



**ALUNO: FILIPE MACIEL DE SOUZA ANDRADE**

**TURMA: 9001**

**CURSO: DESENVOLVIMENTO FULL STACK**

**DISCIPLINA: INICIANDO O CAMINHO PELO JAVA**

**MATRÍCULA: 2023.04.65842-1**

**POLO PORTO ALEGRE – RS**

**3º PERÍODO**

## **Missão Prática | Nível 3 | Mundo 3**

### **Objetivos da Prática**

1. Implementar persistência com base no middleware JDBC.
2. Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
3. Implementar o mapeamento objeto-relacional em sistemas Java.
4. Criar sistemas cadastrais com persistência em banco relacional.

### **1º Procedimento | Mapeamento Objeto-Relacional e DAO**

```
1 package model;
2
3 public class Pessoa {
4     protected int id;
5     protected String nome;
6     protected String logradouro;
7     protected String cidade;
8     protected String estado;
9     protected String telefone;
10    protected String email;
11
12    public Pessoa() {
13    }
14
15    public Pessoa(int id, String nome, String logradouro, String cidade,
16                  String estado, String telefone, String email) {
17        this.id = id;
18        this.nome = nome;
19        this.logradouro = logradouro;
20        this.cidade = cidade;
21        this.estado = estado;
22        this.telefone = telefone;
23        this.email = email;
24    }
25
26    public int getId() {
27        return id;
28    }
29
30    public void setId(int id) {
31        this.id = id;
32    }
33
34    public String getNome() {
35        return nome;
36    }
37
38    public void setNome(String nome) {
39        this.nome = nome;
40    }
41
42    public String getLogradouro() {
43        return logradouro;
44    }
45
46    public void setLogradouro(String logradouro) {
47        this.logradouro = logradouro;
48    }
49
50    public String getCidade() {
51        return cidade;
52    }
53
54    public void setCidade(String cidade) {
55        this.cidade = cidade;
56    }
57
58    public String getEstado() {
59        return estado;
60    }
61
62    public void setEstado(String estado) {
63        this.estado = estado;
64    }
65
66    public String getTelefone() {
67        return telefone;
68    }
69
70    public void setTelefone(String telefone) {
71        this.telefone = telefone;
72    }
73
74    public String getEmail() {
75        return email;
76    }
77
78    public void setEmail(String email) {
79        this.email = email;
80    }
81
82    public void exibir() {
83        System.out.println("-----");
84        System.out.println("ID: " + id);
85        System.out.println("Nome: " + nome);
86        System.out.println("Logradouro: " + logradouro);
87        System.out.println("Cidade: " + cidade);
88        System.out.println("Estado: " + estado);
89        System.out.println("Telefone: " + telefone);
90        System.out.println("Email: " + email);
91        System.out.println("-----");
92    }
93 }
94
```

```
1 package model;
2
3 public class PessoaFisica extends Pessoa {
4     protected String cpf;
5
6     public PessoaFisica() {
7
8     }
9
10    public PessoaFisica(int id, String nome, String logradouro, String cidade,
11                        String estado, String telefone, String email, String cpf) {
12        super(id, nome, logradouro, cidade, estado, telefone, email);
13        this.cpf = cpf;
14    }
15
16    public String getCpf() {
17        return cpf;
18    }
19
20    public void setCpf(String cpf) {
21        this.cpf = cpf;
22    }
23
24    @Override
25    public void exibir() {
26        super.exibir();
27        System.out.println("CPF: " + cpf);
28    }
29 }
30 }
```

```
1 package model;
2
3 public class PessoaJuridica extends Pessoa {
4     protected String cnpj;
5
6     public PessoaJuridica() {
7
8
9         public PessoaJuridica(int id, String nome, String logradouro, String cidade,
10             String estado, String telefone, String email, String cnpj) {
11                 super(id, nome, logradouro, cidade, estado, telefone, email);
12                 this.cnpj = cnpj;
13             }
14
15         public String getCnpj() {
16             return cnpj;
17         }
18
19         public void setCnpj(String cnpj) {
20             this.cnpj = cnpj;
21         }
22
23         @Override
24         public void exibir() {
25             super.exibir();
26             System.out.println("CPF: " + cnpj);
27         }
28     }
29 }
```

```

1 package model;
2
3 import model.util.ConektorBD;
4 import java.sql.Connection;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 public class PessoaFisicaDAO {
12     public PessoaFisica getPessoa(int id) throws SQLException {
13         String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
14             "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaFisica.cpf " +
15             "FROM Pessoa " +
16             "JOIN PessoaFisica ON Pessoa.idPessoa = PessoaFisica.idPessoa " +
17             "WHERE Pessoa.idPessoa = ?";
18         try (Connection conn = ConektorBD.getConnection();
19              PreparedStatement stmt = ConektorBD.getPreparedStatement(sql)) {
20             stmt.setInt(1, id);
21             try (ResultSet rs = ConektorBD.getSelect(stmt)) {
22                 if (rs.next()) {
23                     PessoaFisica pessoa = new PessoaFisica();
24                     pessoa.setId(rs.getInt("idPessoa"));
25                     pessoa.setNome(rs.getString("nome"));
26                     pessoa.setLogradouro(rs.getString("logradouro"));
27                     pessoa.setCidade(rs.getString("cidade"));
28                     pessoa.setEstado(rs.getString("estado"));
