Missão Prática | Nível 3 | Mundo 3

Lorena Rosa Borges Sanches | Matrícula: 202204376067

Estácio Uberlândia (Polo Santa Mônica)/MG – Desenvolvimento Full Stack – Número da Turma – Mundo 3
Link GitHub: github.com/LorenaBorgesSanches/Mundo3-Pratica3

Objetivo da Prática

Trata-se de material de apoio e esclarecimento para o trabalho prático desenvolvido como requisito de aprovação no curso de Desenvolvimento Full Stack, do 3° semestre da instituição de ensino Estácio.

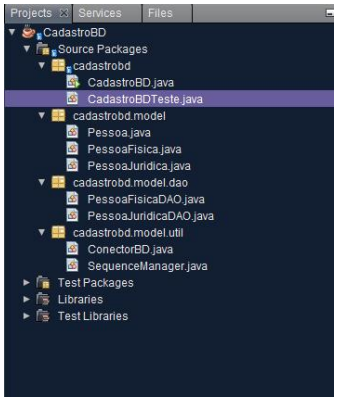
O trabalho é dividido em dois procedimentos, e nos dois procedimentos teremos a mesma organização de conteúdo. Sendo a primeira parte do conteúdo referente a prints dos códigos e seus respectivos resultados, e após, teremos perguntas elaboradas pela instituição de ensino seguidas de respostas.

Os softwares utilizados no banco de dados foi o SQL Server Management Studio, e no projeto de desenvolvimento NetBeans.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Códigos do 1° procedimento estão organizados conforme a sequência do material do trabalho do curso. Segue abaixo:

1. A- Criar um projeto no NetBeans, utilizando o nome CadastroBD, do tipo Aplicativo Java Padrão (modelo Ant).



2. (Os tópicos referentes a conexão com o banco não irei postar prints para simplificar o trabalho, já que no próprio material da prática há os prints explicando).

3. Criar o pacote c**adastrobd.model**, e nele criar as classes apresentadas a seguir:

A- Classe **Pessoa**, com os campos **id**, **nome**, **logradouro**, **cidade**, **estado**, **telefone** e **email**, construtor padrão e completo, além de método **exibir**, para impressão dos dados no console;

```
*/
public class Pessoa implements Serializable {

    private int id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

    public Pessoa(){

    }

    public Pessoa(int id, String nome, String logradouro, String cidade,
            String 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 exibir(){
        return "Id: "
                + id
                + "\nNome: "
                + nome
                + "\nLogradouro: "
                + logradouro
                + "\nCidade: "
                + cidade
                +"\nEstado: "
                + estado
                + "\nTelefone: "
                + telefone
                + "\nEmail: "
                + email;
    }

}
```

B- Classe **PessoaFisica**, herdando de Pessoa, com acréscimo do campo **cpf**, além da reescrita dos construtores e uso de polimorfismo em exibir;

```java
public class PessoaFisica extends Pessoa {

    private String cpf;

    public PessoaFisica(){

    }

    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;
    }

    /**
     * @return the cpf
     */
    public String getCpf() {
        return cpf;
    }

    /**
     * @param cpf the cpf to set
     */
    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    @Override
    public String exibir(){
        return super.exibir()+ "\nCpf: " + getCpf();
    }
}

```

C- Classe **PessoaJurídica**, herdando de Pessoa, com acréscimo do campo **cnpj**, além da reescrita dos construtores e uso de polimorfismo em exibir;

```java
public class PessoaJuridica extends Pessoa {

    private String cnpj;

    public PessoaJuridica(){
    }

    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;
    }

    /**
     * @return the cnpj
     */
    public String getCnpj() {
        return cnpj;
    }

    /**
     * @param cnpj the cnpj to set
     */
    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    @Override
    public String exibir(){
        return super.exibir()+ "\nCnpj: " + cnpj;
    }

}

```

4. Criar o pacotes **cadastro.model.util**, para inclusão das classes utilitárias que são apresentadas a seguir:

A- Classe **ConectorBD**, com os métodos **getConnection**, para retornar uma conexão com o banco de dados, **getPrepared**, para retornar um objeto do tipo **PreparedStatement** a partir de um SQL fornecido com parâmetro, e **getSelect**, para retornar o **ResultSet** relacionado a uma consulta.

```
public class ConectorBD {

    private Connection conexao;
    private PreparedStatement ps;
    private ResultSet rs;

    public Connection getConnection() throws ClassNotFoundException, SQLException {
        Class.forName(className: "com.microsoft.sqlserver.jdbc.SQLServerDriver");
        String connectionUrl = "jdbc:sqlserver://localhost\\DESKTOP-8PQGG84:1433;databaseName=Loja;user=loja;password=loja;trustServerCertificate=true";
        if (conexao == null || conexao.isClosed()) {
            conexao = DriverManager.getConnection(url: connectionUrl);
        }
        return conexao;
    }

    public PreparedStatement getPrepared(String sql) throws ClassNotFoundException, SQLException {
        if (conexao == null || conexao.isClosed()) {
            getConnection();
        }

        ps = conexao.prepareStatement(string: sql);
        return ps;
    }

    public ResultSet getSelect(String sql) throws ClassNotFoundException, SQLException {
        conexao = getConnection();
        ps = getPrepared(sql);
        rs = ps.executeQuery();
        return rs;
    }

```

B- Ainda na classe **ConectorBD**, adicionar métodos **close** sobrecarregados para **Statement**, **ResultSet** e **Connection**, visando garantir o fechamento, ou encerramento, de todos os objetos de acesso ao banco gerados.

```
    public void Close() throws SQLException {
        if (rs != null && !rs.isClosed()) {
            rs.close();
        }
        if (ps != null && !ps.isClosed()) {
            ps.close();
        }
        if (conexao != null && !conexao.isClosed()) {
            conexao.close();
        }
    }
}
```

C- Classe **SequenceManager**, que terá o método **getValue**, recebendo o nome da sequência como parâmetro e retornando o próximo valor.

```java
public class SequenceManager {

    public int getValue(ConectorBD bd) throws ClassNotFoundException, SQLException {
        ResultSet result = bd.getSelect(sql: "SELECT NEXT VALUE FOR idPessoaSequence");
        if (result.next()) {
            return result.getInt(i: 1);
        }
        result.close();
        return 0;
    }
}

