

	<p align="center">SOCIEDADE DE ENSINO SUPERIOR ESTÁCIO DE SÁ POLO SETOR IPIRANGA - APARECIDA DE GOIÂNIA – GO DESENVOLVIMENTO FULL STACK 2023.2 FLEX Relatório da Missão Prática Nível 3 Mundo 3</p>
Aluno:	Teovânio Santos Moreira
Tutora:	Simone Ingrid Monteiro Gama
Repositório GIT	

1º Procedimento | Mapeamento Objeto-Relacional e DAO

1. Título da Prática;

RPG0016 - BackEnd sem banco não tem

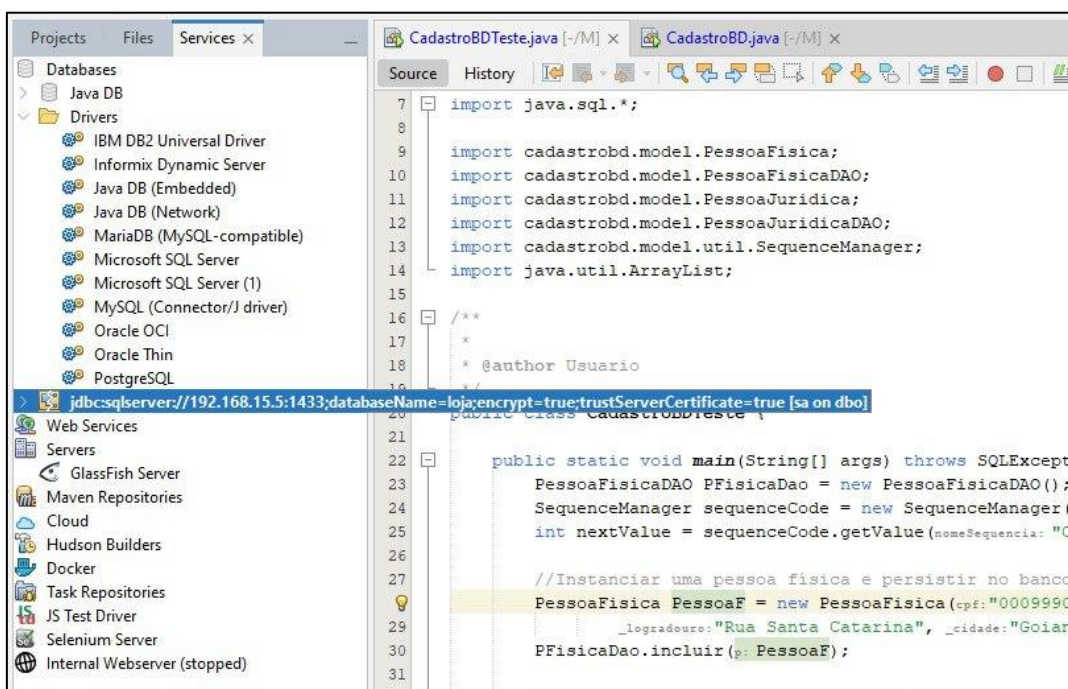
2. Objetivos da prática

- Implementar persistência com base no middleware JDBC.
- Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
- Implementar o mapeamento objeto-relacional em sistemas Java.
- Criar sistemas cadastrais com persistência em banco relacional.
- No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL
- Server na persistência de dados.