29                     pessoa.setTelefone(rs.getString("telefone"));
30                     pessoa.setEmail(rs.getString("email"));
31                     pessoa.setCpf(rs.getString("cpf"));
32                     return pessoa;
33                 }
34             }
35         }
36         return null;
37     }
38
39     public List<PessoaFisica> getPessoas() throws SQLException {
40         List<PessoaFisica> pessoas = new ArrayList<>();
41         String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
42             "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaFisica.cpf " +
43             "FROM Pessoa " +
44             "JOIN PessoaFisica ON Pessoa.idPessoa = PessoaFisica.idPessoa";
45         try (Connection conn = ConektorBD.getConnection();
46              PreparedStatement stmt = ConektorBD.getPreparedStatement(sql);
47              ResultSet rs = ConektorBD.getSelect(stmt)) {
48             while (rs.next()) {
49                 PessoaFisica pessoa = new PessoaFisica();
50                 pessoa.setId(rs.getInt("idPessoa"));
51                 pessoa.setNome(rs.getString("nome"));
52                 pessoa.setLogradouro(rs.getString("logradouro"));
53                 pessoa.setCidade(rs.getString("cidade"));
54                 pessoa.setEstado(rs.getString("estado"));
55                 pessoa.setTelefone(rs.getString("telefone"));
56                 pessoa.setEmail(rs.getString("email"));
57                 pessoa.setCpf(rs.getString("cpf"));
58                 pessoas.add(pessoa);
59             }
60         }
61         return pessoas;
62     }
63
64     public void incluir(PessoaFisica pessoa) throws SQLException {
65         String sqlInsertPessoa = "INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade," +
66             "estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
67         String sqlInsertPessoaFisica = "INSERT INTO PessoaFisica (idPessoa, cpf)" +
68             "VALUES (?, ?)";
69         try (Connection conn = ConektorBD.getConnection();
70              PreparedStatement stmtInsertPessoa = conn.prepareStatement(sqlInsertPessoa);
71              PreparedStatement stmtInsertPessoaFisica = conn.prepareStatement(sqlInsertPessoaFisica)) {
72
73             stmtInsertPessoa.setInt(1, pessoa.getId());
74             stmtInsertPessoa.setString(2, pessoa.getNome());
75             stmtInsertPessoa.setString(3, pessoa.getLogradouro());
76             stmtInsertPessoa.setString(4, pessoa.getCidade());
77             stmtInsertPessoa.setString(5, pessoa.getEstado());
78             stmtInsertPessoa.setString(6, pessoa.getTelefone());
79             stmtInsertPessoa.setString(7, pessoa.getEmail());
80             stmtInsertPessoa.executeUpdate();
81
82             stmtInsertPessoaFisica.setInt(1, pessoa.getId());
83             stmtInsertPessoaFisica.setString(2, pessoa.getCpf());
84             stmtInsertPessoaFisica.executeUpdate();
85         }
86     }
87
88     public void alterar(PessoaFisica pessoa, String novoNome, String novoLogradouro, String novaCidade,
89                         String novoEstado, String novoTelefone, String novoEmail, String novoCpf) throws SQLException {
90         String sql = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, "
91             "+ estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
92         String sqlFisica = "UPDATE PessoaFisica SET cpf = ? WHERE idPessoa = ?";
93
94         try (Connection conn = ConektorBD.getConnection();
95              PreparedStatement stmt = conn.prepareStatement(sql);
96              PreparedStatement stmtFisica = conn.prepareStatement(sqlFisica)) {
97
98             stmt.setString(1, novoNome);
99             stmt.setString(2, novoLogradouro);
100            stmt.setString(3, novaCidade);
101            stmt.setString(4, novoEstado);
102            stmt.setString(5, novoTelefone);
103            stmt.setString(6, novoEmail);
104            stmt.setInt(7, pessoa.getId());
105            stmt.executeUpdate();
106
107            stmtFisica.setString(1, novoCpf);
108            stmtFisica.setInt(2, pessoa.getId());
109            stmtFisica.executeUpdate();
110        }
111    }
112
113    public void excluir(int id) throws SQLException {
114        String sql = "DELETE FROM PessoaFisica WHERE idPessoa = ?";
115        String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
116
117        try (Connection conn = ConektorBD.getConnection();
118              PreparedStatement stmt = conn.prepareStatement(sql);
119              PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
120            stmt.setInt(1, id);
121            stmt.executeUpdate();
122            stmtPessoa.setInt(1, id);
123            stmtPessoa.executeUpdate();
124
125            System.out.println("Pessoa fisica excluida com ID: " + id);
126        }
127    }
128}
129
```

```

1 package model;
2
3 import model.util.ConektorBD;
4 import java.sql.Connection;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 public class PessoaJuridicaDAO {
12     public PessoaJuridica getPessoa(int id) throws SQLException {
13         String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
14             "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaJuridica.cnpj " +
15             "FROM Pessoa " +
16             "JOIN PessoaJuridica ON Pessoa.idPessoa = PessoaJuridica.idPessoa " +
17             "WHERE Pessoa.idPessoa = ?";
18         try (Connection conn = ConektorBD.getConnection();
19              PreparedStatement stmt = ConektorBD.getPreparedStatement(sql)) {
