado (SOMENTE A SIGLA)
MG
Insira o telefone (DDD9xxxxxxxx)
31987654321
Insira o email
gabrielrocha@gmail.com
Insira o cpf
73156236047
BUILD SUCCESSFUL (total time: 3 minutes 7 seconds)
```

| | idPessoaFisica | CPF | Nome | Logradouro | Cidade | Estado | Telefone | Email |
|----|----------------|-------------|-----------------------|------------|----------------|--------|-------------|---------------------------|
| 1 | 190 | 02313901491 | Rafael | parque | Rio de Janeiro | SP | 11987654321 | Daniel@gmail.com |
| 2 | 204 | 23988273058 | Lucia | Condominio | Salvador | BA | 73998765432 | Ayme@gmail.com |
| 3 | 202 | 26850513095 | Ayme | Condominio | Jequié | BA | 7334567890 | Ayme@gmail.com |
| 4 | 1 | 41412651561 | Pedro Boaventura | APTO | Jitauna | BA | 73984521203 | pedroboaventura@gmail.com |
| 5 | 179 | 48217469521 | Rafaela | rua | Jequié | BA | 12345678900 | Israel@gmail.com |
| 6 | 191 | 49092382477 | Natalia | rua | Salvador | BA | 71987654321 | natalia@gmail.com |
| 7 | 193 | 55696105530 | Julia | rua | Ilhéus | BA | 7333456789 | Julia@gmail.com |
| 8 | 206 | 59821873057 | Lucia | Condominio | Salvador | BA | 73998765432 | Lucia@gmail.com |
| 9 | 221 | 73156236047 | Gabriel Rocha Andrade | Rua | Betim | MG | 31987654321 | gabrielrocha@gmail.com |
| 10 | 189 | 75150973459 | Daniel | parque | São Paulo | SP | 11912345678 | Daniel@gmail.com |
| 11 | 200 | 75647250017 | Femanda | Condominio | Jequié | BA | 7335356789 | Femanda@gmail.com |

- Testando incluir uma pessoa juridica no banco

```
run:
=====
1 - inserir
2 - alterar
3 - excluir
4 - buscar pelo id
5 - buscar todos
0 - encerrar programa

1
Você deseja inserir uma pessoa física ou jurídica? J
Insira o nome
Bar do Bartolomeu
Insira o logradouro
rua
Insira a cidade
Betim
Insira o estado (SOMENTE A SIGLA
MG
Insira o telefone (DDD9xxxxxxxx
3740028922
Insira o email
barthpix@gmail.com
Insira o CNPJ (xxxxxxxxxxx
97869592056
BUILD SUCCESSFUL (total time: 54 seconds)
```

| | idPJ | CNPJ | Nome | Logradouro | Cidade | Estado | Telefone | Email |
|----|------|-------------|------------------------|------------|----------------|--------|-------------|---------------------------------|
| 6 | 6 | 1516541351 | Heiniken | Parque | Belo Horizonte | MG | 31999999999 | Heiniken@email.com |
| 7 | 4 | 2151321531 | Budwiser | Rua | São Luis | Ma | 11999999999 | budwiser@email.com |
| 8 | 226 | 41181298091 | Bar do Zeca | rua | Betim | MG | 3140028922 | zecabar@pixemail.com |
| 9 | 198 | 52843617090 | Bar do Pedro | Avenida | Jequié | BA | 7335178901 | Pedraoda51@gmail.com |
| 10 | 212 | 55115465000 | Bar do Jocemar | avenida | Jequié | BA | 7335501234 | PIXBarDoJocemar@gmail.com |
| 11 | 7 | 5531546535 | Corona | Parque | Belo Horizonte | MG | 31999999999 | Corona@email.com |
| 12 | 8 | 58796413561 | Coca-cola | Fabrica | Uberlandia | MG | 31336541325 | cocacolabr@gmail.com |
| 13 | 215 | 61619480018 | Café Recanto Tropical | avenida | Jequié | BA | 7332324567 | recantotropical@cafetropical.je |
| 14 | 199 | 62221978099 | Bar do Agemiro | rua | Jequié | BA | 7335654321 | bardoAgemiro@gmail.com |
| 15 | 214 | 71332463096 | Café Canto dos Páss... | avenida | Jequié | BA | 733102-0033 | cantodospassaros@cafejequ... |
| 16 | 219 | 79331345054 | Café Aroma da Terra | avenida | Jequié | BA | 7335411212 | aromadaterra@cafejequie.br |
| 17 | 227 | 97869592056 | Bar do Bartolomeu | rua | Betim | MG | 3740028922 | barthpix@gmail.com |

- Testando aterar uma pessoa física do banco