3. Todos os códigos solicitados neste roteiro de aula;

https://github.com/Teovanio/m_pratica03

4. Os resultados da execução dos códigos também devem ser apresentados;

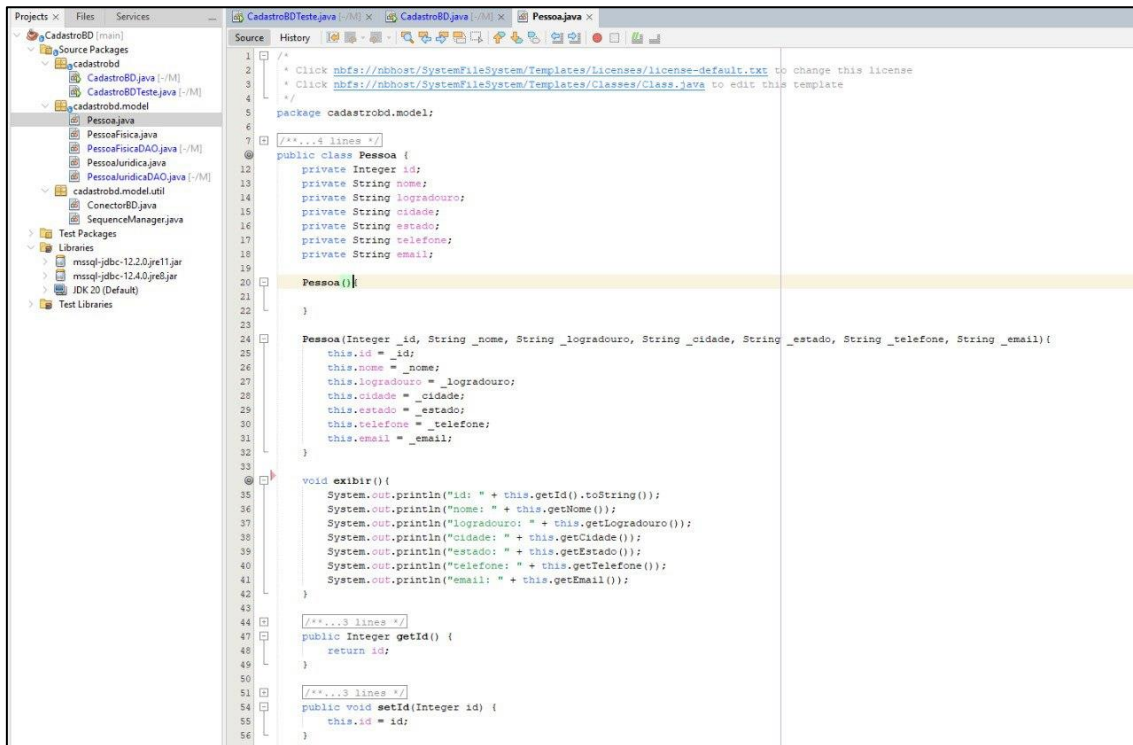


The screenshot shows an IDE with two tabs: 'CadastroBDTeste.java' and 'CadastroBD.java'. The 'Services' tab is active, showing a list of database drivers. A JDBC connection string is highlighted: `jdbc:sqlServer://192.168.15.5:1433;databaseName=loja;encrypt=true;trustServerCertificate=true [sa on dbo]`. The 'CadastroBD.java' tab shows the following code:

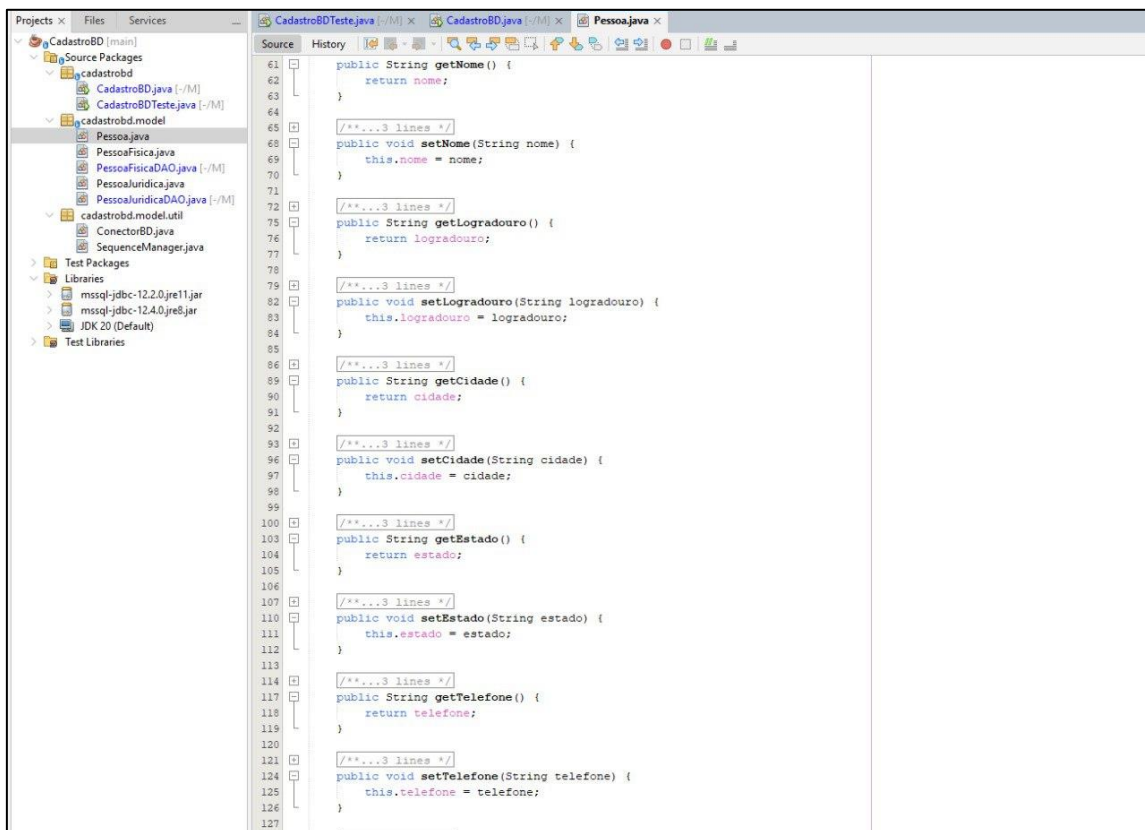
```

7  import java.sql.*;
8
9  import cadastrobd.model.PessoaFisica;
10 import cadastrobd.model.PessoaFisicaDAO;
11 import cadastrobd.model.PessoaJuridica;
12 import cadastrobd.model.PessoaJuridicaDAO;
13 import cadastrobd.model.util.SequenceManager;
14 import java.util.ArrayList;
15
16 /**
17  * @author Usuario
18  */
19 public class CadastroBDTeste {
20
21
22     public static void main(String[] args) throws SQLException {
23         PessoaFisicaDAO PFisicaDao = new PessoaFisicaDAO();
24         SequenceManager sequenceManager = new SequenceManager();
25         int nextValue = sequenceManager.getValue(nomeSequencia: "C");
26
27         //Instanciar uma pessoa fisica e persistir no banco
28         PessoaFisica PessoaF = new PessoaFisica(cpf:"0009990",
29             _logradouro:"Rua Santa Catarina", _cidade:"Goian
30         PFisicaDao.incluir(p: PessoaF);
31
32     }
33 }

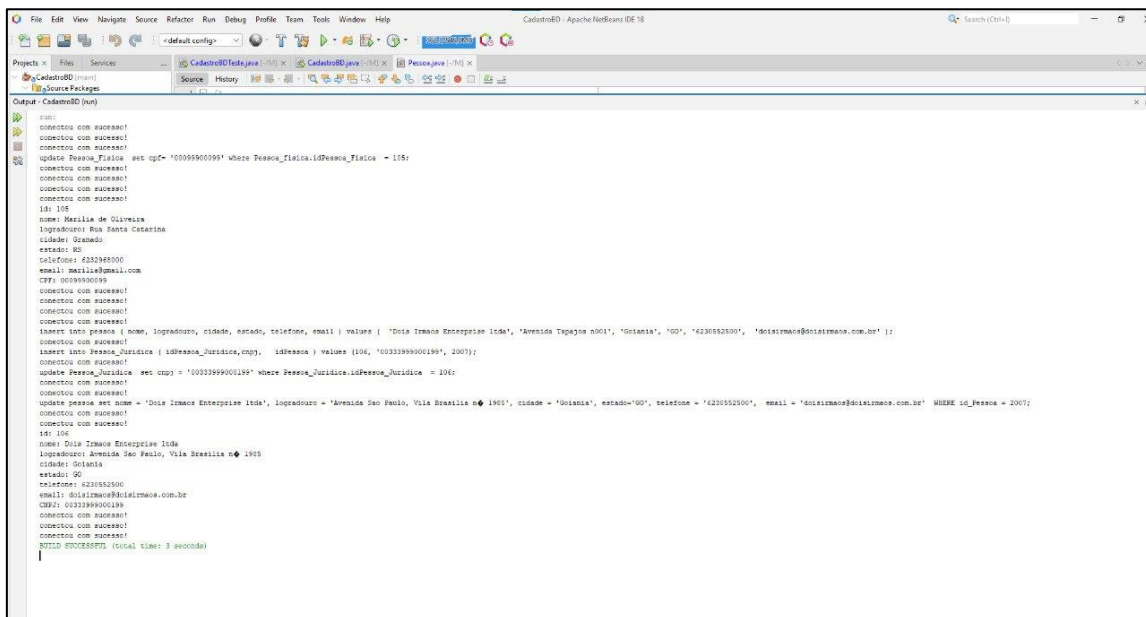
```



