



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 3º semestre da instituição de ensino Estácio.

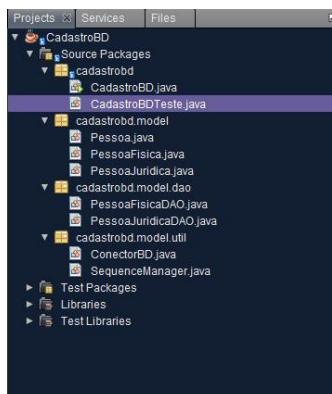
O trabalho é dividido em dois procedimentos, e nos dois procedimentos teremos a mesma organização de conteúdo. Sendo a primeira parte do conteúdo referente a prints dos códigos e seus respectivos resultados, e após, teremos perguntas elaboradas pela instituição de ensino seguidas de respostas.

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 1º 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 **cadastrobd.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;

```
12 public class Pessoa implements Serializable {
13
14     private int id;
15     private String nome;
16     private String logradouro;
17     private String cidade;
18     private String estado;
19     private String telefone;
20     private String email;
21
22     public Pessoa(){
23     }
24
25     public Pessoa(int id, String nome, String logradouro, String cidade,
26         String estado, String telefone, String email){
27         this.id= id;
28         this.nome = nome;
29         this.logradouro = logradouro;
30         this.cidade = cidade;
31         this.estado = estado;
32         this.telefone = telefone;
33         this.email = email;
34     }
35 }
36
37
```

```
135 public String exibir(){
136     return "Id: "
137         + id
138         + "\nNome: "
139         + nome
140         + "\nLogradouro: "
141         + logradouro
142         + "\nCidade: "
143         + cidade
144         + "\nEstado: "
145         + estado
146         + "\nTelefone: "
147         + telefone
148         + "\nEmail: "
149         + email;
150 }
151
152 }
153
154
```

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

```
11 public class PessoaFisica extends Pessoa {
12
13     private String cpf;
14
15     public PessoaFisica(){
16     }
17
18     public PessoaFisica(int id, String nome, String logradouro, String cidade,
19         String estado, String telefone, String email, String cpf){
20         super(id, nome, logradouro, cidade, estado, telefone, email);
21         this.cpf = cpf;
22     }
23
24     /**
25      * @return the cpf
26      */
27     public String getCpf() {
28         return cpf;
29     }
30
31     /**
32      * @param cpf the cpf to set
33      */
34     public void setCpf(String cpf) {
35         this.cpf = cpf;
36     }
37
38     @Override
39     public String exibir(){
40         return super.exibir() + "\nCPF: " + getCpf();
41     }
42 }
43
44
```

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

```
11 public class PessoaJuridica extends Pessoa {
12
13     private String cnpj;
14
15     public PessoaJuridica(){
16     }
17
18     public PessoaJuridica(int id, String nome, String logradouro, String cidade,
19         String estado, String telefone, String email, String cnpj){
20         super(id, nome, logradouro, cidade, estado, telefone, email);
21         this.cnpj = cnpj;
22     }
23
24     /**
25      * @return the cnpj
26      */
27     public String getCnpj() {
28         return cnpj;
29     }
30
31     /**
32      * @param cnpj the cnpj to set
33      */
34     public void setCnpj(String cnpj) {
35         this.cnpj = cnpj;
36     }
37
38     @Override
39     public String exibir(){
40         return super.exibir() + "\nCNPJ: " + cnpj;
41     }
42 }
43
44
```

4. Criar o pacote **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.

```
17 public class ConectorBD {
18
19     private Connection conexao;
20     private PreparedStatement ps;
21     private ResultSet rs;
22
23     public Connection getConnection() throws ClassNotFoundException, SQLException {
24         Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
25         String connectionUrl = "jdbc:sqlserver://localhost\\DESKTOP-8PQGG84:1433;databaseName=Loja;user=loja;password=loja;trustServerCertificate=true";
26         if (conexao == null || conexao.isClosed()) {
27             conexao = DriverManager.getConnection(uri: connectionUrl);
28         }
29         return conexao;
30     }
31
32     public PreparedStatement getPrepared(String sql) throws ClassNotFoundException, SQLException {
33         if (conexao == null || conexao.isClosed()) {
34             getConnection();
35         }
36
37         ps = conexao.prepareStatement(sql);
38         return ps;
39     }
40
41     public ResultSet getSelect(String sql) throws ClassNotFoundException, SQLException {
42         conexao = getConnection();
43         ps = getPrepared(sql);
44         rs = ps.executeQuery();
45         return rs;
46     }
47 }
```