```
Output - CadastroDB (run) × CadastroDB.java × Menu.java [-/M] ×
-----
4 - buscar pelo id
5 - buscar todos
0 - encerrar programa

2
Você deseja alterar uma pessoa física ou jurídica? f
ID: 190
Nome: Rafael
Logradouro: parque
Cidade: Rio de Janeiro
Estado: SP
Telefone: 11987654321
Email: Daniel@gmail.com
-----
ID: 204
Nome: Lucia
Logradouro: Condominio
Cidade: Salvador
Estado: BA
Telefone: 73998765432
Email: Ayme@gmail.com
-----
ID: 202
Nome: Ayme
Logradouro: Condominio
Cidade: Jequié
Estado: BA
Telefone: 7334567890
Email: Ayme@gmail.com
-----
ID: 1
Nome: Pedro Boaventura
Logradouro: APTO
Cidade: Jitauna
Estado: BA
Telefone: 73984521203
Email: pedroboaventura@gmail.com
```



```
Qual o id da pessoa física que você deseja alterar 200
Insira o novo nome
Fernanda Marcia
Insira o novo logradouro
Condominio
Insira a nova cidade cidade
Jequié
Insira o novo estado (SOMENTE A SIGLA
BA
Insira o novo telefone (DDD9xxxxxxxx
7389562214
Insira novo o email
fernandamarciamateriais@gmail.com
CPF: 75647250017
Nome: Fernanda Marcia
Logradouro: Condominio
Cidade: Condominio
Estado: BA
Telefone 7389562214
Email: fernandamarciamateriais@gmail.com
BUILD SUCCESSFUL (total time: 1 minute 50 seconds)
```

- Testando alterar uma pessoa jurídica do banco

0 - encerrar programa

2

Você deseja alterar uma pessoa física ou jurídica? J

ID: 224

Nome: Bar do Zeca

Logradouro: rua

Cidade: Betim

Estado: MG

Telefone: 3140028922

Email: zecabar@pixemail.com

ID: 211

Nome: Bar do Iran

Logradouro: rua

Cidade: Jequié

Estado: BA

Telefone: 7335312345

Email: JoaoDa51@gmail.com

ID: 197

Nome: Magno Express

Logradouro: Avenida

Cidade: Jequié

Estado: BA

Telefone: 7335512345

Email: magnoCafes@gmail.com

ID: 5

Nome: Ambev

Logradouro: Avenida

Cidade: Petropolis

Estado: RJ

Telefone: 21999999999

Email: ambev@gmail.com

Qual o id da pessoa jurídica que você deseja alterar 199

Insira o novo nome

Agemiro's Br

Insira o novo logradouro

rua

Insira a nova cidade cidade

Jequié

Insira o novo estado (SOMENTE A SIGLA

BA

Insira o novo telefone (DDD9xxxxxxxxx

7335286414

Insira novo o email

bardoagemiro@gmail.com

CNPJ62221978099

Nome: Bar do Agemiro

Logradouro: rua

Cidade: Jequié

Estado: BA

Telefone BA

Email: bardoAgemiro@gmail.com

BUILD SUCCESSFUL (total time: 43 seconds)

| | idPJ | CNPJ | Nome | Logradouro | Cidade | Estado | Telefone | Email |
|----|------|-------------|------------------------|------------|----------------|--------|-------------|---------------------------------|
| 6 | 6 | 1516541351 | Heiniken | Parque | Belo Horizonte | MG | 31999999999 | Heiniken@email.com |
| 7 | 4 | 2151321531 | Budwiser | Rua | São Luis | Ma | 11999999999 | budwiser@email.com |
| 8 | 226 | 41181298091 | Bar do Zeca | rua | Betim | MG | 3140028922 | zecabar@pixemail.com |
| 9 | 198 | 52843617090 | Bar do Pedro | Avenida | Jequié | BA | 7335178901 | PedraDa51@gmail.com |
| 10 | 212 | 55115465000 | Bar do Jocemar | avenida | Jequié | BA | 7335501234 | PIXBarDoJocemar@gmail.com |
| 11 | 7 | 5531546535 | Corona | Parque | Belo Horizonte | MG | 31999999999 | Corona@email.com |
| 12 | 8 | 58796413561 | Coca-cola | Fabrica | Uberlandia | MG | 31336541325 | cocacolabr@gmail.com |
| 13 | 215 | 61619480018 | Café Recanto Tropical | avenida | Jequié | BA | 7332324567 | recantotropical@cafetropical.je |
| 14 | 199 | 62221978099 | Bar do Agemiro | rua | Jequié | BA | 7335654321 | bardoAgemiro@gmail.com |
| 15 | 214 | 71332463096 | Café Canto dos Páss... | avenida | Jequié | BA | 733102-0033 | cantodospassaros@cafejequ... |
| 16 | 219 | 79331345054 | Café Aroma da Terra | avenida | Jequié | BA | 7335411212 | aromadaterra@cafejequie.br |
| 17 | 227 | 97869592056 | Agemiro's Br | rua | Jequié | BA | 7335286414 | bardoagemiro@gmail.com |

- Testando excluir uma pessoa física do banco