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package cadastrobd.model;
6
7  /**...3 lines */
8  @
9  public class Pessoa {
10     private Integer id;
11     private String nome;
12     private String logradouro;
13     private String cidade;
14     private String estado;
15     private String telefone;
16     private String email;
17
18     Pessoa();
19
20     Pessoa(Integer _id, String _nome, String _logradouro, String _cidade, String _estado, String _telefone, String _email) {
21         this.id = _id;
22         this.nome = _nome;
23         this.logradouro = _logradouro;
24         this.cidade = _cidade;
25         this.estado = _estado;
26         this.telefone = _telefone;
27         this.email = _email;
28     }
29
30     void exibir() {
31         System.out.println("Id: " + this.getId().toString());
32         System.out.println("Nome: " + this.getNome());
33         System.out.println("Logradouro: " + this.getLogradouro());
34         System.out.println("Cidade: " + this.getCidade());
35         System.out.println("Estado: " + this.getEstado());
36         System.out.println("Telefone: " + this.getTelefone());
37         System.out.println("Email: " + this.getEmail());
38     }
39
40     /**...3 lines */
41     public Integer getId() {
42         return id;
43     }
44
45     /**...3 lines */
46     public void setId(Integer id) {
47         this.id = id;
48     }
49 }
```



```
61 public String getNome() {
62     return nome;
63 }
64
65 /**...3 lines */
66 public void setNome(String nome) {
67     this.nome = nome;
68 }
69
70 /**...3 lines */
71 public String getLogradouro() {
72     return logradouro;
73 }
74
75 /**...3 lines */
76 public void setLogradouro(String logradouro) {
77     this.logradouro = logradouro;
78 }
79
80 /**...3 lines */
81 public String getCidade() {
82     return cidade;
83 }
84
85 /**...3 lines */
86 public void setCidade(String cidade) {
87     this.cidade = cidade;
88 }
89
90 /**...3 lines */
91 public String getEstado() {
92     return estado;
93 }
94
95 /**...3 lines */
96 public void setEstado(String estado) {
97     this.estado = estado;
98 }
99
100 /**...3 lines */
101 public String getTelefone() {
102     return telefone;
103 }
104
105 /**...3 lines */
106 public void setTelefone(String telefone) {
107     this.telefone = telefone;
108 }
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
```



```
import java.sql.*;

public class CadastroBD {
    public static void main(String[] args) {
        // Conectar ao banco de dados
        Connection conn = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/cadastro_bd", "root", "");
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Inserir dados
        String sql = "INSERT INTO pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES ('Dona Irmae Enterprise Ltda', 'Avenida Tapajoa n001', 'Goiânia', 'GO', '4230552300', 'donairmae@donairmae.com.br')";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.executeUpdate();

        // Atualizar dados
        String sql2 = "UPDATE pessoa SET nome = 'Dona Irmae Enterprise Ltda', logradouro = 'Avenida Sao Paulo, Vila Brasileira n0 1905', cidade = 'Goiânia', estado = 'GO', telefone = '4230552300', email = 'donairmae@donairmae.com.br' WHERE id_pessoa = 2007";
        PreparedStatement stmt2 = conn.prepareStatement(sql2);
        stmt2.executeUpdate();