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.

```
47
48 public void close() throws SQLException {
49     if (rs != null && !rs.isClosed()) {
50         rs.close();
51     }
52     if (ps != null && !ps.isClosed()) {
53         ps.close();
54     }
55     if (conexao != null && !conexao.isClosed()) {
56         conexao.close();
57     }
58 }
59
60 }
```

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

```

14 public class SequenceManager {
15
16     public int getValue(ConectorBD bd) throws ClassNotFoundException, SQLException {
17         ResultSet result = bd.getSelect(sql: "SELECT NEXT VALUE FOR idPessoaSequence");
18         if (result.next()) {
19             return result.getInt(1);
20         }
21         result.close();
22         return 0;
23     }
24 }
25

```

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.

```

20 public class PessoaFisicaDAO {
21
22     private final ConectorBD bd;
23     private final SequenceManager sequence;
24
25     public PessoaFisicaDAO(ConectorBD bd) {
26         this.bd = bd;
27         this.sequence = new SequenceManager();
28     }
29
30     public PessoaFisica getPessoa(int id) throws ClassNotFoundException, SQLException {
31         String consulta = "select * from Pessoa "
32             + "inner join PessoaFisica on (Pessoa.idPessoa = PessoaFisica.Pessoa_idPessoa) "
33             + "where idPessoa = ?";
34
35         PreparedStatement ps = bd.getPrepared(sql: consulta);
36         ps.setInt(1, id);
37         ResultSet rs = ps.executeQuery();
38
39         if (rs.next()) {
40             PessoaFisica pessoa = new PessoaFisica();
41             pessoa.setId(rs.getInt(string: "idPessoa"));
42             pessoa.setNome(rs.getString(string: "nome"));
43             pessoa.setLogradouro(rs.getString(string: "logradouro"));
44             pessoa.setCidade(rs.getString(string: "cidade"));
45             pessoa.setEstado(rs.getString(string: "estado"));
46             pessoa.setTelefone(rs.getString(string: "telefone"));
47             pessoa.setEmail(rs.getString(string: "email"));
48             pessoa.setCpf(rs.getString(string: "cpf"));
49             bd.close();
50             return pessoa;
51         }
52
53         bd.close();
54         return null;
55     }
56 }

```

```

57 public List<PessoaFisica> getPessoas() throws ClassNotFoundException, SQLException {
58     ArrayList<PessoaFisica> arraypf = new ArrayList();
59     String consulta = "select * from Pessoa "
60         + "inner join PessoaFisica on (Pessoa.idPessoa = PessoaFisica.Pessoa_idPessoa) ";
61
62     ResultSet rs = bd.getSelect(sql: consulta);
63     while (rs.next()) {
64         PessoaFisica pessoa = new PessoaFisica();
65         pessoa.setId(id: rs.getInt(string: "idPessoa"));
66         pessoa.setNome(nome: rs.getString(string: "nome"));
67         pessoa.setLogradouro(logradouro: rs.getString(string: "logradouro"));
68         pessoa.setCidade(cidade: rs.getString(string: "cidade"));
69         pessoa.setEstado(estado: rs.getString(string: "estado"));
70         pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
71         pessoa.setEmail(email: rs.getString(string: "email"));
72         pessoa.setCpf(cpf: rs.getString(string: "cpf"));
73
74         arraypf.add(e: pessoa);
75     }
76
77     bd.Close();
78     return arraypf;
79 }