20             stmt.setInt(1, id);
21             try (ResultSet rs = ConektorBD.getSelect(stmt)) {
22                 if (rs.next()) {
23                     PessoaJuridica pessoa = new PessoaJuridica();
24                     pessoa.setId(rs.getInt("idPessoa"));
25                     pessoa.setName(rs.getString("nome"));
26                     pessoa.setLogradouro(rs.getString("logradouro"));
27                     pessoa.setCidade(rs.getString("cidade"));
28                     pessoa.setEstado(rs.getString("estado"));
29                     pessoa.setTelefone(rs.getString("telefone"));
30                     pessoa.setEmail(rs.getString("email"));
31                     pessoa.setCnpj(rs.getString("cnpj"));
32                     return pessoa;
33                 }
34             }
35         }
36         return null;
37     }
38
39     public List<PessoaJuridica> getPessoas() throws SQLException {
40         List<PessoaJuridica> pessoas = new ArrayList<>();
41         String sql = "SELECT Pessoa.idPessoa, Pessoa.nome, Pessoa.logradouro, Pessoa.cidade, " +
42             "Pessoa.estado, Pessoa.telefone, Pessoa.email, PessoaJuridica.cnpj " +
43             "FROM Pessoa " +
44             "JOIN PessoaJuridica ON Pessoa.idPessoa = PessoaJuridica.idPessoa";
45         try (Connection conn = ConektorBD.getConnection();
46              PreparedStatement stmt = ConektorBD.getPreparedStatement(sql)) {
47             try (ResultSet rs = ConektorBD.getSelect(stmt)) {
48                 while (rs.next()) {
49                     PessoaJuridica pessoa = new PessoaJuridica();
50                     pessoa.setId(rs.getInt("idPessoa"));
51                     pessoa.setName(rs.getString("nome"));
52                     pessoa.setLogradouro(rs.getString("logradouro"));
53                     pessoa.setCidade(rs.getString("cidade"));
54                     pessoa.setEstado(rs.getString("estado"));
55                     pessoa.setTelefone(rs.getString("telefone"));
56                     pessoa.setEmail(rs.getString("email"));
57                     pessoa.setCnpj(rs.getString("cnpj"));
58                     pessoas.add(pessoa);
59                 }
60             }
61         }
62         return pessoas;
63     }
64
65     public void incluir(PessoaJuridica pessoa) throws SQLException {
66         String sqlInsertPessoa = "INSERT INTO Pessoa (idPessoa, nome, logradouro, cidade, " +
67             "estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
68         String sqlInsertPessoaJuridica = "INSERT INTO PessoaJuridica (idPessoa, cnpj)" +
69             "VALUES (?)";
70         try (Connection conn = ConektorBD.getConnection();
71              PreparedStatement stmtInsertPessoa = conn.prepareStatement(sqlInsertPessoa);
72              PreparedStatement stmtInsertPessoaJuridica = conn.prepareStatement(sqlInsertPessoaJuridica)) {
73             stmtInsertPessoa.setInt(1, pessoa.getId());
74             stmtInsertPessoa.setString(2, pessoa.getName());
75             stmtInsertPessoa.setString(3, pessoa.getLogradouro());
76             stmtInsertPessoa.setString(4, pessoa.getCidade());
77             stmtInsertPessoa.setString(5, pessoa.getEstado());
78             stmtInsertPessoa.setString(6, pessoa.getTelefone());
79             stmtInsertPessoa.setString(7, pessoa.getEmail());
80             stmtInsertPessoa.executeUpdate();
81
82             stmtInsertPessoaJuridica.setInt(1, pessoa.getId());
83             stmtInsertPessoaJuridica.setString(2, pessoa.getCnpj());
84             stmtInsertPessoaJuridica.executeUpdate();
85         }
86     }
87
88     public void alterar(PessoaJuridica pessoa, String novoNome, String novoLogradouro, String novaCidade,
89                         String novoEstado, String novoTelefone, String novoEmail, String novoCnpj) throws SQLException {
90         String sql = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, " +
91             "+ estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
92         String sqlJuridica = "UPDATE PessoaJuridica SET cnpj = ? WHERE idPessoa = ?";
93
94         try (Connection conn = ConektorBD.getConnection();
95              PreparedStatement stmt = conn.prepareStatement(sql);
96              PreparedStatement stmtJuridica = conn.prepareStatement(sqlJuridica)) {
97             stmt.setString(1, novoNome);
98             stmt.setString(2, novoLogradouro);
99             stmt.setString(3, novaCidade);
100            stmt.setString(4, novoEstado);
101            stmt.setString(5, novoTelefone);
102            stmt.setString(6, novoEmail);
103            stmt.setInt(7, pessoa.getId());
104            stmt.executeUpdate();
105            stmtJuridica.setString(1, novoCnpj);
106            stmtJuridica.setInt(2, pessoa.getId());
107            stmtJuridica.executeUpdate();
108        }
109    }
110
111    public void excluir(int id) throws SQLException {
112        String sql = "DELETE FROM PessoaJuridica WHERE idPessoa = ?";
113        String sqlPessoa = "DELETE FROM Pessoa WHERE idPessoa = ?";
114
115        try (Connection conn = ConektorBD.getConnection();
116             PreparedStatement stmt = conn.prepareStatement(sql);
117             PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
118            stmt.setInt(1, id);
119            stmt.executeUpdate();
120
121            stmtPessoa.setInt(1, id);
122            stmtPessoa.executeUpdate();
123            System.out.println("Pessoa juridica excluida com ID: " + id);
124        }
125    }
126}
127}
128