```

3
Você deseja excluir uma pessoa física ou jurídica? F
ID: 190
Nome: Rafael
Logradouro: parque
Cidade: Rio de Janeiro
Estado: SP
Telefone: 11987654321
Email: Daniel@gmail.com
-----
ID: 204
Nome: Lucia
Logradouro: Condominio
Cidade: Salvador
Estado: BA
Telefone: 73998765432
Email: Ayme@gmail.com
-----
ID: 202
Nome: Ayme
Logradouro: Condominio
Cidade: Jequié
Estado: BA
Telefone: 7334567890
Email: Ayme@gmail.com
-----
ID: 1
Nome: Pedro Boaventura
Logradouro: APT0
Cidade: Jitauna
Estado: BA
Telefone: 73984521203
Email: pedroboaventura@gmail.com
-----
ID: 179
Nome: Rafaela
Logradouro: rua
Cidade: Jequié

```

```
-----
Qual o id da pessoa física que você deseja excluir221
BUILD SUCCESSFUL (total time: 19 seconds)
```

| | idPessoaFísica | CPF | Nome | Logradouro | Cidade | Estado | Telefone | Email |
|----|----------------|-------------|------------------|------------|----------------|--------|-------------|----------------------------------|
| 1 | 190 | 02313901491 | Rafael | parque | Rio de Janeiro | SP | 11987654321 | Daniel@gmail.com |
| 2 | 204 | 23988273058 | Lucia | Condominio | Salvador | BA | 73998765432 | Ayme@gmail.com |
| 3 | 202 | 26850513095 | Ayme | Condominio | Jequié | BA | 7334567890 | Ayme@gmail.com |
| 4 | 1 | 41412651561 | Pedro Boaventura | APTO | Jitauna | BA | 73984521203 | pedroboaventura@gmail.com |
| 5 | 179 | 48217469521 | Rafaela | rua | Jequié | BA | 12345678900 | Israel@gmail.com |
| 6 | 191 | 49092382477 | Natalia | rua | Salvador | BA | 71987654321 | natalia@gmail.com |
| 7 | 193 | 55696105530 | Julia | rua | Ilhéus | BA | 7333456789 | Julia@gmail.com |
| 8 | 206 | 59821873057 | Lucia | Condominio | Salvador | BA | 73998765432 | Lucia@gmail.com |
| 9 | 189 | 75150973459 | Daniel | parque | São Paulo | SP | 11912345678 | Daniel@gmail.com |
| 10 | 200 | 75647250017 | Femanda Marcia | Condominio | Jequié | BA | 7389562214 | femandamarciamateriais@gmail.com |

- Testando excluir uma pessoa jurídica do banco

```
run:
=====
1 - inserir
2 - alterar
3 - excluir
4 - buscar pelo id
5 - buscar todos
0 - encerrar programa

3
Você deseja excluir uma pessoa física ou jurídica? j
ID: 224
Nome: Bar do Zeca
Logradouro: rua
Cidade: Betim
Estado: MG
Telefone: 3140028922
Email: zecabar@pixemail.com
-----
ID: 211
Nome: Bar do Iran
Logradouro: rua
Cidade: Jequié
Estado: BA
Telefone: 7335312345
Email: JoaoDa51@gmail.com
-----
ID: 197
Nome: Magno Express
Logradouro: Avenida
Cidade: Jequié
Estado: BA
Telefone: 7335512345
Email: magnoCafes@gmail.com
-----
ID: 5
Nome: Ambev
Logradouro: Avenida
```