```

```

61 public void incluir(PessoaFisica pf) throws SQLException, ClassNotFoundException {
62     String inserir = "insert into Pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email) values \n"
63         + "(?, ?, ?, ?, ?, ?, ?);\n"
64         + "\n"
65         + "insert into PessoaFisica(Pessoa_idPessoa, cpf) values \n"
66         + "(?, ?);";
67
68     int id = sequence.getValue(bd);
69
70     PreparedStatement ps = bd.getPrepared(sql: inserir);
71     ps.setInt(i: 1, ii: id);
72     ps.setString(i: 2, string: pf.getNome());
73     ps.setString(i: 3, string: pf.getLogradouro());
74     ps.setString(i: 4, string: pf.getCidade());
75     ps.setString(i: 5, string: pf.getEstado());
76     ps.setString(i: 6, string: pf.getTelefone());
77     ps.setString(i: 7, string: pf.getEmail());
78     ps.setInt(i: 8, ii: id);
79     ps.setString(i: 9, string: pf.getCpf());
80
81     ps.executeUpdate();
82
83     bd.Close();
84 }

```



```

107 public void alterar(PessoaFisica pf) throws SQLException, ClassNotFoundException {
108     String alterar = "UPDATE Pessoa "
109         + "SET nome = ?, "
110         + "logradouro = ?, "
111         + "cidade = ?, "
112         + "estado = ?, "
113         + "telefone = ?, "
114         + "email = ? "
115         + "WHERE idPessoa = ?;\n"
116         + "UPDATE PessoaFisica "
117         + "SET cpf = ? "
118         + "WHERE Pessoa_idPessoa = ?";
119
120     PreparedStatement ps = bd.getPrepared(sql: alterar);
121     ps.setString(i: 1, string: pf.getNome());
122     ps.setString(i: 2, string: pf.getLogradouro());
123     ps.setString(i: 3, string: pf.getCidade());
124     ps.setString(i: 4, string: pf.getEstado());
125     ps.setString(i: 5, string: pf.getTelefone());
126     ps.setString(i: 6, string: pf.getEmail());
127     ps.setInt(i: 7, i1: pf.getId());
128     ps.setString(i: 8, string: pf.getCpf());
129     ps.setInt(i: 9, i1: pf.getId());
130     ps.executeUpdate();
131
132     bd.close();
133 }
134
135

```

```

136 public void excluir(PessoaFisica pf) throws SQLException, ClassNotFoundException {
137     String delete = "DELETE FROM PessoaFisica WHERE Pessoa_idPessoa = ?;\n"
138         + "DELETE FROM Pessoa WHERE idPessoa = ?";
139     PreparedStatement ps = bd.getPrepared(sql: delete);
140     ps.setInt(i: 1, i1: pf.getId());
141     ps.setInt(i: 2, i1: pf.getId());
142     ps.executeUpdate();
143
144     bd.close();
145 }
146
147 }
148