```

```
1 package model.util;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.sql.Statement;
9
10 public class ConectorBD {
11     private static final String URL = "jdbc:sqlserver://localhost:1433;databaseName=Loja;" +
12             + "encrypt=true;trustServerCertificate=true";
13     private static final String USER = "loja";
14     private static final String PASSWORD = "loja";
15
16     public static Connection getConnection() throws SQLException {
17         return DriverManager.getConnection(URL, USER, PASSWORD);
18     }
19
20     public static PreparedStatement getPrepared(String sql) throws SQLException {
21         return getConnection().prepareStatement(sql);
22     }
23
24     public static ResultSet getSelect(PreparedStatement stmt) throws SQLException {
25         return stmt.executeQuery();
26     }
27
28     public static void close(Connection conn) throws SQLException {
29         if (conn != null) {
30             conn.close();
31         }
32     }
33
34     public static void close(Statement stmt) throws SQLException {
35         if (stmt != null) {
36             stmt.close();
37         }
38     }
39
40     public static void close	ResultSet rs) throws SQLException {
41         if (rs != null) {
42             rs.close();
43         }
44     }
45 }
46
```

```
1 package model.util;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7
8 public class SequenceManager {
9     public static int getValue(String sequenceName) throws SQLException {
10         String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS nextval";
11         try (Connection conn = ConectorBD.getConnection();
12              PreparedStatement stmt = conn.prepareStatement(sql);
13              ResultSet rs = stmt.executeQuery()) {
14             if (rs.next()) {
15                 return rs.getInt("nextval");
16             } else {
17                 throw new SQLException("Não foi possível obter o próximo valor da sequência"
18                                     + sequenceName);
19             }
20         }
21     }
22 }
23
```

```
1 import model.PessoaFisica;
2 import model.PessoaFisicaDAO;
3 import model.PessoaJuridica;
4 import model.PessoaJuridicadao;
5 import model.util.ConectorBD;
6 import java.sql.SQLException;
7 import java.sql.Connection;
8 import java.util.List;
9
10 public class CadastroBDTeste {
11     public static void main(String[] args) {
12         try {
13             Connection conn = ConectorBD.getConnection();
14             PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
15             PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();
16
17             PessoaFisica pf = new PessoaFisica(6, "Filipe", "Rua Um, 10", "Resende", "RJ",
18                     "1234-1234", "filipe@email.com", "12345678910");
19
20             pfDAO.incluir(pf);
21             System.out.println("Pessoa fisica criada:");
22             pf.exibir();
23             System.out.println();
24
25             pfDAO.alterar(pf, "Filipe Alterado", "Rua Dois, 20", "Araguari", "MG",
26                     "9999-8888", "filipealterado@email.com", "12345678911");
27             System.out.println("-----");
28             System.out.println("Dados da pessoa fisica alterados.");
29             System.out.println("-----");
30             System.out.println();
31
32             List<PessoaFisica> pessoasFisicas = pfDAO.getPessoas();
33             System.out.println("Todas as pessoas fisicas:");
34             for (PessoaFisica pessoaFisica : pessoasFisicas) {
35                 pessoaFisica.exibir();
36             }
37             System.out.println();
38
39             System.out.println("-----");
40             pfDAO.excluir(pf.getId());
41             System.out.println("-----");
42             System.out.println();
43
44             PessoaJuridica pj = new PessoaJuridica(7, "Empresa Um", "Avenida Legal, 10",
45                     "Sao Paulo", "SP", "1234-1234", "empresalegal@email.com", "123456789101");
46
47             pjDAO.incluir(pj);
48             System.out.println("Pessoa juridica criada:");
49             pj.exibir();
50             System.out.println();
51
52             pjDAO.alterar(pj, "Empresa Mais legal", "Avenida Dois, 20", "Rio de Janeiro", "RJ",
53                     "4444-4444", "empresamais@email.com", "10987654321");
54             System.out.println("-----");
55             System.out.println("Dados da pessoa juridica alterados.");
56             System.out.println("-----");
57             System.out.println();
58
59             List<PessoaJuridica> pessoasJuridicas = pjDAO.getPessoas();
60             System.out.println("Todas as pessoas juridicas:");
61             for (PessoaJuridica pessoaJuridica : pessoasJuridicas) {
62                 pessoaJuridica.exibir();
63             }
64             System.out.println();
65
66             System.out.println("-----");
67             pjDAO.excluir(pj.getId());
68             System.out.println("-----");
69             System.out.println();
70
71             ConectorBD.close(conn);
72
73         } catch (SQLException e) {
74             System.out.println("Ocorreu um erro: " + e.getMessage());
75         }
76     }
77 }
78 }
```