```
-----
Qual o id da pessoa jurídica que você deseja excluir214
BUILD SUCCESSFUL (total time: 10 seconds)
```

| | idPJ | CNPJ | Nome | Logradouro | Cidade | Estado | Telefone | Email |
|----|------|-------------|-----------------------|------------|----------------|--------|-------------|---------------------------------|
| 1 | 224 | | Bar do Zeca | rua | Betim | MG | 3140028922 | zecabar@pixemail.com |
| 2 | 211 | 05840379530 | Bar do Iran | rua | Jequié | BA | 7335312345 | JoaoDa51@gmail.com |
| 3 | 197 | 06718410098 | Magno Express | Avenida | Jequié | BA | 7335512345 | magnoCafes@gmail.com |
| 4 | 5 | 1232151546 | Ambev | Avenida | Petropolis | RJ | 21999999999 | ambev@email.com |
| 5 | 213 | 13786036055 | Casa do Café | avenida | Jequié | BA | 7335301234 | PIXCasaDoCafé@gmail.com |
| 6 | 6 | 1516541351 | Heiniken | Parque | Belo Horizonte | MG | 31999999999 | Heiniken@email.com |
| 7 | 4 | 2151321531 | Budwiser | Rua | São Luis | Ma | 11999999999 | budwiser@email.com |
| 8 | 226 | 41181298091 | Bar do Zeca | rua | Betim | MG | 3140028922 | zecabar@pixemail.com |
| 9 | 198 | 52843617090 | Bar do Pedro | Avenida | Jequié | BA | 7335178901 | PedraDa51@gmail.com |
| 10 | 212 | 55115465000 | Bar do Jocemar | avenida | Jequié | BA | 7335501234 | PIXBarDoJocemar@gmail.com |
| 11 | 7 | 5531546535 | Corona | Parque | Belo Horizonte | MG | 31999999999 | Corona@email.com |
| 12 | 8 | 58796413561 | Coca-cola | Fabrica | Uberlandia | MG | 31336541325 | cocacolabr@gmail.com |
| 13 | 215 | 61619480018 | Café Recanto Tropical | avenida | Jequié | BA | 7332324567 | recantotropical@cafetropical.je |
| 14 | 199 | 62221978099 | Bar do Agemiro | rua | Jequié | BA | 7335654321 | bardoAgemiro@gmail.com |
| 15 | 219 | 79331345054 | Café Aroma da Terra | avenida | Jequié | BA | 7335411212 | aromadaterra@cafejequie.br |
| 16 | 227 | 97869592056 | Agemiro's Br | rua | Jequié | BA | 7335286414 | bardoagemiro@gmail.com |

- Testando buscar uma pessoa fisica do banco

```

run:
=====
1 - inserir
2 - alterar
3 - excluir
4 - buscar pelo id
5 - buscar todos
0 - encerrar programa

4
Você deseja buscar uma pessoa física ou jurídica? F
Qual o id da pessoa física que você desja buscar202
CPF: 26850513095
Nome: Ayme
Logradouro: Condominio
Cidade: Condominio
Estado: BA
Telefone 7334567890
Email: Ayme@gmail.com
BUILD SUCCESSFUL (total time: 11 seconds)
|

```

- Testando buscar uma pessoa juridica do banco

```

=====
1 - inserir
2 - alterar
3 - excluir
4 - buscar pelo id
5 - buscar todos
0 - encerrar programa

4
Você deseja buscar uma pessoa física ou jurídica? J
Qual o id da pessoa jurídica que você deseja buscar226
CNPJ41181298091
Nome: Bar do Zeca
Logradouro: rua
Cidade: Betim
Estado: null
Telefone MG
Email: zecabar@pixemail.com
BUILD SUCCESSFUL (total time: 16 seconds)

```

- Testando buscar todas as pessoas físicas do banco:

```

5
Você quer buscar todas as pessoas físicas ou jurídicas?
F
ID: 190
Nome: Rafael
Logradouro: parque
Cidade: Rio de Janeiro
Estado: SP
Telefone: 11987654321
Email: Daniel@gmail.com
-----
ID: 204
Nome: Lucia
Logradouro: Condominio
Cidade: Salvador
Estado: BA
Telefone: 73998765432
Email: Ayme@gmail.com
-----
ID: 202
Nome: Ayme
Logradouro: Condominio
Cidade: Jequié
Estado: BA
Telefone: 7334567890
Email: Ayme@gmail.com
-----
ID: 1
Nome: Pedro Boaventura
Logradouro: APTO
Cidade: Jitauna

```