```

5. Codificar as classes no padrão DAO, no pacote **cadastro.model.**

A- Classe **PessoaFisicaDAO**, com os métodos **getPessoa**, retornando uma pessoa física a partir do seu **id**, **getPessoas**, para retorno de todas as pessoas físicas do banco de dados, **incluir**, para inclusão de uma pessoa física, fornecida como **parâmetro**, nas tabelas Pessoa e PessoaFisica, **alterar**, para alteração dos dados de uma pessoa física, e **excluir**, para remoção da pessoa do banco em ambas as tabelas.

```java
public class PessoaFisicaDAO {

    private final ConectorBD bd;
    private final SequenceManager sequence;

    public PessoaFisicaDAO(ConectorBD bd) {
        this.bd = bd;
        this.sequence = new SequenceManager();
    }

    public PessoaFisica getPessoa(int id) throws ClassNotFoundException, SQLException {
        String consulta = "select * from Pessoa "
                + "inner join PessoaFisica on (Pessoa.idPessoa = PessoaFisica.Pessoa_idPessoa) "
                + "where idPessoa = ?";

        PreparedStatement ps = bd.getPrepared(sql: consulta);
        ps.setInt(i: 1, i1: id);
        ResultSet rs = ps.executeQuery();

        if (rs.next()) {
            PessoaFisica pessoa = new PessoaFisica();
            pessoa.setId(id: rs.getInt(string: "idPessoa"));
            pessoa.setNome(nome:rs.getString(string: "nome"));
            pessoa.setLogradouro(logradouro:rs.getString(string: "logradouro"));
            pessoa.setCidade(cidade: rs.getString(string: "cidade"));
            pessoa.setEstado(estado: rs.getString(string: "estado"));
            pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
            pessoa.setEmail(email:rs.getString(string: "email"));
            pessoa.setCpf(cpf: rs.getString(string: "cpf"));
            bd.Close();
            return pessoa;
        }

        bd.Close();
        return null;
    }
```

```java
public List<PessoaFisica> getPessoas() throws ClassNotFoundException, SQLException {
    ArrayList<PessoaFisica> arraypf = new ArrayList();
    String consulta = "select * from Pessoa "
            + "inner join PessoaFisica on (Pessoa.idPessoa = PessoaFisica.Pessoa_idPessoa) ";

    ResultSet rs = bd.getSelect(sql: consulta);
    while (rs.next()) {
        PessoaFisica pessoa = new PessoaFisica();
        pessoa.setId(id: rs.getInt(string: "idPessoa"));
        pessoa.setNome(nome:rs.getString(string: "nome"));
        pessoa.setLogradouro(logradouro:rs.getString(string: "logradouro"));
        pessoa.setCidade(cidade: rs.getString(string: "cidade"));
        pessoa.setEstado(estado: rs.getString(string: "estado"));
        pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
        pessoa.setEmail(email:rs.getString(string: "email"));
        pessoa.setCpf(cpf: rs.getString(string: "cpf"));

        arraypf.add(e: pessoa);
    }

    bd.Close();
    return arraypf;
}
```

```java
public void incluir(PessoaFisica pf) throws SQLException, ClassNotFoundException {
    String inserir = "insert into Pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email) values \n"
            + "(?, ?, ?, ?, ?, ?, ?);\n"
            + "\n"
            + "insert into PessoaFisica(Pessoa_idPessoa, cpf) values \n"
            + "(?, ?);";

    int id = sequence.getValue(bd);

    PreparedStatement ps = bd.getPrepared(sql: inserir);
    ps.setInt(i: 1, i1: id);
    ps.setString(i: 2, string: pf.getNome());
    ps.setString(i: 3, string: pf.getLogradouro());
    ps.setString(i: 4, string: pf.getCidade());
    ps.setString(i: 5, string: pf.getEstado());
    ps.setString(i: 6, string: pf.getTelefone());
    ps.setString(i: 7, string: pf.getEmail());
    ps.setInt(i: 8, i1: id);
    ps.setString(i: 9, string: pf.getCpf());

    ps.executeUpdate();

    bd.Close();

}
```

```java
    public void alterar(PessoaFisica pf) throws SQLException, ClassNotFoundException {
        String alterar = "UPDATE Pessoa "
                + "SET nome = ?,"
                + "logradouro = ?,"
                + "cidade= ?,"
                + "estado = ?,"
                + "telefone = ?,"
                + "email = ? "
                + "WHERE idPessoa = ?;\n"
                + "UPDATE PessoaFisica "
                + "SET cpf = ? "
                + "WHERE Pessoa_idPessoa = ?";

        PreparedStatement ps = bd.getPrepared(sql: alterar);
        ps.setString(i: 1, string: pf.getNome());
        ps.setString(i: 2, string: pf.getLogradouro());
        ps.setString(i: 3, string: pf.getCidade());
        ps.setString(i: 4, string: pf.getEstado());
        ps.setString(i: 5, string: pf.getTelefone());
        ps.setString(i: 6, string: pf.getEmail());
        ps.setInt(i: 7, i1: pf.getId());
        ps.setString(i: 8, string: pf.getCpf());
        ps.setInt(i: 9, i1: pf.getId());
        ps.executeUpdate();

        bd.close();

    }
```

```java
    public void excluir(PessoaFisica pf) throws SQLException, ClassNotFoundException {
        String delete = "DELETE FROM PessoaFisica WHERE Pessoa_idPessoa = ?;\n"
                + "DELETE FROM Pessoa WHERE idPessoa = ?";
        PreparedStatement ps = bd.getPrepared(sql: delete);
        ps.setInt(i: 1, i1: pf.getId());
        ps.setInt(i: 2, i1: pf.getId());
        ps.executeUpdate();

        bd.close();
    }

}
```

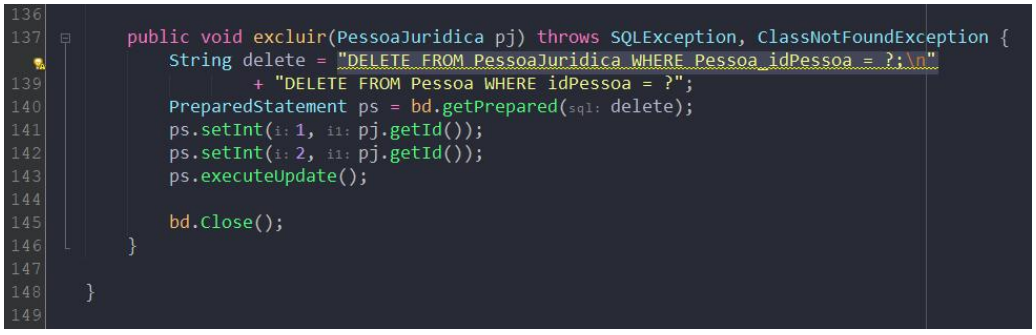
B- Classe **PessoaJuridicaDAO**, com os métodos **getPessoa**, retornando uma pessoa jurídica a partir do seu **id**, **getPessoas**, para retorno de todas as pessoas jurídicas do banco de dados, **incluir**, para inclusão de uma pessoa jurídica, fornecida como **parâmetro**, nas tabelas Pessoa e PessoaJuridica, **alterar**, para alteração dos dados de uma pessoa jurídica, e **excluir**, para remoção da pessoa do banco em ambas as tabelas.

```java
public class PessoaJuridicaDAO {

    private final ConectorBD bd;
    private final SequenceManager sequence;

    public PessoaJuridicaDAO(ConectorBD bd) {
        this.bd = bd;
        this.sequence = new SequenceManager();
    }

    public PessoaJuridica getPessoa(int id) throws ClassNotFoundException, SQLException {
        String consulta = "select * from Pessoa "
                + "inner join PessoaJuridica on (Pessoa.idPessoa = PessoaJuridica.Pessoa_idPessoa) "
                + "where Pessoa_idPessoa = ?";

        PreparedStatement ps = bd.getPrepared(sql: consulta);
        ps.setInt(i: 1, i1: id);
        ResultSet rs = ps.executeQuery();

        if (rs.next()) {
            PessoaJuridica pessoa = new PessoaJuridica();
            pessoa.setId(id: rs.getInt(string: "Pessoa_idPessoa"));
            pessoa.setNome(nome:rs.getString(string: "nome"));
            pessoa.setLogradouro(logradouro:rs.getString(string: "logradouro"));
            pessoa.setCidade(cidade: rs.getString(string: "cidade"));
            pessoa.setEstado(estado: rs.getString(string: "estado"));
            pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
            pessoa.setEmail(email:rs.getString(string: "email"));
            pessoa.setCnpj(cnpj:rs.getString(string: "cnpj"));
            bd.Close();
            return pessoa;
        }

        bd.Close();
        return null;
    }
```

```java
    public List<PessoaJuridica> getPessoas() throws ClassNotFoundException, SQLException {
        ArrayList<PessoaJuridica> arraypj = new ArrayList();
        String consulta = "select * from Pessoa "
                + "inner join PessoaJuridica on (Pessoa.idPessoa = PessoaJuridica.Pessoa_idPessoa) ";

        ResultSet rs = bd.getSelect(sql: consulta);
        while (rs.next()) {
            PessoaJuridica pessoa = new PessoaJuridica();
            pessoa.setId(id: rs.getInt(string: "Pessoa_idPessoa"));
            pessoa.setNome(nome:rs.getString(string: "nome"));
            pessoa.setLogradouro(logradouro:rs.getString(string: "logradouro"));
            pessoa.setCidade(cidade: rs.getString(string: "cidade"));
            pessoa.setEstado(estado: rs.getString(string: "estado"));
            pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
            pessoa.setEmail(email:rs.getString(string: "email"));
            pessoa.setCnpj(cnpj:rs.getString(string: "cnpj"));

            arraypj.add(e: pessoa);
        }

        bd.Close();
        return arraypj;
    }
```

```
    public void incluir(PessoaJuridica pj) throws SQLException, ClassNotFoundException {
        String inserir = "insert into Pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email) values \n"
                + "(?, ?, ?, ?, ?, ?, ?);\n"
                + "\n"
                + "insert into PessoaJuridica(Pessoa_idPessoa, cnpj) values \n"
                + "(?, ?);";

        int id = sequence.getValue(bd);

        PreparedStatement ps = bd.getPrepared(sql: inserir);
        ps.setInt(i: 1, i1: id);
        ps.setString(i: 2, string: pj.getNome());
        ps.setString(i: 3, string: pj.getLogradouro());
        ps.setString(i: 4, string: pj.getCidade());
        ps.setString(i: 5, string: pj.getEstado());
        ps.setString(i: 6, string: pj.getTelefone());
        ps.setString(i: 7, string: pj.getEmail());
        ps.setInt(i: 8, i1: id);
        ps.setString(i: 9, string: pj.getCnpj());

        ps.executeUpdate();

        bd.Close();

    }
```

```
    public void alterar(PessoaJuridica pj) throws SQLException, ClassNotFoundException {
        String alterar = "UPDATE Pessoa "
                + "SET nome = ?,"
                + "logradouro = ?,"
                + "cidade= ?,"
                + "estado = ?,"
                + "telefone = ?,"
                + "email = ? "
                + "WHERE idPessoa = ?;\n"
                + "UPDATE PessoaJuridica "
                + "SET cnpj = ? "
                + "WHERE Pessoa_idPessoa = ?";

        PreparedStatement ps = bd.getPrepared(sql: alterar);
        ps.setString(i: 1, string: pj.getNome());
        ps.setString(i: 2, string: pj.getLogradouro());
        ps.setString(i: 3, string: pj.getCidade());
        ps.setString(i: 4, string: pj.getEstado());
        ps.setString(i: 5, string: pj.getTelefone());
        ps.setString(i: 6, string: pj.getEmail());
        ps.setInt(i: 7, i1: pj.getId());
        ps.setString(i: 8, string: pj.getCnpj());
        ps.setInt(i: 9, i1: pj.getId());
        ps.executeUpdate();

        bd.Close();

    }
```