```

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.

```
21 public class PessoaJuridicaDAO {
22
23     private final ConectorBD bd;
24     private final SequenceManager sequence;
25
26     public PessoaJuridicaDAO(ConectorBD bd) {
27         this.bd = bd;
28         this.sequence = new SequenceManager();
29     }
30
31     public PessoaJuridica getPessoa(int id) throws ClassNotFoundException, SQLException {
32         String consulta = "select * from Pessoa "
33             + "inner join PessoaJuridica on (Pessoa.idPessoa = PessoaJuridica.Pessoa_idPessoa) "
34             + "where Pessoa_idPessoa = ?";
35
36         PreparedStatement ps = bd.getPrepared(sql: consulta);
37         ps.setInt(1, id);
38         ResultSet rs = ps.executeQuery();
39
40         if (rs.next()) {
41             PessoaJuridica pessoa = new PessoaJuridica();
42             pessoa.setId(id: rs.getInt(string: "Pessoa_idPessoa"));
43             pessoa.setNome(nome: rs.getString(string: "nome"));
44             pessoa.setLogradouro(logradouro: rs.getString(string: "logradouro"));
45             pessoa.setCidade(cidade: rs.getString(string: "cidade"));
46             pessoa.setEstado(estado: rs.getString(string: "estado"));
47             pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
48             pessoa.setEmail(email: rs.getString(string: "email"));
49             pessoa.setCnpj(cnpj: rs.getString(string: "cnpj"));
50             bd.Close();
51             return pessoa;
52         }
53
54         bd.Close();
55         return null;
56     }
57 }
```

```
57
58     public List<PessoaJuridica> getPessoas() throws ClassNotFoundException, SQLException {
59         ArrayList<PessoaJuridica> arraypj = new ArrayList();
60         String consulta = "select * from Pessoa "
61             + "inner join PessoaJuridica on (Pessoa.idPessoa = PessoaJuridica.Pessoa_idPessoa) ";
62
63         ResultSet rs = bd.getSelect(sql: consulta);
64         while (rs.next()) {
65             PessoaJuridica pessoa = new PessoaJuridica();
66             pessoa.setId(id: rs.getInt(string: "Pessoa_idPessoa"));
67             pessoa.setNome(nome: rs.getString(string: "nome"));
68             pessoa.setLogradouro(logradouro: rs.getString(string: "logradouro"));
69             pessoa.setCidade(cidade: rs.getString(string: "cidade"));
70             pessoa.setEstado(estado: rs.getString(string: "estado"));
71             pessoa.setTelefone(telefone: rs.getString(string: "telefone"));
72             pessoa.setEmail(email: rs.getString(string: "email"));
73             pessoa.setCnpj(cnpj: rs.getString(string: "cnpj"));
74
75             arraypj.add(e: pessoa);
76         }
77
78         bd.Close();
79         return arraypj;
80     }
81 }
```



```

101
102 public void incluir(PessoaJuridica pj) throws SQLException, ClassNotFoundException {
103     String inserir = "insert into Pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email) values \n"
104         + "(?, ?, ?, ?, ?, ?, ?);\n"
105         + "\n"
106         + "insert into PessoaJuridica(Pessoa_idPessoa, cnpj) values \n"
107         + "(?, ?);";
108
109     int id = sequence.getValue(bd);
110
111     PreparedStatement ps = bd.getPrepared(sql: inserir);
112     ps.setInt(i: 1, i1: id);
113     ps.setString(i: 2, string: pj.getNome());
114     ps.setString(i: 3, string: pj.getLogradouro());
115     ps.setString(i: 4, string: pj.getCidade());
116     ps.setString(i: 5, string: pj.getEstado());
117     ps.setString(i: 6, string: pj.getTelefone());
118     ps.setString(i: 7, string: pj.getEmail());
119     ps.setInt(i: 8, i1: id);
120     ps.setString(i: 9, string: pj.getCnpj());
121
122     ps.executeUpdate();
123
124     bd.Close();
125 }
126

```

```

108 public void alterar(PessoaJuridica pj) throws SQLException, ClassNotFoundException {
109     String alterar = "UPDATE Pessoa "
110         + "SET nome = ?,"
111         + "logradouro = ?,"
112         + "cidade= ?,"
113         + "estado = ?,"
114         + "telefone = ?,"
115         + "email = ? "
116         + "WHERE idPessoa = ?;\n"
117         + "UPDATE PessoaJuridica "
118         + "SET cnpj = ? "
119         + "WHERE Pessoa_idPessoa = ?";
120
121     PreparedStatement ps = bd.getPrepared(sql: alterar);
122     ps.setString(i: 1, string: pj.getNome());
123     ps.setString(i: 2, string: pj.getLogradouro());
124     ps.setString(i: 3, string: pj.getCidade());
125     ps.setString(i: 4, string: pj.getEstado());
126     ps.setString(i: 5, string: pj.getTelefone());
127     ps.setString(i: 6, string: pj.getEmail());
128     ps.setInt(i: 7, i1: pj.getId());
129     ps.setString(i: 8, string: pj.getCnpj());
130     ps.setInt(i: 9, i1: pj.getId());
131     ps.executeUpdate();
132
133     bd.Close();
134 }
135

```

```

136 public void excluir(PessoaJuridica pj) throws SQLException, ClassNotFoundException {
137     String delete = "DELETE FROM PessoaJuridica WHERE Pessoa_idPessoa = ?;\n"
138         + "DELETE FROM Pessoa WHERE idPessoa = ?";
139     PreparedStatement ps = bd.getPrepared(sql: delete);
140     ps.setInt(i: 1, i1: pj.getId());
141     ps.setInt(i: 2, i1: pj.getId());
142     ps.executeUpdate();
143
144     bd.Close();
145 }
146
147 }
148
149