        // Fechar conexão
        conn.close();
    }
}
```

5. Análise e Conclusão:

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

Os componentes de middleware, como o JDBC (Java Database Connectivity), desempenham um papel crucial no desenvolvimento de aplicativos e sistemas de software. Eles permitem que aplicativos de diferentes tipos e origens se comuniquem e interajam uns com os outros. Eles fornecem uma camada de abstração que permite que os aplicativos se concentrem em suas tarefas específicas, sem ter que se preocupar com os detalhes da comunicação com outros aplicativos ou sistemas.

O JDBC permite que aplicativos Java acessem dados de um banco de dados relacional. Fornecendo uma interface padrão para aplicativos Java se conectarem a bancos de dados e executarem consultas e atualizações SQL. Ele também torna mais fácil para os desenvolvedores de aplicativos Java acessar dados de bancos de dados, independentemente do tipo de banco de dados ou do fabricante do driver JDBC.

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

A principal diferença entre Statement e PreparedStatement é que o PreparedStatement é pré-compilado pelo banco de dados, enquanto o Statement é compilado toda vez que é executado. Isso significa que o PreparedStatement é geralmente mais rápido do que o Statement, especialmente para consultas que são executadas várias vezes.

Além disso, o PreparedStatement permite que você especifique parâmetros para as consultas. Isso pode ajudar a melhorar a segurança e a flexibilidade de suas consultas.

Segue tabela que resume as principais diferenças entre Statement e PreparedStatement:

Característica	Statement	PreparedStatement
Compilação	Compilado toda vez que é executado	Pré-compilado pelo banco de dados
Velocidade	Mais lento	Mais rápido
Parâmetros	Não suporta	Suporta
Segurança	Menos seguro	Mais seguro
Flexibilidade	Menos flexível	Mais flexível

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

O padrão DAO (Data Access Object) é um padrão de projeto de software que separa a lógica de acesso a dados da lógica de negócios.

Aqui estão alguns exemplos de como o padrão DAO pode ser usado para melhorar a manutenibilidade do software:

- Um aplicativo que gerencia uma lista de contatos pode usar o padrão DAO para separar a lógica de acessar e armazenar dados de contatos da lógica de exibir e gerenciar contatos. Isso torna o código mais fácil de entender e manter, pois a lógica de acesso a dados é isolada da lógica de negócios.
- Um aplicativo web que realiza vendas online pode usar o padrão DAO para separar a lógica de acessar e armazenar dados de produtos da lógica de processar pedidos. Isso torna o código mais fácil de testar, pois a lógica de acesso a dados pode ser testada isoladamente da lógica de negócios.
- Um aplicativo de desktop que gerencia um banco de dados de clientes pode usar o padrão DAO para separar a lógica de acessar e armazenar dados de clientes da lógica de exibir e gerenciar dados de clientes. Isso torna o código mais fácil de manter, pois a lógica de acesso a dados é centralizada em uma única classe.

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

A herança em um modelo de banco de dados estritamente relacional pode ser implementada de várias maneiras, dependendo do sistema de gerenciamento de banco de dados (DBMS) que você está usando e das necessidades específicas do seu aplicativo.

Por exemplo, considerando a seguinte hierarquia de classes:

```
class Animal {  
    int id;  
    String nome;  
}  
  
class Cachorro extends Animal {  
    String raça;  
}  
  
class Gato extends Animal {  
    String cor;  
}
```

Para refletir essa hierarquia no banco de dados, podemos usar a seguinte estrutura de tabelas:

```
tabela animal (  
    id int primary key,  
    nome varchar(255)  
);  
  
tabela cachorro (  
    id int primary key,  
    raça varchar(255),  
    foreign key (id) references animal (id)  
);  
  
tabela gato (  
    id int primary key,  
    cor varchar(255),  
    foreign key (id) references animal (id)  
);
```

A tabela `animal` contém os dados comuns a todas as classes da hierarquia, ou seja, o identificador (`id`) e o nome (`nome`). As tabelas `cachorro` e `gato` estendem a tabela `animal` e adicionam atributos específicos para cada classe.

Aqui está um exemplo de como podemos inserir um registro na tabela `animal`:

```
INSERT INTO animal (id, nome) VALUES (1, 'Rex');
```

A herança em um modelo relacional pode ser implementada de várias maneiras. A abordagem de tabelas de herança é uma das mais comuns.

```

1  package cadastrobd.model;
2
3  /**
4   * Click nhfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license.
5   * Click nhfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template.
6   */
7
8  package cadastrobd.model;
9
10 /**
11  *
12  * @author Usuario
13  */
14
15 public class PessoaJuridica extends Pessoa {
16
17     private String cnpj;
18
19     public PessoaJuridica() {
20         super();
21     }
22
23     public PessoaJuridica(String cnpj, Integer _id, String _nome, String _logradouro, String _cidade, String _estado, String _telefone, String _email) {
24         super(_id, _nome, _logradouro, _cidade, _estado, _telefone, _email);
25         this.cnpj = cnpj;
26     }
27
28     @Override
29     public void exibir() {
30         super.exibir();
31         System.out.println("CNPJ: " + this.getCnpj().toString());
32     }
33
34     /**
35      * @return the cnpj
36      */
37     public String getCnpj() {
38         return cnpj;
39     }
40
41     /**
42      * @param cnpj the cnpj to set
43      */
44     public void setCnpj(String cnpj) {
45         this.cnpj = cnpj;
46     }
47
48 }

```