## Resultados:

```
Pessoa fisica criada:  
-----  
ID: 6  
Nome: Filipe  
Logradouro: Rua Um, 10  
Cidade: Resende  
Estado: RJ  
Telefone: 1234-1234  
Email: filipe@email.com  
-----  
CPF: 12345678910  
-----  
Dados da pessoa fisica alterados.  
-----
```

```
-----  
ID: 6  
Nome: Filipe Alterado  
Logradouro: Rua Dois, 20  
Cidade: Araguari  
Estado: MG  
Telefone: 9999-8888  
Email: filipealterado@email.com  
-----  
CPF: 12345678911  
-----
```

## **Análise e Conclusão:**

### **1. Qual a importância dos componentes de middleware, com o JDBC?**

**R:** O JDBC com o middleware é uma solução para integrar diferentes sistemas e banco de dados. Permitindo que as aplicações sejam seguras e fáceis de manter.

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

**R:** O Statement pode ser utilizado em casos mais específicos mas não é recomendado quando a mesma consulta será executada várias vezes com diferentes valores. O PreparedStatement é mais recomendado na maioria dos casos por oferecer mais segurança, desempenho e facilidade de manutenção.

### **3. Como o padrão DAO melhora a manutenibilidade do software?**

**R:** O padrão DAO torna o código mais modular, testável e fácil de entender. Isso trás diversos benefícios para a manutenção do código.