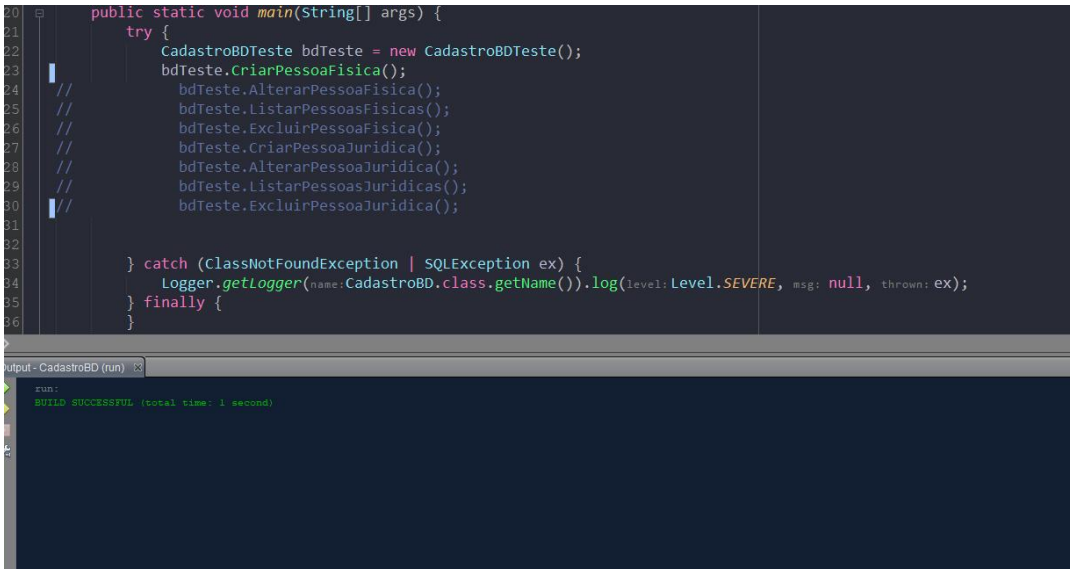
```
    public void excluir(PessoaJuridica pj) throws SQLException, ClassNotFoundException {
        String delete = "DELETE FROM PessoaJuridica WHERE Pessoa_idPessoa = ?;\n"
                + "DELETE FROM Pessoa WHERE idPessoa = ?";
        PreparedStatement ps = bd.getPrepared(sql: delete);
        ps.setInt(i: 1, i1: pj.getId());
        ps.setInt(i: 2, i1: pj.getId());
        ps.executeUpdate();

        bd.Close();
    }

}
```

C- Utilizar nas classes objetos dos tipos **ConectorBD** e **SequenceManager.**

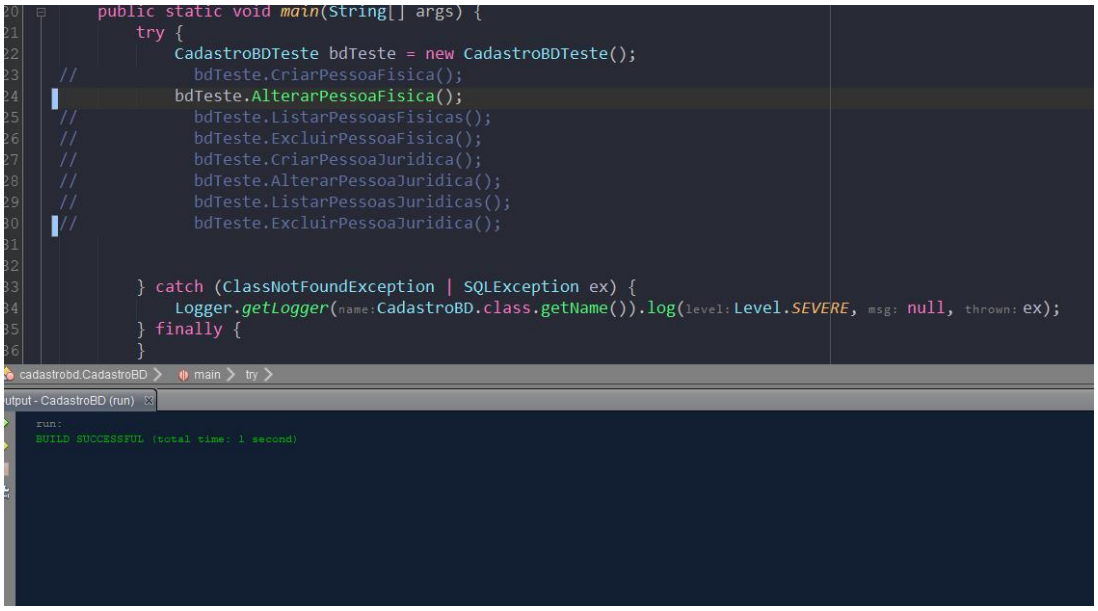
Está sendo utilizado, conforme os print's acima.