```

1  package cadastrobd.model;
2
3  import java.sql.ResultSet;
4  import java.sql.SQLException;
5  import java.util.ArrayList;
6
7  import cadastrobd.model.util.ConectorBD;
8  import cadastrobd.model.util.SequenceManager;
9
10 /**
11  * Click nhfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license.
12  * Click nhfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template.
13  */
14
15 /**
16  * @author Usuario
17  */
18
19 public class PessoaFisicaDAO {
20
21     ConectorBD cnx = new ConectorBD();
22     SequenceManager sequenceCode = new SequenceManager();
23
24     public PessoaFisica getPessoa(Integer id) throws SQLException {
25         ResultSet rs = cnx.getSelect("select *\n"
26             + "    from Pessoa_Fisica as pf\n"
27             + "    where pf.idPessoa = " + id.toString());
28
29         if (rs != null) {
30             PessoaFisica pf = new PessoaFisica();
31             pf.setId(rs.getInt("idPessoa"));
32             pf.setNome(rs.getString("nome"));
33             pf.setLogradouro(rs.getString("logradouro"));
34             pf.setCidade(rs.getString("cidade"));
35             pf.setEstado(rs.getString("estado"));
36             pf.setTelefone(rs.getString("telefone"));
37             pf.setEmail(rs.getString("email"));
38
39             return pf;
40         }
41
42         return null;
43     }
44
45     public ArrayList<PessoaFisica> getPessoas() throws SQLException {
46         ResultSet rs = cnx.getSelect("select *\n"
47             + "    from Pessoa_Fisica as pf\n"
48             + "    where pf.idPessoa = " + id.toString());
49
50         if (rs != null) {
51             ArrayList<PessoaFisica> list = new ArrayList<PessoaFisica>();
52             while (rs.next()) {
53                 PessoaFisica pf = new PessoaFisica();
54                 pf.setId(rs.getInt("idPessoa"));
55                 pf.setNome(rs.getString("nome"));
56                 pf.setLogradouro(rs.getString("logradouro"));
57                 pf.setCidade(rs.getString("cidade"));
58                 pf.setEstado(rs.getString("estado"));
59                 pf.setTelefone(rs.getString("telefone"));
60                 pf.setEmail(rs.getString("email"));
61
62                 list.add(pf);
63             }
64
65             return list;
66         }
67
68         return null;
69     }
70
71 }

```



```

1 package cadastrobd.model;
2
3 import cadastrobd.model.util.ConnectorBD;
4 import cadastrobd.model.util.SequenceManager;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.util.ArrayList;
8
9
10 /*
11  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
12  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
13  */
14
15
16
17
18
19 public class PessoaJuridicaDAO {
20
21     ConnectorBD cnx = new ConnectorBD();
22
23     public PessoaJuridica getPessoa(Integer id) throws SQLException {
24         ResultSet rs = cnx.getSelect("select\n"
25             + "    Pessoa_Juridica.idPessoa_Juridica as id,\n"
26             + "    Pessoa_Juridica.cnpj,\n"
27             + "    p.nome,\n"
28             + "    p.logradouro,\n"
29             + "    p.cidade,\n"
30             + "    p.estado,\n"
31             + "    p.telefone,\n"
32             + "    p.email\n"
33             + "    from Pessoa_Juridica\n"
34             + "    INNER JOIN Pessoa as p on Pessoa_Juridica.idPessoa = p.id_Pessoa\n"
35             + "    WHERE\n"
36             + "    Pessoa_Juridica.idPessoa = " + id.toString());
37
38         rs.next();
39         PessoaJuridica p = new PessoaJuridica(
40             cnx.getString(setting: "cnpj"),
41             rs.getInt(setting: "id"),
42             rs.getString(setting: "nome"),
43             rs.getString(setting: "logradouro"),
44             rs.getString(setting: "cidade"),
45             rs.getString(setting: "estado"),
46             rs.getString(setting: "telefone"),
47             rs.getString(setting: "email")
48         );
49         p.exibir();
50         cnx.close();
51         return p;
52     }
53 }

```

```

25 public class CadastroBD {
26
27     /**
28      * @param args the command line arguments
29      */
30     public static void main(String[] args) {
31         BufferedReader reader = new BufferedReader(
32             new InputStreamReader(System.in));
33         PessoaFisicaDAO pFisicaDao = new PessoaFisicaDAO();
34         PessoaJuridicaDAO pJuridicaDao = new PessoaJuridicaDAO();
35         SequenceManager sequenceCode = new SequenceManager();
36         int nextValue;
37
38         String opcao = "";
39
40         while (!"0".equals(opcao)) {
41             try {
42                 System.out.println("1 - Incluir");
43                 System.out.println("2 - Alterar");
44                 System.out.println("3 - Excluir");
45                 System.out.println("4 - Exibir pelo ID");
46                 System.out.println("5 - Exibir Todos");
47                 System.out.println("0 - Finalizar Execução");
48
49                 System.out.println("-----");
50
51                 // Reading data using readLine
52                 opcao = reader.readLine();
53
54                 // Printing the read line
55                 System.out.println(opcao);
56
57                 String pessoa;
58
59                 switch (opcao) {
60
61                     case "1" -> {
62                         System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
63                         pessoa = reader.readLine();
64                         nextValue = sequenceCode.getValue(sequence: "CodigoPessoa");
65                         switch (pessoa) {
66                             case "F" -> {
67                                 PessoaFisica p = new PessoaFisica();
68                                 System.out.println("Digite o CPF da pessoa: ");
69                                 p.setCpf(reader.readLine());
70                                 p.setId(nextValue);
71                                 System.out.println("Digite o nome da pessoa: ");
72                                 p.setNome(reader.readLine());
73                                 System.out.println("Digite o endereço da pessoa: ");

```