```
Cidade: Jitauna
Estado: BA
Telefone: 73984521203
Email: pedroboaventura@gmail.com
-----
```

```
ID: 179
Nome: Rafaela
Logradouro: rua
Cidade: Jequié
Estado: BA
Telefone: 12345678900
Email: Israel@gmail.com
-----
```

```
ID: 191
Nome: Natalia
Logradouro: rua
Cidade: Salvador
Estado: BA
Telefone: 71987654321
Email: natalia@gmail.com
-----
```

```
ID: 193
Nome: Julia
Logradouro: rua
Cidade: Ilheus
Estado: BA
Telefone: 7333456789
Email: Julia@gmail.com
-----
```

```
ID: 206
Nome: Lucia
Logradouro: Condominio
Cidade: Salvador
Estado: BA
Telefone: 73998765432
Email: Lucia@gmail.com
-----
```

```
ID: 189
```

```
ID: 189
Nome: Daniel
Logradouro: parque
Cidade: São Paulo
Estado: SP
Telefone: 11912345678
Email: Daniel@gmail.com
-----
```

```
ID: 200
Nome: Fernanda Marcia
Logradouro: Condominio
Cidade: Jequié
Estado: BA
Telefone: 7389562214
Email: fernandamarciamateriais@gmail.com
-----
```

```
BUILD SUCCESSFUL (total time: 7 seconds)
```

- Testando buscar todas as pessoas jurídicas do banco

```
5
Você quer buscar todas as pessoas físicas ou jurídicas?
J
ID: 224
Nome: Bar do Zeca
Logradouro: rua
Cidade: Betim
Estado: MG
Telefone: 3140028922
Email: zecabar@pixemail.com
-----
ID: 211
Nome: Bar do Iran
Logradouro: rua
Cidade: Jequié
Estado: BA
Telefone: 7335312345
Email: JoaoDa51@gmail.com
-----
ID: 197
Nome: Magno Express
Logradouro: Avenida
Cidade: Jequié
Estado: BA
Telefone: 7335512345
Email: magnoCafes@gmail.com
-----
ID: 5
Nome: Ambev
Logradouro: Avenida
Cidade: Petropolis
Estado: RJ
Telefone: 21999999999
```



```
Telefone: 3199999999
Email: ambev@email.com
-----
ID: 213
Nome: Casa do Café
Logradouro: avenida
Cidade: Jequié
Estado: BA
Telefone: 7335301234
Email: PIXCasaDoCafé@gmail.com
-----
ID: 6
Nome: Heiniken
Logradouro: Parque
Cidade: Belo Horizonte
Estado: MG
Telefone: 31999999999
Email: Heiniken@email.com
-----
ID: 4
Nome: Budwiser
Logradouro: Rua
Cidade: São Luis
Estado: Ma
Telefone: 11999999999
Email: budwiser@email.com
-----
ID: 226
Nome: Bar do Zeca
Logradouro: rua
Cidade: Betim
Estado: MG
Telefone: 3140028922
Email: zecabar@pixemail.com
-----
ID: 198
Nome: Bar do Pedro
Logradouro: Avenida
```

Output - CadastroDB (run) X CadastroDB.java X Menu.java [-7/M] X

```
Logradouro: Avenida
Cidade: Jequié
Estado: BA
Telefone: 7335178901
Email: Pedraoda51@gmail.com
-----
ID: 212
Nome: Bar do Jocemar
Logradouro: avenida
Cidade: Jequié
Estado: BA
Telefone: 7335501234
Email: PIXBarDoJocemar@gmail.com
-----
ID: 7
Nome: Corona
Logradouro: Parque
Cidade: Belo Horizonte
Estado: MG
Telefone: 31999999999
Email: Corona@email.com
-----
ID: 8
Nome: Coca-cola
Logradouro: Fabrica
Cidade: Uberlandia
Estado: MG
Telefone: 31336541325
Email: cocacolabr@gmail.com
-----
ID: 215
Nome: Café Recanto Tropical
Logradouro: avenida
Cidade: Jequié
Estado: BA
Telefone: 7332324567
Email: recantotropical@cafetropical.je
-----
```

```

Email: recantotropical@cafetropical.je
-----
ID: 199
Nome: Bar do Agemiro
Logradouro: rua
Cidade: Jequié
Estado: BA
Telefone: 7335654321
Email: bardoAgemiro@gmail.com
-----
ID: 219
Nome: Café Aroma da Terra
Logradouro: avenida
Cidade: Jequié
Estado: BA
Telefone: 7335411212
Email: aromadaterra@cafejequeie.br
-----
ID: 227
Nome: Agemiro's Br
Logradouro: rua
Cidade: Jequié
Estado: BA
Telefone: 7335286414
Email: bardoagemiro@gmail.com
-----
BUILD SUCCESSFUL (total time: 4 seconds)

```

- Testando sair do programa