6. Criar uma classe principal de testes com o nome **CadastroBDTeste**, efetuando as operações seguintes no método **main**:

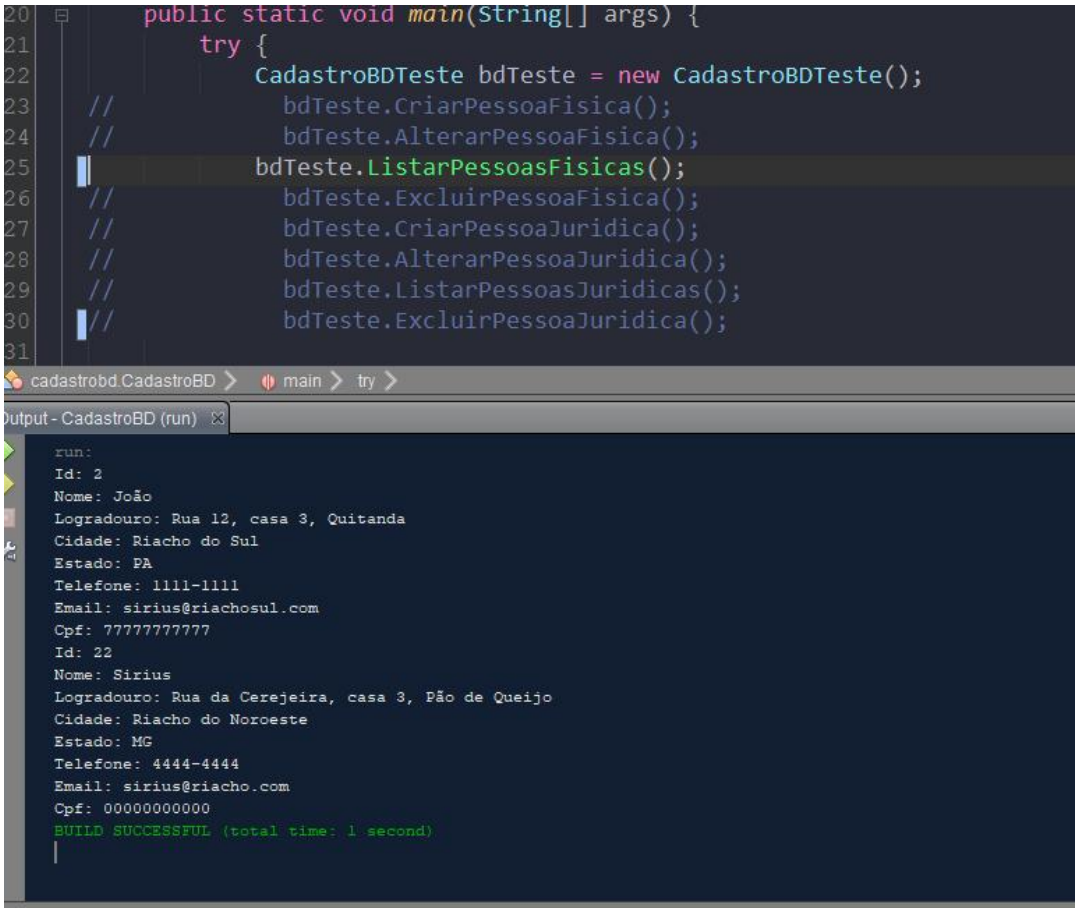
A- Instanciar uma pessoa física e persistir no banco de dados;



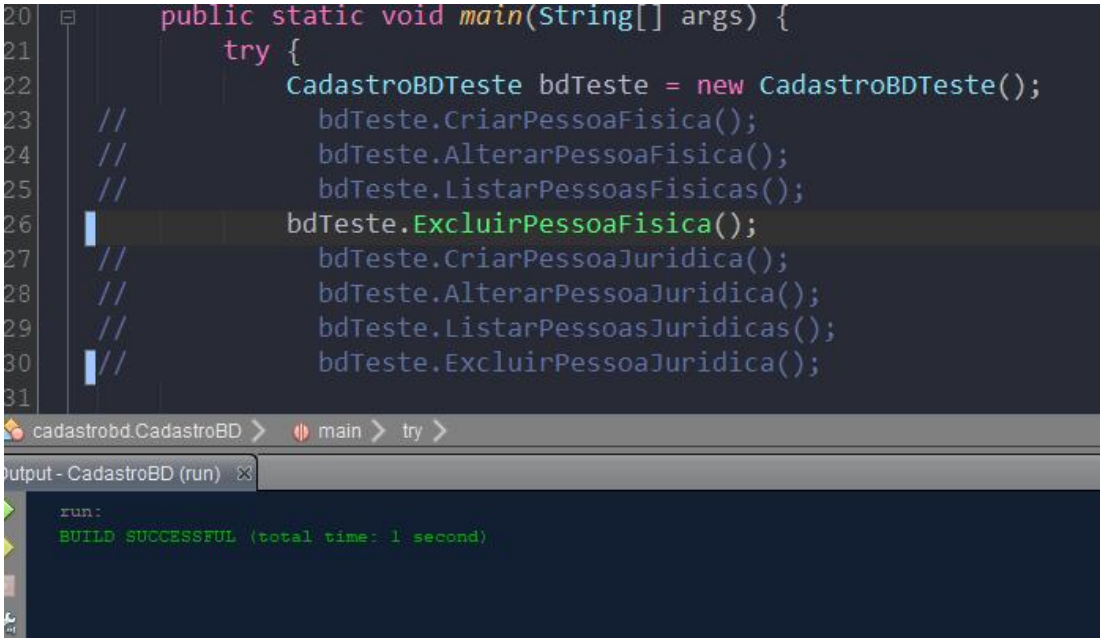
B- Alterar os dados da pessoa física no banco;



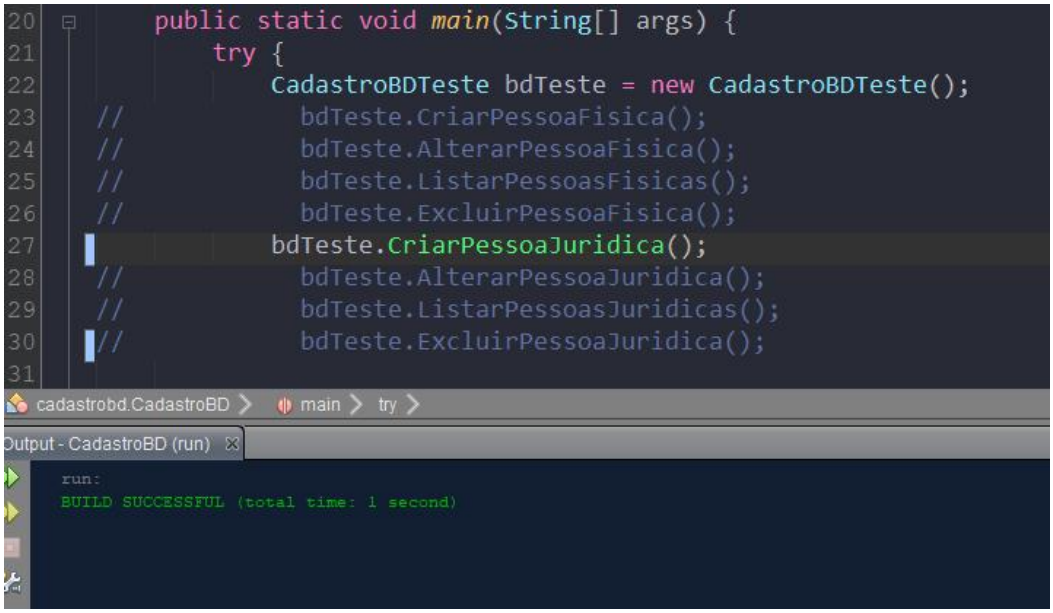
C- Consultar todas as pessoas físicas do banco de dados e listar no console;



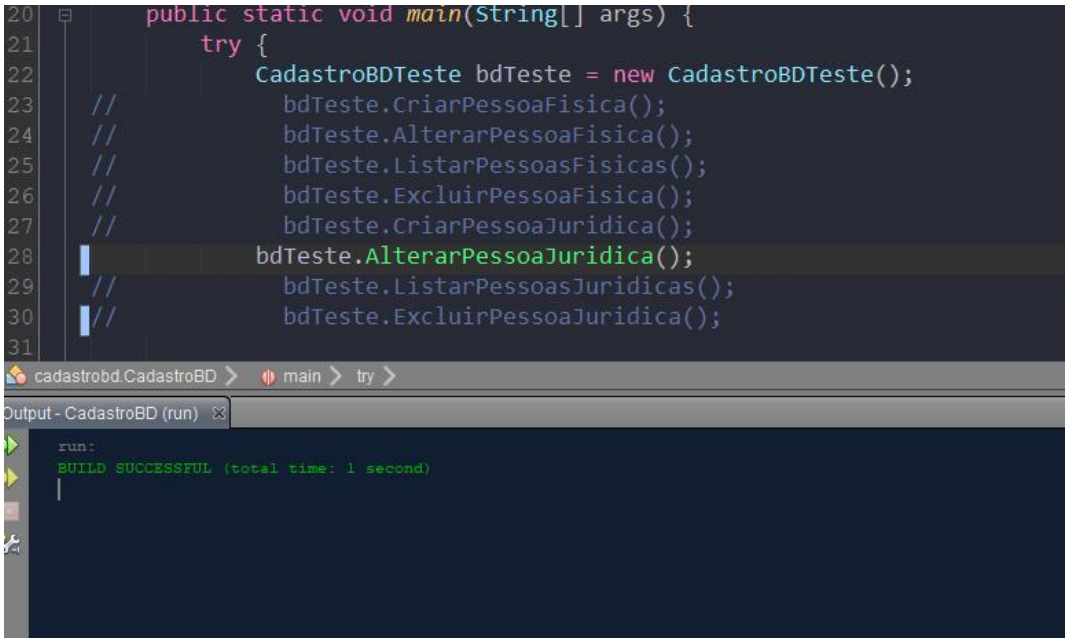
D- Excluir a pessoa física criada anteriormente no banco;