```

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;

```
20 public static void main(String[] args) {
21     try {
22         CadastroBDTeste bdTeste = new CadastroBDTeste();
23         bdTeste.CriarPessoaFisica();
24         // bdTeste.AlterarPessoaFisica();
25         // bdTeste.ListarPessoasFisicas();
26         // bdTeste.ExcluirPessoaFisica();
27         // bdTeste.CriarPessoaJuridica();
28         // bdTeste.AlterarPessoaJuridica();
29         // bdTeste.ListarPessoasJuridicas();
30         // bdTeste.ExcluirPessoaJuridica();
31
32     } catch (ClassNotFoundException | SQLException ex) {
33         Logger.getLogger(name: CadastroBD.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
34     } finally {
35     }
36 }
```

Output - CadastroBD (run)

run:
BUILD SUCCESSFUL (total time: 1 second)

B- Alterar os dados da pessoa física no banco;

```
20 public static void main(String[] args) {
21     try {
22         CadastroBDTeste bdTeste = new CadastroBDTeste();
23         // bdTeste.CriarPessoaFisica();
24         bdTeste.AlterarPessoaFisica();
25         // bdTeste.ListarPessoasFisicas();
26         // bdTeste.ExcluirPessoaFisica();
27         // bdTeste.CriarPessoaJuridica();
28         // bdTeste.AlterarPessoaJuridica();
29         // bdTeste.ListarPessoasJuridicas();
30         // bdTeste.ExcluirPessoaJuridica();
31
32     } catch (ClassNotFoundException | SQLException ex) {
33         Logger.getLogger(name: CadastroBD.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
34     } finally {
35     }
36 }
```

cadastrobd.CadastroBD > main > try >

Output - CadastroBD (run)

run:
BUILD SUCCESSFUL (total time: 1 second)

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

```
20 public static void main(String[] args) {
21     try {
22         CadastroBDTeste bdTeste = new CadastroBDTeste();
23         // bdTeste.CriarPessoaFisica();
24         // bdTeste.AlterarPessoaFisica();
25         bdTeste.ListarPessoasFisicas();
26         // bdTeste.ExcluirPessoaFisica();
27         // bdTeste.CriarPessoaJuridica();
28         // bdTeste.AlterarPessoaJuridica();
29         // bdTeste.ListarPessoasJuridicas();
30         // bdTeste.ExcluirPessoaJuridica();
31     }
}
```

cadastrobd.CadastroBD > main > try >

Output - CadastroBD (run) x

```
run:
Id: 2
Nome: João
Logradouro: Rua 12, casa 3, Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: sirius@riachosul.com
Cpf: 7777777777
Id: 22
Nome: Sirius
Logradouro: Rua da Cerejeira, casa 3, Pão de Queijo
Cidade: Riacho do Noroeste
Estado: MG
Telefone: 4444-4444
Email: sirius@riacho.com
Cpf: 0000000000
BUILD SUCCESSFUL (total time: 1 second)
```

D- Excluir a pessoa física criada anteriormente no banco;

```
20 public static void main(String[] args) {
21     try {
22         CadastroBDTeste bdTeste = new CadastroBDTeste();
23         // bdTeste.CriarPessoaFisica();
24         // bdTeste.AlterarPessoaFisica();
25         // bdTeste.ListarPessoasFisicas();
26         bdTeste.ExcluirPessoaFisica();
27         // bdTeste.CriarPessoaJuridica();
28         // bdTeste.AlterarPessoaJuridica();
29         // bdTeste.ListarPessoasJuridicas();
30         // bdTeste.ExcluirPessoaJuridica();
31     }
}
```

cadastrobd.CadastroBD > main > try >

Output - CadastroBD (run) x

```
run:
BUILD SUCCESSFUL (total time: 1 second)
```

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

```
20 public static void main(String[] args) {
21     try {
22         CadastroBDTeste bdTeste = new CadastroBDTeste();
23         // bdTeste.CriarPessoaFisica();
24         // bdTeste.AlterarPessoaFisica();
25         // bdTeste.ListarPessoasFisicas();
26         // bdTeste.ExcluirPessoaFisica();
27         bdTeste.CriarPessoaJuridica();
28         // bdTeste.AlterarPessoaJuridica();
29         // bdTeste.ListarPessoasJuridicas();
30         // bdTeste.ExcluirPessoaJuridica();
31     }
}
```

cadastrobd.CadastroBD > main > try >

Output - CadastroBD (run)

run:
BUILD SUCCESSFUL (total time: 1 second)