```

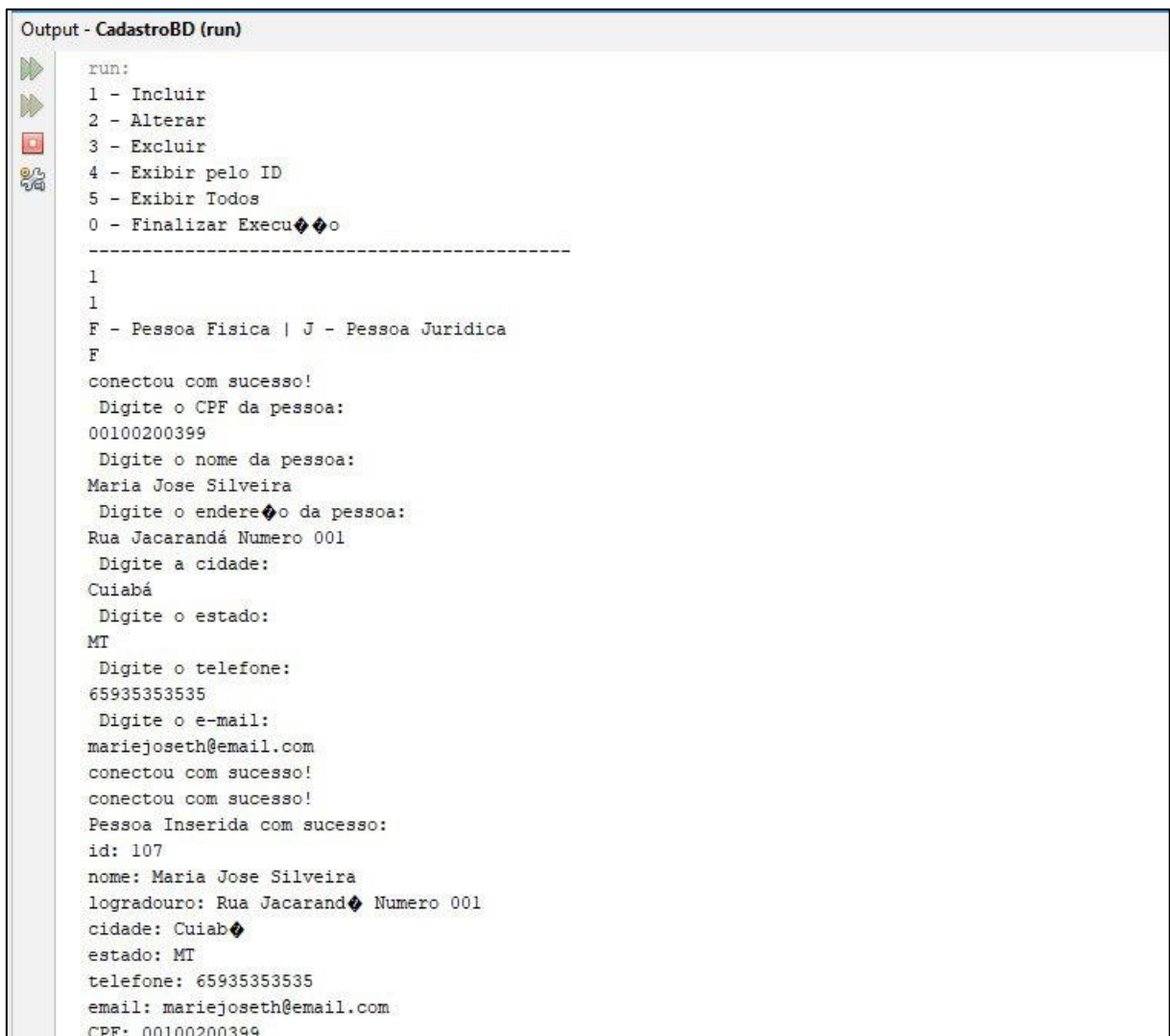
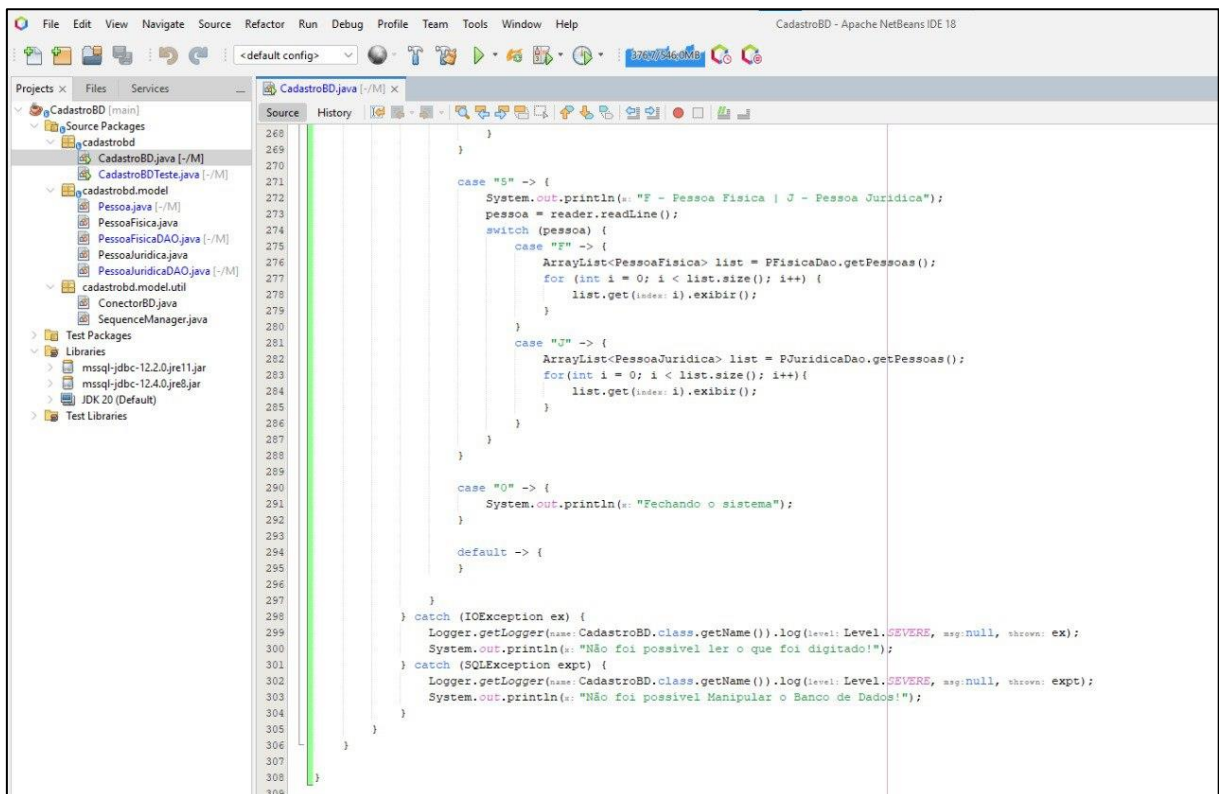
85 p.exibir();
86
87 case "3" -> {
88     PessoaJuridica j = new PessoaJuridica();
89     System.out.println(" Digite o CNPJ da pessoa: ");
90     j.setCnpj(cnpj: reader.readLine());
91     j.setId(id: nextValue());
92     System.out.println(" Digite o nome da pessoa: ");
93     j.setNome(nome: reader.readLine());
94     System.out.println(" Digite o endereço da pessoa: ");
95     j.setLogradouro(logradouro: reader.readLine());
96     System.out.println(" Digite a cidade: ");
97     j.setCidade(cidade: reader.readLine());
98     System.out.println(" Digite o estado: ");
99     j.setEstado(estado: reader.readLine());
100     System.out.println(" Digite o telefone: ");
101     j.setTelefone(telefone: reader.readLine());
102     System.out.println(" Digite o e-mail: ");
103     j.setEmail(email: reader.readLine());
104     System.out.println(" Digite o Nome da pessoa");
105     j.setNome(nome: reader.readLine());
106     PJuridicaDao.incluir(j);
107     System.out.println("Pessoa Inserida com sucesso: ");
108     j.exibir();
109 }
110
111 }
112
113 case "2" -> {
114     System.out.println(" F - Pessoa Fisica | J - Pessoa Juridica");
115     pessoa = reader.readLine();
116     switch (pessoa) {
117         case "F" -> {
118             System.out.println(" Digite o Id que deseja alterar: ");
119             int id = Integer.parseInt(reader.readLine());
120             PessoaFisica pf = PFisicaDao.getPessoa(id);
121
122             System.out.println("O CPF Antigo da Pessoa é: " + pf.getCpf() +
123 " Digite um novo CPF da pessoa ou deixe em branco para não alterar: ");
124             String cpf = reader.readLine();
125             if(!cpf.isBlank() && !cpf.isEmpty()){
126                 pf.setCpf(cpf);
127             }
128
129             System.out.println("O Nome Antigo da Pessoa é: " + pf.getNome() +
130 " Digite o novo Nome da pessoa ou deixe em branco para não alterar: ");
131             String nome = reader.readLine();
132             if(!nome.isBlank() && !nome.isEmpty()){
133                 pf.setNome(nome);

```