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

**R:** A herança em dados relacionais é uma adaptação do conceito de herança da programação orientada a objetos, utilizando estruturas tabulares para representar hierarquias entre entidades.

## 2º Procedimento | Alimentando a Base

```
1 import model.PessoaFisica;
2 import model.PessoaFisicaDAO;
3 import model.PessoaJuridica;
4 import model.PessoaJuridicaDAO;
5 import java.util.Scanner;
6 import java.sql.SQLException;
7 import java.util.ArrayList;
8
9 public class CadastroBD {
10     private static final Scanner sc = new Scanner(System.in);
11     private static final PessoaFisicaDAO pfDao = new PessoaFisicaDAO();
12     private static final PessoaJuridicaDAO pjDao = new PessoaJuridicaDAO();
13
14     public static void main(String[] args) {
15         int opcao = -1;
16         while (opcao != 0) {
17             printMenu();
18             opcao = inputInt("ESCOLHA: ");
19             switch (opcao) {
20                 case 1 -> incluir();
21                 case 2 -> alterar();
22                 case 3 -> excluir();
23                 case 4 -> buscarPeloId();
24                 case 5 -> exibirTodos();
25                 case 0 -> System.out.println("Finalizando...");
26                 default -> System.out.println("Escolha invalida!");
27             }
28         }
29     }
30
31     private static void printMenu() {
32         System.out.println("\n-----");
33         System.out.println("1 - Incluir");
34         System.out.println("2 - Alterar");
35         System.out.println("3 - Excluir");
36         System.out.println("4 - Buscar pelo ID");
37         System.out.println("5 - Exibir todos");
38         System.out.println("0 - Sair");
39         System.out.println("-----");
40     }
41
42     private static String input(String prompt) {
43         System.out.print(prompt);
44         return sc.nextLine();
45     }
46
47     private static int inputInt(String prompt) {
48         System.out.print(prompt);
49         try {
50             return Integer.parseInt(sc.nextLine());
51         } catch (NumberFormatException e) {
52             System.out.println("Erro: Entrada invalida. Tente novamente.");
53             return inputInt(prompt);
54         }
55     }
56 }
```

```
57     private static void incluir() {
58         System.out.println("\nIncluindo pessoa...");
59         System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
60         String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
61         Integer id = inputInt("Informe o ID: ");
62         switch (tipoPessoa) {
63             case "F" -> {
64                 try {
65                     pfDao.incluir(criarPessoaFisica(id));
66                     System.out.println("Pessoa física incluída com sucesso!");
67                 } catch (SQLException e) {
68                     System.out.println("Erro ao incluir pessoa física: " + e.getMessage());
69                 }
70             }
71             case "J" -> {
72                 try {
73                     pjDao.incluir(criarPessoaJuridica(id));
74                     System.out.println("Pessoa jurídica incluída com sucesso!");
75                 } catch (SQLException e) {
76                     System.out.println("Erro ao incluir pessoa jurídica: " + e.getMessage());
77                 }
78             }
79             default -> System.out.println("Tipo de pessoa inválido!");
80         }
81     }
82
83     private static PessoaFisica criarPessoaFisica(Integer id) {
84         System.out.println("Criando Pessoa Física...");
85         String nome = input("Informe o nome: ");
86         String logradouro = input("Informe o logradouro: ");
87         String cidade = input("Informe a cidade: ");
88         String estado = input("Informe o estado: ");
89         String telefone = input("Informe o telefone: ");
90         String email = input("Informe o email: ");
91         String cpf = input("Informe o CPF: ");
92         return new PessoaFisica(id, nome, logradouro, cidade, estado, telefone, email, cpf);
93     }
94
95     private static PessoaJuridica criarPessoaJuridica(Integer id) {
96         System.out.println("Criando Pessoa Jurídica...");
97         String nome = input("Informe o nome: ");
98         String logradouro = input("Informe o logradouro: ");
99         String cidade = input("Informe a cidade: ");
100        String estado = input("Informe o estado: ");
101        String telefone = input("Informe o telefone: ");
102        String email = input("Informe o email: ");
103        String cnpj = input("Informe o CNPJ: ");
104        return new PessoaJuridica(id, nome, logradouro, cidade, estado, telefone, email, cnpj);
105    }
106}
```

```

107     private static void alterar() {
108         System.out.println("\nAlterando pessoa...");
109         System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
110         String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
111         if (tipoPessoa.equals("F")) {
112             try {
113                 Integer id = inputInt("Informe o ID da Pessoa Física: ");
114                 PessoaFisica pf = pfDao.getPessoa(id);
115                 if (pf != null) {
116                     System.out.println("Dados atuais da Pessoa Física:");
117                     pf.exibir();
118
119                     String novoNome = input("Informe o novo nome: ");
120                     String novoLogradouro = input("Informe o novo logradouro: ");
121                     String novaCidade = input("Informe a nova cidade: ");
122                     String novoEstado = input("Informe o novo estado: ");
123                     String novoTelefone = input("Informe o novo telefone: ");
124                     String novoEmail = input("Informe o novo email: ");
125                     String novoCpf = input("Informe o novo CPF: ");
126
127                     pfDao.alterar(pf, novoNome, novoLogradouro, novaCidade,
128                             novoEstado, novoTelefone, novoEmail, novoCpf);
129                     System.out.println("Pessoa física alterada com sucesso!");
130                 } else {
131                     System.out.println("ID errado!");
132                 }
133             } catch (NullPointerException | SQLException e) {