F- Alterar os dados da pessoa jurídica no banco;

```
20 public static void main(String[] args) {
21     try {
22         CadastroBDTeste bdTeste = new CadastroBDTeste();
23         // bdTeste.CriarPessoaFisica();
24         // bdTeste.AlterarPessoaFisica();
25         // bdTeste.ListarPessoasFisicas();
26         // bdTeste.ExcluirPessoaFisica();
27         // bdTeste.CriarPessoaJuridica();
28         bdTeste.AlterarPessoaJuridica();
29         // bdTeste.ListarPessoasJuridicas();
30         // bdTeste.ExcluirPessoaJuridica();
31     }
}
```

cadastrobd.CadastroBD > main > try >

Output - CadastroBD (run)

run:
BUILD SUCCESSFUL (total time: 1 second)

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

```
20 public static void main(String[] args) {
21     try {
22         CadastroBDTeste bdTeste = new CadastroBDTeste();
23         // bdTeste.CriarPessoaFisica();
24         // bdTeste.AlterarPessoaFisica();
25         // bdTeste.ListarPessoasFisicas();
26         // bdTeste.ExcluirPessoaFisica();
27         // bdTeste.CriarPessoaJuridica();
28         // bdTeste.AlterarPessoaJuridica();
29         bdTeste.ListarPessoasJuridicas();
30         // bdTeste.ExcluirPessoaJuridica();
31
32     } catch (ClassNotFoundException | SQLException ex) {
33         Logger.getLogger(name: CadastroBD.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
34     }
}
```

cadastrobd.CadastroBD > main > try >

Output - CadastroBD (run)

```
run:
Id: 3
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjco@riacho.com
Cnpj: 222222222222
Id: 18
Nome: KQO
Logradouro: Rua das Pitangas, n 60, Doce de Leite
Cidade: Riacho do Sudeste
Estado: GO
Telefone: 5656-5656
Email: KQO@riacho.com
Cnpj: 44444444444444
Id: 23
Nome: KQO
Logradouro: Rua das Pitangas, n 60, Doce de Leite
Cidade: Riacho do Sudeste
Estado: GO
Telefone: 5656-5656
Email: empresa@teste.com
Cnpj: 00000000000000
BUILD SUCCESSFUL (total time: 1 second)
```

H- Excluir a pessoa jurídica criada anteriormente no banco;

```
20 public static void main(String[] args) {
21     try {
22         CadastroBDTeste bdTeste = new CadastroBDTeste();
23         // bdTeste.CriarPessoaFisica();
24         // bdTeste.AlterarPessoaFisica();
25         // bdTeste.ListarPessoasFisicas();
26         // bdTeste.ExcluirPessoaFisica();
27         // bdTeste.CriarPessoaJuridica();
28         // bdTeste.AlterarPessoaJuridica();
29         // bdTeste.ListarPessoasJuridicas();
30         bdTeste.ExcluirPessoaJuridica();
31
32     } catch (ClassNotFoundException | SQLException ex) {
33         Logger.getLogger(name: CadastroBD.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
34     }
}
```