```

232
233 case "3" -> {
234     System.out.println(" F - Pessoa Fisica | J - Pessoa Juridica");
235     pessoa = reader.readLine();
236     switch (pessoa) {
237         case "F" -> {
238             System.out.println(" Digite o Id da pessoa");
239             int id = Integer.parseInt(reader.readLine());
240             PFisicaDao.excluir(id);
241         }
242         case "J" -> {
243             System.out.println(" Digite o Id da pessoa");
244             int id = Integer.parseInt(reader.readLine());
245             PJuridicaDao.excluir(id);
246         }
247     }
248 }
249
250 case "4" -> {
251     System.out.println(" F - Pessoa Fisica | J - Pessoa Juridica");
252     pessoa = reader.readLine();
253     switch (pessoa) {
254         case "F" -> {
255             System.out.println(" Digite o Id da pessoa");
256             int id = Integer.parseInt(reader.readLine());
257             PessoaFisica pf = PFisicaDao.getPessoa(id);
258             pf.exibir();
259         }
260         case "J" -> {
261             System.out.println(" Digite o Id da pessoa");
262             int id = Integer.parseInt(reader.readLine());
263             PessoaJuridica pj = PJuridicaDao.getPessoa(id);
264             pj.exibir();
265         }
266     }
267 }
268
269 }

```



```

Output - CadastroBD (run)
1
1
F - Pessoa Fisica | J - Pessoa Juridica
J
conectou com sucesso!
Digite o CNPJ da pessoa:
11001002000199
Digite o nome da pessoa:
Marie Cosmeticos Ltda
Digite o endereço da pessoa:
Avenida Mangalo Numero 321
Digite a cidade:
Cuiaba
Digite o estado:
MT
Digite o telefone:
6535353535
Digite o e-mail:
mariecosmeticos@email.com.br
Digite o Nome da pessoa
Marie Cosmeticos Ltda
insert into pessoa ( nome, logradouro, cidade, estado, telefone, email ) values ( 'Marie Cosmeticos Ltda', 'Avenida Mangalo Numero 321', 'Cuiaba', 'MT', '6535353535', 'mariecosmeticos@email.com.br' );
conectou com sucesso!
insert into Pessoa_Juridica ( idPessoa_Juridica,cnpj, idPessoa ) values (108, '11001002000199', 2009);
conectou com sucesso!
Pessoa Inserida com sucesso:
id: 108
nome: Marie Cosmeticos Ltda
logradouro: Avenida Mangalo Numero 321
cidade: Cuiaba
estado: MT
telefone: 6535353535
email: mariecosmeticos@email.com.br
CNPJ: 11001002000199
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir Todos
0 - Finalizar Execu

```