134                 System.out.println("Erro ao alterar pessoa física: " + e.getMessage());
135             }
136         } else if (tipoPessoa.equals("J")) {
137             try {
138                 Integer id = inputInt("Informe o ID da Pessoa Jurídica: ");
139                 PessoaJuridica pj = pjDao.getPessoa(id);
140                 if (pj != null) {
141                     System.out.println("Dados atuais da Pessoa Jurídica:");
142                     pj.exibir();
143
144                     String novoNome = input("Informe o novo nome: ");
145                     String novoLogradouro = input("Informe o novo logradouro: ");
146                     String novaCidade = input("Informe a nova cidade: ");
147                     String novoEstado = input("Informe o novo estado: ");
148                     String novoTelefone = input("Informe o novo telefone: ");
149                     String novoEmail = input("Informe o novo email: ");
150                     String novoCnpj = input("Informe o novo CNPJ: ");
151
152                     pjDao.alterar(pj, novoNome, novoLogradouro, novaCidade,
153                             novoEstado, novoTelefone, novoEmail, novoCnpj);
154                     System.out.println("Pessoa jurídica alterada com sucesso!");
155                 } else {
156                     System.out.println("ID errado!");
157                 }
158             } catch (NullPointerException | SQLException e) {
159                 System.out.println("Erro ao alterar pessoa jurídica: " + e.getMessage());
160             }
161         } else {
162             System.out.println("Tipo de pessoa inválido!");
163         }
164     }
165 }
```

```

166     private static void excluir() {
167         System.out.println("\nExcluindo pessoa...");
168         System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
169         String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
170         switch (tipoPessoa) {
171             case "F" -> {
172                 try {
173                     Integer id = inputInt("Informe o ID da Pessoa Fisica: ");
174                     PessoaFisica pf = pfDao.getPessoa(id);
175                     if (pf != null) {
176                         pfDao.excluir(pf.getId());
177                         System.out.println("Sucesso ao excluir!");
178                     } else {
179                         System.out.println("ID errado!");
180                     }
181                 } catch (NullPointerException | SQLException e) {
182                     System.out.println("Erro ao excluir pessoa fisica: " + e.getMessage());
183                 }
184             }
185             case "J" -> {
186                 try {
187                     Integer id = inputInt("Informe o ID da Pessoa Juridica: ");
188                     PessoaJuridica pj = pjDao.getPessoa(id);
189                     if (pj != null) {
190                         pjDao.excluir(pj.getId());
191                         System.out.println("Sucesso ao excluir!");
192                     } else {
193                         System.out.println("ID errado!");
194                     }
195                 } catch (NullPointerException | SQLException e) {
196                     System.out.println("Erro ao excluir pessoa juridica: " + e.getMessage());
197                 }
198             }
199             default -> System.out.println("Tipo de pessoa invalido!");
200         }
201     }
202
203     private static void buscarPeloId() {
204         System.out.println("\nBuscando pessoa pelo ID...");
205         System.out.println("F - Pessoa Fisica | J - Pessoa Juridica");
206         String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
207         switch (tipoPessoa) {
208             case "F" -> {
209                 try {
210                     Integer id = inputInt("Informe o ID da Pessoa Fisica: ");
211                     PessoaFisica pf = pfDao.getPessoa(id);
212                     if (pf != null) {
213                         pf.exibir();
214                     } else {
215                         System.err.println("Pessoa fisica com o ID " + id + " nao encontrada!");
216                     }
217                 } catch (SQLException e) {
218                     System.err.println("Erro ao buscar pessoa fisica: " + e.getMessage());
219                 }
220             }
221             case "J" -> {
222                 try {
223                     Integer id = inputInt("Informe o ID da Pessoa Juridica: ");
224                     PessoaJuridica pj = pjDao.getPessoa(id);
225                     if (pj != null) {
226                         pj.exibir();
227                     } else {
228                         System.err.println("Pessoa juridica com o ID " + id + " nao encontrada!");
229                     }
230                 } catch (SQLException e) {
231                     System.err.println("Erro ao buscar pessoa juridica: " + e.getMessage());
232                 }
233             }
234             default -> System.out.println("Tipo de pessoa invalido!");
235         }
236     }
237 }
```

```

238     private static void exibirTodos() {
239         System.out.println("\nExibindo todas as pessoas...");
240         System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
241         String tipoPessoa = input("TIPO DE PESSOA: ").toUpperCase();
242         try {
243             switch (tipoPessoa) {
244                 case "F" -> {
245                     ArrayList<PessoaFísica> listaPf = (ArrayList<PessoaFísica>) pfDao.getPessoas();
246                     for (PessoaFísica pessoa : listaPf) {
247                         pessoa.exibir();
248                     }
249                 }
250                 case "J" -> {
251                     ArrayList<PessoaJurídica> listaPj = (ArrayList<PessoaJurídica>) pjDao.getPessoas();
252                     for (PessoaJurídica pessoa : listaPj) {
253                         pessoa.exibir();
254                     }
255                 }
256                 default -> System.out.println("Tipo de pessoa invalido!");
257             }
258         } catch (SQLException e) {
259             System.out.println("Erro ao exibir pessoas: " + e.getMessage());
260         }
261     }
262 }
263