cadastrobd.CadastroBD > main > try >

Output - CadastroBD (run)

```
run:
BUILD SUCCESSFUL (total time: 1 second)
```

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 as instruções para serem executadas através de strings para o banco, e dessa forma é necessário analisar e 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 previne 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 benefício 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 Access 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 2º 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(x: "=====");
    System.out.println(x: "1 - Incluir Pessoa");
    System.out.println(x: "2 - Alterar Pessoa");
    System.out.println(x: "3 - Excluir Pessoa");
    System.out.println(x: "4 - Buscar pelo Id");
    System.out.println(x: "5 - Exibir Todos");
    System.out.println(x: "0 - Finalizar Programa");
    System.out.println(x: "=====");
```

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(x: "Inserir dados da Pessoa Fisica: ");
119
120         System.out.println(x: "Id: ");
121         int id = sc.nextInt();
122         pf.setId(id);
123         sc.nextLine();
124         System.out.println(x: "Nome: ");
125         String nome = sc.nextLine();
126         pf.setNome(nome);
127
128         System.out.println(x: "Logradouro: ");
129         String logradouro = sc.nextLine();
130         pf.setLogradouro(logradouro);
131
132         System.out.println(x: "Cidade: ");
133         String cidade = sc.nextLine();
134         pf.setCidade(cidade);
135
136         System.out.println(x: "Estado: ");
137         String estado = sc.nextLine();
138         pf.setEstado(estado);
139
140         System.out.println(x: "Telefone: ");
141         String telefone = sc.nextLine();
142         pf.setTelefone(telefone);
143
144         System.out.println(x: "Email: ");
145         String email = sc.nextLine();
146         pf.setEmail(email);
147
148         System.out.println(x: "Cpf: ");
149         String cpf = sc.nextLine();
150         pf.setCpf(cpf);
151
152         pfDao.incluir(pf);
153
154         System.out.println(x: "=====");
155         System.out.println(x: "Dados de Pessoa Fisica incluido com sucesso!");
156         System.out.println(x: "=====");
157     }
```

```

158         if (opcaoPessoa == 'J' || opcaoPessoa == 'j') {
159             PessoaJuridica pj = new PessoaJuridica();
160             System.out.println(x: "Inserir dados da Pessoa Juridica: ");
161
162             System.out.println(x: "Id: ");
163             int id = sc.nextInt();
164             pj.setId(id);
165             sc.nextLine();
166             System.out.println(x: "Nome: ");
167             String nome = sc.nextLine();
168             pj.setNome(nome);
169
170             System.out.println(x: "Logradouro: ");
171             String logradouro = sc.nextLine();
172             pj.setLogradouro(logradouro);
173
174             System.out.println(x: "Cidade: ");
175             String cidade = sc.nextLine();
176             pj.setCidade(cidade);
177
178             System.out.println(x: "Estado: ");
179             String estado = sc.nextLine();
180             pj.setEstado(estado);
181
182             System.out.println(x: "Telefone: ");
183             String telefone = sc.nextLine();
184             pj.setTelefone(telefone);
185
186             System.out.println(x: "Email: ");
187             String email = sc.nextLine();
188             pj.setEmail(email);
189
190             System.out.println(x: "Cnpj: ");
191             String cnpj = sc.nextLine();
192             pj.setCnpj(cnpj);
193
194             pjDao.incluir(pj);
195
196             System.out.println(x: "=====");
197             System.out.println(x: "Dados de Pessoa Juridica incluido com sucesso!");
198             System.out.println(x: "=====");
199         }
200     }

```

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.

```
201 private static void alterar() throws SQLException, ClassNotFoundException {
202     char opcaoPessoa = opcaoPessoa();
203
204     if (opcaoPessoa == 'F' || opcaoPessoa == 'f') {
205
206         System.out.println("Qual o ID do cadastro que deseja alterar: ");
207         int id = sc.nextInt();
208         sc.nextLine();
209         PessoaFisica pf = pfDao.getPessoa(id);
210         System.out.println("Exibindo dados do ID selecionado para alteracao: ");
211         System.out.println(pf.exibir());
212
213         System.out.println("Nome: ");
214         String nome = sc.nextLine();
215         pf.setNome(nome);
216
217         System.out.println("Logradouro: ");
218         String logradouro = sc.nextLine();
219         pf.setLogradouro(logradouro);
220
221         System.out.println("Cidade: ");
222         String cidade = sc.nextLine();
223         pf.setCidade(cidade);
224
225         System.out.println("Estado: ");
226         String estado = sc.nextLine();
227         pf.setEstado(estado);
228
229         System.out.println("Telefone: ");
230         String telefone = sc.nextLine();
231         pf.setTelefone(telefone);
232
233         System.out.println("Email: ");
234         String email = sc.nextLine();
235         pf.setEmail(email);
236
237         System.out.println("Cpf: ");
238         String cpf = sc.nextLine();
239         pf.setCpf(cpf);
240
241         pfDao.alterar(pf);
242
243         System.out.println("=====");
244         System.out.println("Dados de Pessoa Fisica alterado com sucesso!");
245         System.out.println("=====");
246     }
247 }
```

```
249 if (opcaoPessoa == 'J' || opcaoPessoa == 'j') {
250     System.out.println("Qual o ID do cadastro que deseja alterar: ");
251     int id = sc.nextInt();
252     sc.nextLine();
253     PessoaJuridica pj = pjDao.getPessoa(id);
254     System.out.println("Exibindo dados do ID selecionado para alteracao: ");
255     System.out.println(pj.exibir());
256
257     System.out.println("Nome: ");
258     String nome = sc.nextLine();
259     pj.setNome(nome);
260
261     System.out.println("Logradouro: ");
262     String logradouro = sc.nextLine();
263     pj.setLogradouro(logradouro);
264
265     System.out.println("Cidade: ");
266     String cidade = sc.nextLine();
267     pj.setCidade(cidade);
268
269     System.out.println("Estado: ");
270     String estado = sc.nextLine();
271     pj.setEstado(estado);
272
273     System.out.println("Telefone: ");
274     String telefone = sc.nextLine();
275     pj.setTelefone(telefone);
276
277     System.out.println("Email: ");
278     String email = sc.nextLine();
279     pj.setEmail(email);
280
281     System.out.println("Cnpj: ");
282     String cnpj = sc.nextLine();
283     pj.setCnpj(cnpj);
284
285     pjDao.alterar(pj);
286
287     System.out.println("=====");
288     System.out.println("Dados de Pessoa Juridica alterado com sucesso!");
289     System.out.println("=====");
290 }
291 }
```