```

Output - CadastroBD (run)
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir Todos
0 - Finalizar Execu
-----
2
2
F - Pessoa Fisica | J - Pessoa Juridica
F
Digite o Id que deseja alterar:
107
conectou com sucesso!
id: 107
nome: Maria Jose Silveira
logradouro: Rua Jacarand Numero 001
cidade: Cuiab
estado: MT
telefone: 65935353535
email: mariejoseth@email.com
CPF: 00100200399
O CPF Antigo da Pessoa : 00100200399 Digite um novo CPF da pessoa ou deixe em branco para n alterar:

O Nome Antigo da Pessoa : Maria Jose Silveira Digite o novo Nome da pessoa ou deixe em branco para n alterar:
Maria Jose Junqueira
O Logradouro Antigo da Pessoa : Rua Jacarand Numero 001 Digite o novo Logradouro da pessoa ou deixe em branco para n alterar:

A Cidade Antigo da Pessoa : Cuiab Digite a nova Cidade da pessoa ou deixe em branco para n alterar:
Goiania
O Estado Antigo da Pessoa : MT Digite o novo Estado da pessoa ou deixe em branco para n alterar:
GO
O Telefone Antigo da Pessoa : 65935353535 Digite o novo Telefone da pessoa ou deixe em branco para n alterar:

O Email Antigo da Pessoa : mariejoseth@email.com Digite um novo Email da pessoa ou deixe em branco para n alterar:

update Pessoa_Fisica set cpf= '00100200399' where Pessoa_fisica.idPessoa_Fisica = 107;

```

```

1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir Todos
0 - Finalizar Execu
-----
3
3
F - Pessoa Fisica | J - Pessoa Juridica
J
Digite o Id da pessoa
108
conectou com sucesso!
conectou com sucesso!
conectou com sucesso!

```

```
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir Todos
0 - Finalizar Execu♦♦o

-----

3
3
F - Pessoa Fisica | J - Pessoa Juridica
J
  Digite o Id da pessoa
108
conectou com sucesso!
conectou com sucesso!
conectou com sucesso!
```

```
1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir Todos
0 - Finalizar Execu♦♦o

-----

4
4
F - Pessoa Fisica | J - Pessoa Juridica
F
  Digite o Id da pessoa
107
conectou com sucesso!
id: 107
nome: Maria Jose Junqueira
logradouro: Rua Jacarand♦ Numero 001
cidade: Goiania
estado: GO
telefone: 65935353535
email: mariejoseth@email.com
CPF: 00100200399
id: 107
nome: Maria Jose Junqueira
logradouro: Rua Jacarand♦ Numero 001
cidade: Goiania
estado: GO
telefone: 65935353535
email: mariejoseth@email.com
CPF: 00100200399
```

```

1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir Todos
0 - Finalizar Execu♦♦o
-----
5
5
F - Pessoa Fisica | J - Pessoa Juridica
F
conectou com sucesso!
id: 107
nome: Maria Jose Junqueira
logradouro: Rua Jacarand♦ Numero 001
cidade: Goiania
estado: GO
telefone: 65935353535
email: mariejoseth@email.com
CPF: 00100200399

```

```

1 - Incluir
2 - Alterar
3 - Excluir
4 - Exibir pelo ID
5 - Exibir Todos
0 - Finalizar Execu♦♦o
-----
0
0
Fechando o sistema
BUILD SUCCESSFUL (total time: 16 minutes 29 seconds)

```

5. Análise e Conclusão:

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

A persistência em arquivo é um método mais simples e direto de armazenamento de dados. Os dados são armazenados em um arquivo no disco rígido, e o acesso aos dados é feito diretamente pelo aplicativo. Isso pode ser uma vantagem em termos de desempenho, pois o aplicativo não precisa passar por um intermediário, como um banco de dados, para acessar os dados.

No entanto, a persistência em arquivo também tem algumas desvantagens. Por exemplo, os arquivos podem ser facilmente corrompidos, o que pode levar à perda de dados. Além disso, não é adequada para aplicativos que precisam lidar com grandes quantidades de dados ou que precisam acessar os dados de forma rápida e eficiente.

A persistência em banco de dados é um método mais complexo de armazenamento de dados, mas também é mais robusto e confiável. Os dados são armazenados em uma estrutura de banco de dados, que oferece uma série de recursos para garantir a integridade e a segurança destes.

Por exemplo, os bancos de dados oferecem recursos para evitar a corrupção de dados, para garantir a consistência dos dados e para proteger os dados de acesso não autorizado.

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

O operador lambda é uma expressão anônima que pode ser usada para representar uma função. Ele foi introduzido no Java 8 e desde então tem sido usado para simplificar a codificação em uma variedade de cenários.

Um dos cenários em que o operador lambda pode ser usado para simplificar a codificação é a impressão dos valores contidos nas entidades de forma concisa e fácil de ler.

Isso acontece pois ele usa uma expressão lambda para iterar sobre os valores da entidade. Antes do Java 8, era necessário escrever código explícito para iterar sobre as entidades e imprimir seus valores. Isso podia ser um processo repetitivo e propenso a erros. Com isso ele permite que os desenvolvedores se concentrem na lógica da impressão, em vez de na implementação de código repetitivo.

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

Por conta de o método main não ter um objeto associado a ele, precisamos marcar como static os métodos acionados diretamente pelo método main. Por ser um método estático, ele não está associado a um objeto específico da classe.

Quando um método é chamado a partir de um objeto, o método tem acesso a todos os membros deste, incluindo seus campos e métodos. No entanto, quando um método é chamado sem o uso de um objeto, o método não tem acesso a nenhum membro de nenhum objeto.

Um método static tem acesso a todos os membros da classe, incluindo seus campos e métodos. Dessa forma permitimos que um método seja chamado sem o uso de um objeto.