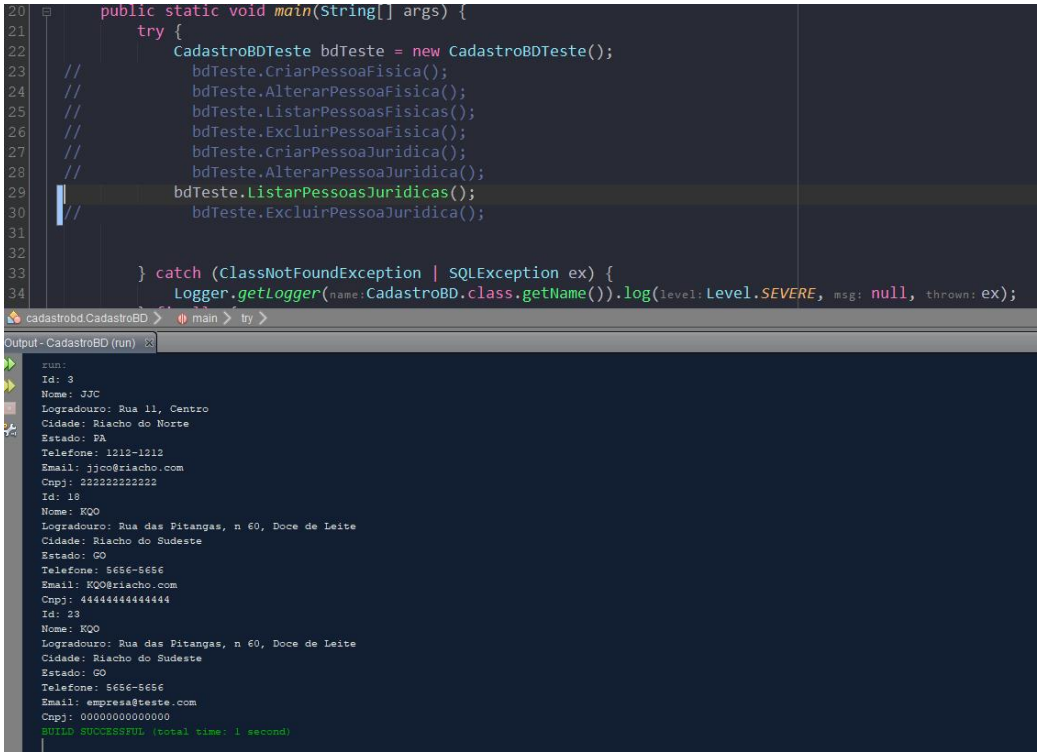
E- Instanciar uma pessoa jurídica e persistir no banco de dados;



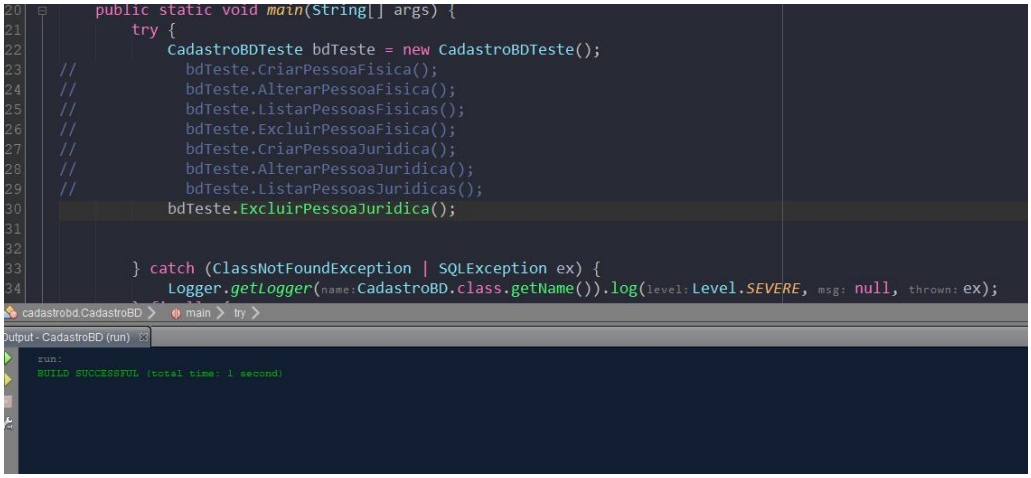
F- Alterar os dados da pessoa jurídica no banco;



G- Consultar todas as pessoas jurídicas e listar no console;



H- Excluir a pessoa jurídica criada anteriormente no banco;



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

O middleware é um software intermediário, que fica entre o frontend e o backend, e possui a função de comunicar ambas camadas. A importância do middleware é que ele é uma tecnologia que garante a comunicação e integração de sistemas, de forma abstrata, permitindo que a mudança de software seja possível

com pouca ou nenhuma alteração de código. Em função disso, há maior independência e portabilidade aos sistemas.

A consulta e manipulação nos bancos de dados relacionais são realizadas por meio de comandos SQL dentro do código Java, através de classes e métodos para ler, inserir, atualizar e excluir dados. Os comandos SQL devem ser o padrão SQL ANSI.

O JDBC (Java Database Connectivity) é um middleware Java que faz a conexão com diversos banco de dados, dentre eles o Oracle, o MySQL, SQL Server e o PostgreSQL.

Dessa forma, quando um software não mais atender os requisitos para determinado sistema, há a opção de migrar para outra plataforma, de forma simples, somente modificando o middleware e fazendo ajustes na conexão.

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

O Statement e o PreparedStatement são interfaces da API do JDBC, e ambos possuem o objetivo de enviar instruções ao banco de dados relacional, no entanto, o fazem de maneira distintas.

O Statement envia a instruções para serem executadas através de strings para o banco, e dessa forma é necessário analisar e o compilar cada instrução de forma separada, acarretando em um maior tempo de execução pelo sistema. Nesse caso, como ele faz a leitura da String de forma literal, sem checar antes, está vulnerável para ataques de SQL Injection.

Em contrapartida, o PreparedStatement pré compila e consequentemente otimiza sua execução. Ele usa parâmetros (placeholders "?") para receber as instruções, e utiliza mecanismos de alteração de valores através dos métodos setString, setInt, entre outros, e por isso preveni ataques de SQL Injection.

Se os comandos forem diferentes entre si, pode fazer sentido utilizar o Statement para enviar a instrução ao banco de dados. Caso contrário, se forem iguais e mudarem somente os valores, o melhor indicado é o PreparedStatement. No entanto, em todos os casos haverá o beneficio de proteção dos ataques de SQL Injection no caso de se recorrer ao PreparedStatement.

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

O DAO (Data Acces Object) é um padrão de desenvolvimento, que organiza o código e o separa a lógica de acesso (acesso ao banco de dados) e os dados da lógica de negócios (métodos). O objetivo é agrupar as instruções SQL em uma única classe, facilitando a compreensão e o reuso dos comandos relacionados ao banco de dados. E também, agilizando a mudança de conexão, se necessário, pois os métodos de interação com o banco de dados estarão agrupados e de fácil localização.