```

=====
1 - inserir
2 - alterar
3 - excluir
4 - buscar pelo id
5 - buscar todos
0 - encerrar programa

0
BUILD SUCCESSFUL (total time: 1 second)
|

```

Explicando métodos customizados, criados conforme a minha necessidade:

- Método question: Ele recebe um verbo como parametro que pergunta ao usuário qual o tipo de pessoa que ele quer fazer com o usuário, esta ação vem através do verbo recebido como parametro e retorna o tipo da pessoa

```

private char question(String v) {
    System.out.printf("Você deseja %s uma pessoa física ou jurídica? ", v);
    return sc.next().toUpperCase().charAt(0);
}

```

Por exemplo, no método `inserir`, ele é chamado e a pergunta “Você deseja *inserir* uma pessoa física ou jurídica?”

- Método `questionId`: recebe dois parâmetros que o tipo da pessoa que o usuário deseja manipular, e um verbo, este verbo indica a ação que o usuário deseja fazer

```
private int questionId(String tipo, String verbo) {  
    System.out.printf("Qual o id da pessoa %s que você deseja %s", tipo, verbo);  
    int id = sc.nextInt();  
    return id;  
}
```

Por exemplo, quando queremos excluir um usuário o `questionId` é acionado para perguntar ao usuário qual é o tipo e qual é o id, então a pergunta fica “Qual id da pessoa *física* você deseja *excluir*?”

- Método `errorMessage`: recebe dois parâmetros, um sendo o verbo, que indica falha na tentativa de executar determinada ação e o tipo da pessoa em que ocorreu o erro

```
private String errorMessage(String verbo, String tipo) {  
    String message = "Erro ao %s uma pessoa %t" + verbo + tipo;  
    return message;  
}
```

Por exemplo, se algum erro ocorreu na inclusão de uma pessoa física, a mensagem de erro seria: “Erro ao *incluir* uma pessoa *física*, erro:” e aí apareceria o erro gerado!

Explicando o tratamento de erros:

Como você pode ver eu padronizei as mensagens de erro usando o `errorMessage`, mas também eu fiz o uso de duas bibliotecas do Java chamada `Logger` e `Level`, elas fazem um tratamento de erros mais seguros do que o `printStackTrace`, pois, quando o ele é acionado, ele mostra pontos sensíveis do código, isso causa vulnerabilidade no sistema, por isso o uso do `try-catch` e o `catch` usando o `Logger` foi fundamental, aqui está um exemplo de uso do `Logger` dentro do `catch`

```
} catch (SQLException e) {  
    String erro = errorMessage("incluir", "juridica");  
    LOGGER.log(Level.SEVERE, erro, e);  
}
```

O string `erro`, chama o método que exibe a mensagem de erro padrão, passando apenas os parâmetros necessários

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

a.Resposta: Quando persistimos os dados em um arquivo, criamos um arquivo contendo os dados desejados do programa, e exportamos eles criando uma cópia estática, pois a cada modificação seria necessário gerar um novo arquivo. Quando persistimos em um banco de dados, criamos um armazenamento mais seguro dos dados e mais eficiente de buscar os dados, e quando criamos um dado, excluimos ou atualizamos, não necessitamos criar um arquivo do banco de dados, pois ele pode ser alterável a qualquer momento

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

a.Resposta: O uso de operadores *lambda* entrou no mundo Java a partir do Java 8, antes era necessário iterar sobre a lista (com um laço for por exemplo) para fazer determinada ação agora, não é necessário usar isso! Pode se fazer de forma mais legível e simples

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

a.Resposta: Quando definimos um método como static simboliza que não precisamos instanciar ele para usar, podemos chama-lo diretamente sem necessidade de um mediador, como o main é static, todos os métodos chamados por ele precisam ser static, como o main é o contexto da classe e não uma instancia especifica, ele só pode chamar métodos estáticos diretamente, isso é uma convenção de código para deixar a leitura do código, pois como já mencionado, você não precisa instanciar o objeto pra poder usar ele

Observe que os tópicos acima seguem exatamente o que está na Atividade Prática exigida.

Conclusão

Elabore uma análise crítica da sua Missão Prática.