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 }
350
```

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
77     case 4 -> {
78         try {
79             obter();
80         } catch (ClassNotFoundException | SQLException e) {
81             System.out.println("Erro ao obter pessoa: " + e.getMessage());
82         }
83         break;
84     }
85
86     case 5 -> {
87         try {
88             obterTodos();
89         } catch (ClassNotFoundException | SQLException e) {
90             System.out.println("Erro ao obter todas as pessoas: " + e.getMessage());
91         }
92         break;
93     }
94
95     case 0 ->
96     default ->
97         System.out.println(x: "Programa finalizado. ");
98         System.out.println(x: "Opcao invalida. ");
99 }
```

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

```
94 | case 0 ->  
95 | System.out.println("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  
Opf:  
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:  
34238967  
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:

```
=====
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
j
Qual o ID do cadastro que deseja obter:
27
=====

Exibindo dados de Pessoa Juridica...
=====

Id: 27
Nome: WDC
Logradouro: Rua das Azaleias, 56
Cidade: Uberaba
Estado: MG
Telefone: 34238967
Email: WDC@email.com
Cnpj: 12345678901234
=====
```

✓ Exibir todos Pessoa Física:

```
=====
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
f
=====

Exibindo dados de todas as Pessoas Juridicas...
=====

Id: 25
Nome: Lorena
Logradouro: Rua das Violetas
Cidade: Araguari
Estado: MG
Telefone: 34343434
Email: lorena@email.com
Cpf: 23456789012
=====
```

✓ 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: 8888888888888888
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.

The screenshot shows the NetBeans IDE interface. At the top, there are two tabs for SQL queries: 'SQLQuery2.sql' and 'SQLQuery1.sql'. The 'SQLQuery2.sql' tab is active, displaying three SQL queries: 'select * from Pessoa', 'select * from PessoaFisica', and 'select * from PessoaJuridica'. Below the queries, there is a section for 'Resultados' (Results) and 'Mensagens' (Messages). The 'Resultados' section shows three tables of data. The first table has columns: idPessoa, nome, logradouro, cidade, estado, telefone, and email. The second table has columns: cpf and Pessoa_idPessoa. The third table has columns: cnpj and Pessoa_idPessoa. At the bottom, a green status bar indicates 'Consulta executada com êxito.' (Query executed successfully).

	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	3	JJC	Rua 13, Centro	Riacho do Sudeste	ES	34347878	JJC@email.com
2	25	Lorena	Rua das Violetas	Araguari	MG	34343434	lorena@email.com
3	27	WDC	Rua das Azaleias, 56	Uberaba	MG	34238967	WDC@email.com

	cpf	Pessoa_idPessoa
1	23456789012	25

	cnpj	Pessoa_idPessoa
1	88888888888888	3
2	12345678901234	27

Consulta executada com êxito.

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 garante integridade, segurança, performance e escalabilidade.

- 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 instanciamento 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, poderíamos 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 aos 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 abstraem 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.