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

A herança é um conceito da programação orientada a objetos, que é aplicado na hierarquia de classes, onde uma classe filha herda características de uma classe pai. No banco de dados estritamente relacional, teoricamente, não há o uso de herança entre as entidades, mas na prática, visualizamos que o uso de chave estrangeira é uma forma de herdar atributos de outra entidade. Portanto, no banco de dados, representamos a herança entre as tabelas utilizando a chave estrangeira para vincular uma entidade a outra, e a entidade recebe os atributos da tabela "pai".

2º Procedimento – Alimentando a Base

Códigos do 2° procedimento estão organizados conforme a sequência do material do trabalho do curso. Segue abaixo:

1. Alterar o método **main** da classe principal do projeto, para implementação do cadastro em modo texto:

A- Apresentar as opções do programa para o usuário, sendo 1 para incluir, 2 para alterar, 3 para excluir, 4 para exibir pelo id, 5 para exibir todos e 0 para finalizar a execução.

```
do {
    System.out.println("========================================");
    System.out.println("1 - Incluir 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("========================================");
```

B- Selecionada a opção **incluir**, escolher o tipo (Física ou Jurídica), receber os dados a partir do teclado e adicionar no banco de dados através da classe DAO correta.

```
113     private static void Incluir() throws SQLException, ClassNotFoundException {
114         char opcaoPessoa = opcaoPessoa();
115 
116         if (opcaoPessoa == 'F' || opcaoPessoa == 'f') {
117             PessoaFisica pf = new PessoaFisica();
118             System.out.println("Inserir dados da Pessoa Fisica: ");
119 
120             System.out.println("Id: ");
121             int id = sc.nextInt();
122             pf.setId(id);
123             sc.nextLine();
124             System.out.println("Nome: ");
125             String nome = sc.nextLine();
126             pf.setNome(nome);
127 
128             System.out.println("Logradouro: ");
129             String logradouro = sc.nextLine();
130             pf.setLogradouro(logradouro);
131 
132             System.out.println("Cidade: ");
133             String cidade = sc.nextLine();
134             pf.setCidade(cidade);
135 
136             System.out.println("Estado: ");
137             String estado = sc.nextLine();
138             pf.setEstado(estado);
139 
140             System.out.println("Telefone: ");
141             String telefone = sc.nextLine();
142             pf.setTelefone(telefone);
143 
144             System.out.println("Email: ");
145             String email = sc.nextLine();
146             pf.setEmail(email);
147 
148             System.out.println("Cpf: ");
149             String cpf = sc.nextLine();
150             pf.setCpf(cpf);
151 
152             pfDao.incluir(pf);
153 
154             System.out.println("========================================");
155             System.out.println("Dados de Pessoa Fisica incluido com sucesso!");
156             System.out.println("========================================");
157         }
```

```java
        if (opcaoPessoa == 'J' || opcaoPessoa == 'j') {
            PessoaJuridica pj = new PessoaJuridica();
            System.out.println("Inserir dados da Pessoa Juridica: ");

            System.out.println("Id: ");
            int id = sc.nextInt();
            pj.setId(id);
            sc.nextLine();
            System.out.println("Nome: ");
            String nome = sc.nextLine();
            pj.setNome(nome);

            System.out.println("Logradouro: ");
            String logradouro = sc.nextLine();
            pj.setLogradouro(logradouro);

            System.out.println("Cidade: ");
            String cidade = sc.nextLine();
            pj.setCidade(cidade);

            System.out.println("Estado: ");
            String estado = sc.nextLine();
            pj.setEstado(estado);

            System.out.println("Telefone: ");
            String telefone = sc.nextLine();
            pj.setTelefone(telefone);

            System.out.println("Email: ");
            String email = sc.nextLine();
            pj.setEmail(email);

            System.out.println("Cnpj: ");
            String cnpj = sc.nextLine();
            pj.setCnpj(cnpj);

            pjDao.incluir(pj);

            System.out.println("========================================");
            System.out.println("Dados de Pessoa Juridica incluido com sucesso!");
            System.out.println("========================================");
        }
    }
```

C- Selecionada a opção **alterar**, escolher o tipo (Física ou Jurídica), receber o **id** a partir do teclado, apresentar os dados atuais, solicitar os novos dados e alterar no banco de dados através do DAO.

```java
    private static void alterar() throws SQLException, ClassNotFoundException {
        char opcaoPessoa = opcaoPessoa();

        if (opcaoPessoa == 'F' || opcaoPessoa == 'f') {

            System.out.println("Qual o ID do cadastro que deseja alterar: ");
            int id = sc.nextInt();
            sc.nextLine();
            PessoaFisica pf = pfDao.getPessoa(id);
            System.out.println("Exibindo dados do ID selecionado para alteracao: ");
            System.out.println(pf.exibir());

            System.out.println("Nome: ");
            String nome = sc.nextLine();
            pf.setNome(nome);

            System.out.println("Logradouro: ");
            String logradouro = sc.nextLine();
            pf.setLogradouro(logradouro);

            System.out.println("Cidade: ");
            String cidade = sc.nextLine();
            pf.setCidade(cidade);

            System.out.println("Estado: ");
            String estado = sc.nextLine();
            pf.setEstado(estado);

            System.out.println("Telefone: ");
            String telefone = sc.nextLine();
            pf.setTelefone(telefone);

            System.out.println("Email: ");
            String email = sc.nextLine();
            pf.setEmail(email);

            System.out.println("Cpf: ");
            String cpf = sc.nextLine();
            pf.setCpf(cpf);

            pfDao.alterar(pf);

            System.out.println("========================================");
            System.out.println("Dados de Pessoa Fisica alterado com sucesso!");
            System.out.println("========================================");
        }
```

```java
        if (opcaoPessoa == 'J' || opcaoPessoa == 'j') {
            System.out.println("Qual o ID do cadastro que deseja alterar: ");
            int id = sc.nextInt();
            sc.nextLine();
            PessoaJuridica pj = pjDao.getPessoa(id);
            System.out.println("Exibindo dados do ID selecionado para alteracao: ");
            System.out.println(pj.exibir());

            System.out.println("Nome: ");
            String nome = sc.nextLine();
            pj.setNome(nome);

            System.out.println("Logradouro: ");
            String logradouro = sc.nextLine();
            pj.setLogradouro(logradouro);

            System.out.println("Cidade: ");
            String cidade = sc.nextLine();
            pj.setCidade(cidade);

            System.out.println("Estado: ");
            String estado = sc.nextLine();
            pj.setEstado(estado);

            System.out.println("Telefone: ");
            String telefone = sc.nextLine();
            pj.setTelefone(telefone);

            System.out.println("Email: ");
            String email = sc.nextLine();
            pj.setEmail(email);

            System.out.println("Cnpj: ");
            String cnpj = sc.nextLine();
            pj.setCnpj(cnpj);

            pjDao.alterar(pj);

            System.out.println("========================================");
            System.out.println("Dados de Pessoa Juridica alterado com sucesso!");
            System.out.println("========================================");
        }
    }
```

D- Selecionada a opção **excluir**, escolher o tipo (Física ou Jurídica), receber o **id** a partir do teclado e remover do banco de dados através do DAO.

```
293     private static void excluir() throws SQLException, ClassNotFoundException {
294         char opcaoPessoa = opcaoPessoa();
295 
296         if (opcaoPessoa == 'F' || opcaoPessoa == 'f') {
297             System.out.println(x: "Qual o ID do cadastro que deseja excluir: ");
298             int id = sc.nextInt();
299             sc.nextLine();
300             PessoaFisica pessoa = pfDao.getPessoa(id);
301             pfDao.excluir(pf: pessoa);
302 
303             System.out.println(x: "========================================");
304             System.out.println(x: "Dados de Pessoa Fisica excluido com sucesso!");
305             System.out.println(x: "========================================");
306         }
307 
308         if (opcaoPessoa == 'J' || opcaoPessoa == 'j') {
309             System.out.println(x: "Qual o ID do cadastro que deseja excluir: ");
310             int id = sc.nextInt();
311             sc.nextLine();
312             PessoaJuridica pessoa = pjDao.getPessoa(id);
313             pjDao.excluir(pj: pessoa);
314 
315             System.out.println(x: "========================================");
316             System.out.println(x: "Dados de Pessoa Juridica excluido com sucesso!");
317             System.out.println(x: "========================================");
318         }
319     }
320 
```