```

## Resultados:

```

-----
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
-----
ESCOLHA: 1

Incluindo pessoa...
F - Pessoa Física | J - Pessoa Jurídica
TIPO DE PESSOA: f
Informe o ID: 10
Criando Pessoa Física...
Informe o nome: teste
Informe o logradouro: rua t, 7
Informe a cidade: Brasilia
Informe o estado: DF
Informe o telefone: 7777-8888
Informe o email: teste@gmail.com
Informe o CPF: 99999999999
Pessoa física incluida com sucesso!

-----
1 - Incluir
2 - Alterar
3 - Excluir
4 - Buscar pelo ID
5 - Exibir todos
0 - Sair
-----
ESCOLHA: 5

```

ESCOLHA: 5

Exibindo todas as pessoas...  
F - Pessoa Fisica | J - Pessoa Juridica  
TIPO DE PESSOA: F

-----

ID: 1

Nome: Ian  
Logradouro: Rua A, 1  
Cidade: Resende  
Estado: RJ  
Telefone: 1111-1111  
Email: ian@gmail.com

-----

CPF: 111111111111

-----

ID: 2

Nome: Sofia  
Logradouro: Rua B, 2  
Cidade: Araguari  
Estado: MG  
Telefone: 2222-2222  
Email: sofia@gmail.com

-----

CPF: 222222222222

-----

ID: 3

Nome: Alan  
Logradouro: Rua C, 3  
Cidade: Porto Alegre  
Estado: RS  
Telefone: 3333-3333  
Email: alan@gmail.com

-----

CPF: 333333333333

-----

ID: 10

Nome: teste  
Logradouro: rua t, 7  
Cidade: Brasilia  
Estado: DF  
Telefone: 7777-8888  
Email: teste@gmail.com

-----

CPF: 999999999999

-----

## **Análise e Conclusão:**

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

R: A persistência de dados define como a informação é armazenada permanentemente. Ela possui dois métodos principais, arquivos e bancos de dados. Arquivos são mais simples de implementar mas possuem problemas de segurança. Bancos de dados garantem a segurança mas exigem maior conhecimento para a implementação e gestão.

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

R: Antes das lambdas a impressão de valores exigia loops e métodos para acessar atributos. Com lambdas é possível tratar cada elemento de uma coleção com uma função anônima. As lambdas também se integram com streams para operações como ordenação e filtragem.

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

R: No Java, os métodos marcados como static pertencem à classe e não a uma instância específica. O método main é o ponto de entrada da aplicação e precisa ser static para ser chamado pela JVM antes de criar os objetos. Por isso o uso de static no método main é fundamental para que a JVM inicie a execução da aplicação e garanta que o código seja executado corretamente.