E- Selecionada a opção **obter**, escolher o tipo (Física ou Jurídica), receber o **id** a partir do teclado e apresentar os dados atuais, recuperados do banco através do DAO.

```
321     private static void obter() throws ClassNotFoundException, SQLException {
322         char opcaoPessoa = opcaoPessoa();
323 
324         if (opcaoPessoa == 'F' || opcaoPessoa == 'f') {
325 
326             System.out.println(x: "Qual o ID do cadastro que deseja obter: ");
327             int id = sc.nextInt();
328             sc.nextLine();
329             PessoaFisica pf = pfDao.getPessoa(id);
330             System.out.println(x: "========================================");
331             System.out.println(x: "Exibindo dados de Pessoa Fisica...");
332             System.out.println(x: "========================================");
333 
334             System.out.println(x: pf.exibir());
335         }
336 
337         if (opcaoPessoa == 'J' || opcaoPessoa == 'j') {
338             System.out.println(x: "Qual o ID do cadastro que deseja obter: ");
339             int id = sc.nextInt();
340             sc.nextLine();
341             PessoaJuridica pj = pjDao.getPessoa(id);
342 
343             System.out.println(x: "========================================");
344             System.out.println(x: "Exibindo dados de Pessoa Juridica...");
345             System.out.println(x: "========================================");
346 
347             System.out.println(x: pj.exibir());
348         }
349     }
```

F- Selecionada a opção **obterTodos**, escolher o tipo (Física ou Jurídica) e apresentar os dados de todas as entidades presentes no banco de dados por intermédio do DAO.

```
351     private static void obterTodos() throws ClassNotFoundException, SQLException {
352         char opcaoPessoa = opcaoPessoa();
353 
354         if (opcaoPessoa == 'F' || opcaoPessoa == 'f') {
355             List<PessoaFisica> pfs = pfDao.getPessoas();
356 
357             System.out.println(x: "========================================");
358             System.out.println(x: "Exibindo dados de todas as Pessoas Juridicas...");
359             System.out.println(x: "========================================");
360 
361             pfs.forEach(pf -> System.out.println(x: pf.exibir()));
362         }
363 
364         if (opcaoPessoa == 'J' || opcaoPessoa == 'j') {
365             List<PessoaJuridica> pjs = pjDao.getPessoas();
366 
367             System.out.println(x: "========================================");
368             System.out.println(x: "Exibindo dados de todas as Pessoas Juridicas...");
369             System.out.println(x: "========================================");
370 
371             pjs.forEach(pj -> System.out.println(x: pj.exibir()));
372         }
373     }
```

G- Qualquer exceção que possa ocorrer durante a execução do sistema deverá ser tratada.

```
49              switch (opcao) {
50                  case 1 -> {
51                      try {
52                          Incluir();
53                      } catch (ClassNotFoundException | SQLException e) {
54                          System.out.println("Erro ao incluir pessoa: " + e.getMessage());
55                      }
56                      break;
57                  }
58 
59                  case 2 -> {
60                      try {
61                          alterar();
62                      } catch (ClassNotFoundException | SQLException e) {
63                          System.out.println("Erro ao alterar pessoa: " + e.getMessage());
64                      }
65                      break;
66                  }
67 
68                  case 3 -> {
69                      try {
70                          excluir();
71                      } catch (ClassNotFoundException | SQLException e) {
72                          System.out.println("Erro ao excluir pessoa: " + e.getMessage());
73                      }
74                      break;
75                  }
76                  case 4 -> {
77                      try {
78                          obter();
79                      } catch (ClassNotFoundException | SQLException e) {
80                          System.out.println("Erro ao obter pessoa: " + e.getMessage());
81                      }
82                      break;
83                  }
84 
85                  case 5 -> {
86                      try {
87                          obterTodos();
88                      } catch (ClassNotFoundException | SQLException e) {
89                          System.out.println("Erro ao obter todas as pessoas: " + e.getMessage());
90                      }
91                      break;
92                  }
93 
94                  case 0 ->
95                      System.out.println(x: "Programa finalizado. ");
96                  default ->
97                      System.out.println(x: "Opcao invalida. ");
98 
```

H- Selecionada a opção **sair**, finalizar a execução do sistema.

```
94                  case 0 ->
95                      System.out.println(x: "Programa finalizado. ");
```

2. Testar as funcionalidades do sistema:

A- Efetuar as diversas operações disponibilizadas, tanto para pessoa jurídica quanto para pessoa física.

✓ Incluir Pessoa Física:

```
Output - CadastroBD (run)
run:
========================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
========================================
1
F - Pessoa Fisica | J - Pessoa Juridica
f
Inserir dados da Pessoa Fisica:
Id:
10
Nome:
Sabrina
Logradouro:
Rua das Luas, 19
Cidade:
Ubia
Estado:
PE
Telefone:
34568909
Email:
sabrina@ubia.com
Cpf:
12389067438
========================================
Dados de Pessoa Fisica incluido com sucesso!
========================================
========================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
========================================
```

✓ Incluir Pessoa Jurídica:

```
Output - CadastroBD (run)
run:
========================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
========================================
1
F - Pessoa Fisica | J - Pessoa Juridica
J
Inserir dados da Pessoa Juridica:
Id:
90
Nome:
WDC
Logradouro:
Rua das Azaleias, 56
Cidade:
Uberaba
Estado:
MG
Telefone:
34238567
Email:
WDC@email.com
Cnpj:
12345678901234
========================================
Dados de Pessoa Juridica incluido com sucesso!
========================================
```

✓ Alterar Pessoa Física:

```
==========================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==========================================
2
F - Pessoa Fisica | J - Pessoa Juridica
f
Qual o ID do cadastro que deseja alterar:
25
Id: 25
Nome: Lorena
Logradouro: Rua das Violetas
Cidade: Araguari
Estado: MG
Telefone: 67674545
Email: lorena@email.com
Cpf: 12389012389
Nome:
Lorena
Logradouro:
Rua das Violetas
Cidade:
Araguari
Estado:
MG
Telefone:
34343434
Email:
lorena@email.com
Cpf:
23456789012
==========================================
Dados de Pessoa Fisica alterado com sucesso!
==========================================
```

✓ Alterar Pessoa Jurídica:

```
run:
==========================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==========================================
2
F - Pessoa Fisica | J - Pessoa Juridica
j
Qual o ID do cadastro que deseja alterar:
3
Exibindo dados do ID selecionado para alteracao:
Id: 3
Nome: JJC
Logradouro: Rua 16, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 12121212
Email: JJC@email.com
Cnpj: 222222222222
Nome:
JJC
Logradouro:
Rua 13, Centro
Cidade:
Riacho do Sudeste
Estado:
ES
Telefone:
34347878
Email:
JJC@email.com
Cnpj:
88888888888888
==========================================
Dados de Pessoa Juridica alterado com sucesso!
==========================================
```

✓ Excluir Pessoa Física:

```
==========================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==========================================
3
F - Pessoa Fisica | J - Pessoa Juridica
f
Qual o ID do cadastro que deseja excluir:
26
==========================================
Dados de Pessoa Fisica excluido com sucesso!
==========================================
```

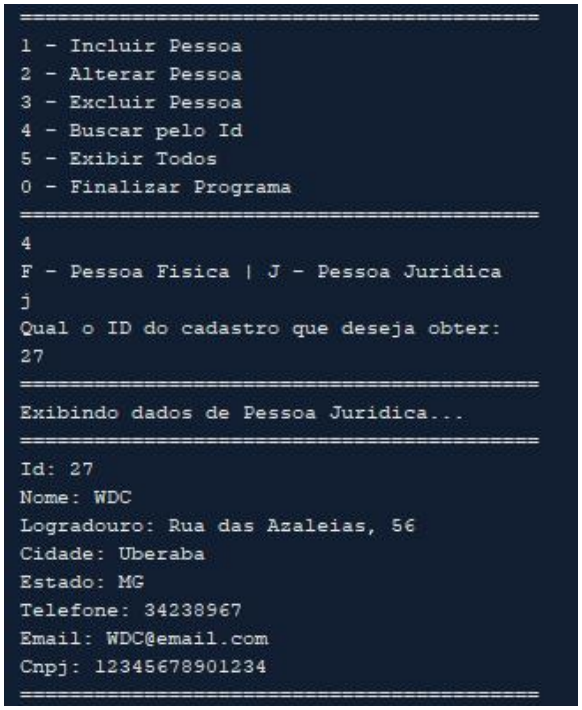
✓ Excluir Pessoa Jurídica:

```
==========================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==========================================
3
F - Pessoa Fisica | J - Pessoa Juridica
j
Qual o ID do cadastro que deseja excluir:
18
==========================================
Dados de Pessoa Juridica excluido com sucesso!
==========================================
```

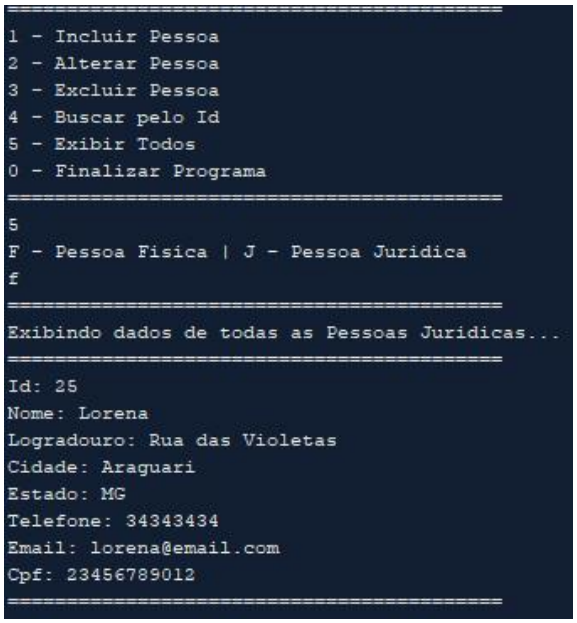
✓ Buscar pelo ID Pessoa Física:

```
==========================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==========================================
4
F - Pessoa Fisica | J - Pessoa Juridica
f
Qual o ID do cadastro que deseja obter:
25
==========================================
Exibindo dados de Pessoa Fisica...
==========================================
Id: 25
Nome: Lorena
Logradouro: Rua das Violetas
Cidade: Araguari
Estado: MG
Telefone: 34343434
Email: lorena@email.com
Cpf: 23456789012
==========================================
```

✓ Buscar pelo ID Pessoa Jurídica:



✓ Exibir todos Pessoa Física:



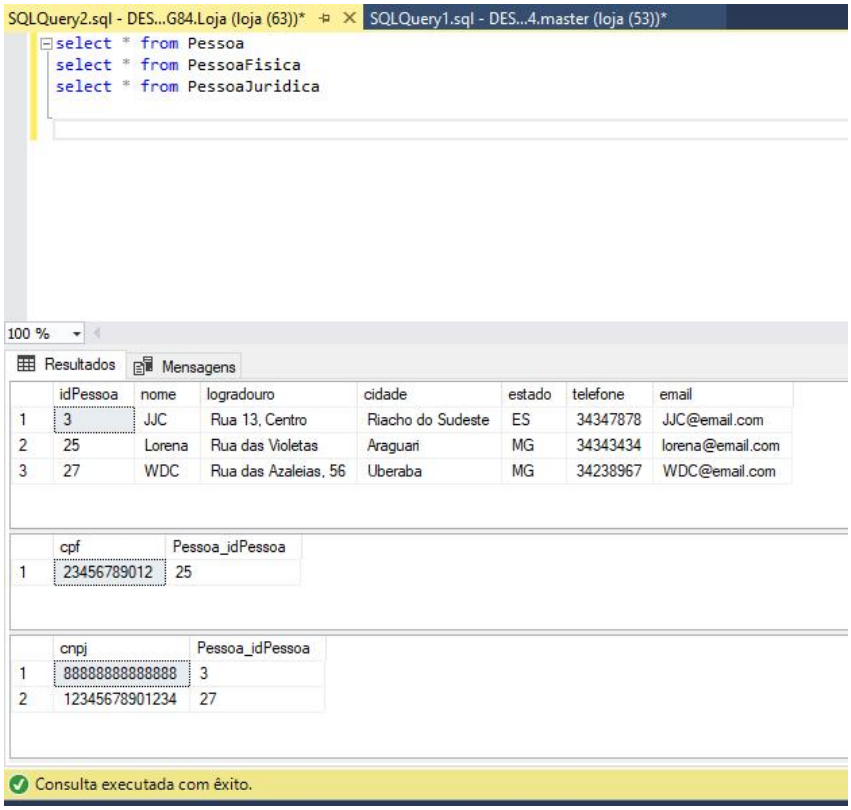
✓ Exibir todos Pessoa Jurídica:

```
==========================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==========================================
5
F - Pessoa Fisica | J - Pessoa Juridica
j
==========================================
Exibindo dados de todas as Pessoas Juridicas...
==========================================
Id: 3
Nome: JJC
Logradouro: Rua 13, Centro
Cidade: Riacho do Sudeste
Estado: ES
Telefone: 34347878
Email: JJC@email.com
Cnpj: 88888888888888
Id: 27
Nome: WDC
Logradouro: Rua das Azaleias, 56
Cidade: Uberaba
Estado: MG
Telefone: 34238967
Email: WDC@email.com
Cnpj: 12345678901234
==========================================
```

✓ Finalizar programa:

```
run:
==========================================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==========================================
0
Programa finalizado.
BUILD SUCCESSFUL (total time: 2 seconds)
```

B- Feitas as operações, verificar os dados no SQL Server, com a utilização da aba Services, divisão Databases, do NetBeans, ou através do SQL Server Management Studio.



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

Os dados persistidos em arquivos, possui todo o gerenciamento do conteúdo do arquivo de responsabilidade do desenvolvedor, portanto, deve haver preocupação com integridade, segurança e disponibilidade das informações. Causando uma maior complexidade ao desenvolvimento, e, em aplicações de grande porte, é totalmente inviável a utilização de arquivos para persistência de dados (por exemplo: não é escalável, difícil manter a integridade e difícil controlar transações).

Enquanto a persistência em banco de dados, são baseados em softwares de mercado, que são mantidos por grandes equipes, que garantem o seu bom funcionamento. Desta forma, a complexidade de gerir os dados é parcialmente abstraída, o desenvolvedor irá se preocupar mais com a forma da utilização dos

dados, e não, em como ele é persistido. Além disso, o banco de dados garatem integridade, segurança, perfomance e escabilidade.

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

Através do uso de lambda, é possível fazer a criação de loops menos verbosos, e também de métodos inline, deixando o código mais enxuto e facilitando a sua leitura. Não há ganhos de performance com a prática, mas a ideia é tornar os desenvolvedores mais produtivos com sua utilização, seguindo a tendência de outras linguagens que adotaram a abordagem, tais como JS, C# e Python.

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

Porque métodos sem a instrução static exigem a instanciação de um objeto da classe em questão para serem chamados. Como o main é um objeto estático, dentro deste contexto, só é possível chamar objetos diretamente que também são estáticos.

Como a questão pede 'sem uso de objeto' essa é a única forma, mas se não fosse o caso, poderiámos instanciar um objeto da classe que contém o main, através da instrução NEW, e com isso será possível chamar métodos não estáticos através do objeto em questão.

Conclusão

O conteúdo desse resumo visa complementar a missão prática do nível 3, que aborda temas relacionados ao conceitos utilizados no desenvolvimento do projeto. As missões práticas do curso em questão vão evoluindo e utilizando ferramentas mais atuais e eficientes, que precisam de menos linhas de código e mais abstrai métodos complexos. Portanto, há sim possibilidade de melhoria, pois os resumos e práticas acompanham o aprendizado do aluno. No entanto, responder questões sobre temas abordados facilita a sedimentação do conhecimento.