



Estácio

UNIVERSIDADE ESTÁCIO DE SÁ DE RIBEIRÃO PRETO

POLO PARQUE ANDORINHAS

DESENVOLVIMENTO FULL STACK

2023.3 FULL STACK ALUNO

NIVEL 3: BACK-END SEM BANCO NÃO TEM

ALESSANDRO SENDI SHIGEMATSU

Título da Prática

RPG0016 - BackEnd sem banco não tem

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.
5. No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

1º Procedimento | Criando o Banco de Dados

Todos os códigos solicitados neste roteiro de aula

```
package cadastrobd.model;
```

```
public class Pessoa {
```

```
    public Pessoa(int id, String nome, String logradouro, String cidade, String estado, String  
    telefone, String email) {
```

```
        this.id = id;
```

```
        this.nome = nome;
```

```
        this.logradouro = logradouro;
```

```
        this.cidade = cidade;
```

```
        this.estado = estado;
```

```
        this.telefone = telefone;
```

```
        this.email = email;
```

```
    }
```

```
    private int id;
```

```
    private String nome;
```

```
    private String logradouro;
```

```
    private String cidade;
```

```
private String estado;  
private String telefone;  
private String email;
```

```
public String getNome() {  
    return nome;  
}
```

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

```
public String getLogradouro() {  
    return logradouro;  
}
```

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

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

```
public void setCidade(String cidade) {  
    this.cidade = cidade;  
}
```

```
public String getEstado() {  
    return estado;  
}
```

```
public void setEstado(String estado) {  
    this.estado = estado;  
}
```

```
public String getTelefono() {  
    return telefono;  
}
```

```
public void setTelefono(String telefono) {  
    this.telefono = telefono;  
}
```

```
public String getEmail() {  
    return email;  
}
```

```
public void setEmail(String email) {  
    this.email = email;  
}
```

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

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

```
public void exhibir() {  
    System.out.println("ID: " + id);  
}
```

```
        System.out.println("Nome: " + nome);

        System.out.println("Logradouro: " + logradouro);

        System.out.println("Cidade: " + cidade);

        System.out.println("Estado: " + estado);

        System.out.println("Telefone: " + telefone);

        System.out.println("Email: " + email);

    }
}



---



public class PessoaFisica extends Pessoa {

    private int cpf;

    public PessoaFisica ( int id, String nome, String logradouro, String cidade, String estado,
String telefone, String email,int cpf ) {

        super( id, nome, logradouro, cidade, estado, telefone, email );

        this.cpf = cpf;

    }

    public int getCpf() {

        return cpf;

    }

    public void setCpf(int cpf) {

        this.cpf = cpf;

    }

    @Override
    public void exibir() {

        super.exibir();

        System.out.println("CPF: " + cpf);

    }

}



---


```

```
package cadastrobd.model;
```

```
import cadastrobd.model.util.ConectorBD;
```

```
import cadastrobd.model.util.SequenceManager;
```

```
import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
```

```
import java.util.ArrayList;
```

```
public class PessoaFisicaDAO {
```

```
    private ConectorBD conector;
```

```
    private SequenceManager sequenceManager;
```

```
    public PessoaFisicaDAO() {
```

```
        this.conector = new ConectorBD();
```

```
        this.sequenceManager = new SequenceManager();
```

```
    }
```

```
    public PessoaFisica getPessoa(int id) {
```

```
        Connection connection = null;
```

```
        PreparedStatement statement = null;
```

```
        ResultSet resultSet = null;
```

```
        try {
```

```
            connection = conector.getConnection();
```

```
            String sql = "SELECT * FROM Pessoa right join PessoaFisica on idPessoa =  
Pessoa_idPessoa where idPessoa = ?";
```

```
            statement = conector.getPrepared( sql, connection );
```

```
            statement.setInt(1, id);
```

```

resultSet = conector.getSelect( statement );

while(resultSet.next())
{

    return new PessoaFisica(
        resultSet.getInt("idPessoa"),
        resultSet.getString("nome"),
        resultSet.getString("logradouro") ,
        resultSet.getString("cidade") ,
        resultSet.getString("estado") ,
        resultSet.getString("telefone") ,
        resultSet.getString("email") ,
        resultSet.getInt("CPF")
    );
}

} catch (SQLException e) {
    e.printStackTrace();
    // Tratamento de exceções
} finally {
    conector.closeAll(connection, statement, resultSet);
}

return null;

}

public ArrayList<PessoaFisica> getPessoas() {

    Connection connection = null;

```

```

PreparedStatement statement = null;

ResultSet resultSet = null;

ArrayList<PessoaFisica> pessoas = new ArrayList<>();

try {

    connection = conector.getConnection();

    String sql = "SELECT * FROM Pessoa right join PessoaFisica on idPessoa =
Pessoa_idPessoa";

    statement = conector.getPrepared( sql, connection );

    resultSet = conector.getSelect( statement );

    while(resultSet.next())
    {
        pessoas.add( new PessoaFisica(
            resultSet.getInt("idPessoa"),
            resultSet.getString("nome"),
            resultSet.getString("logradouro") ,
            resultSet.getString("cidade") ,
            resultSet.getString("estado") ,
            resultSet.getString("telefone") ,
            resultSet.getString("email") ,
            resultSet.getInt("CPF")
        ));
    }

    return pessoas;

} catch (SQLException e) {

    e.printStackTrace();

    // Tratamento de exceções

} finally {

```



```
conector.closeAll(connection, statement, resultSet);  
}  
  
return null;  
}  
  
public void incluir(PessoaFisica pessoa) {  
    Connection connection = null;  
    PreparedStatement statement = null;  
  
    try {  
        int nextId = sequenceManager.getValue("sequencia_pessoa");  
        pessoa.setId(nextId); // Define o ID da pessoa física  
  
        connection = conector.getConnection();  
  
        String sql = ""  
            "INSERT INTO [dbo].[Pessoa]  
            (idPessoa  
            ,nome  
            ,[logradouro]  
            ,[cidade]  
            ,[estado]  
            ,[telefone]  
            ,[email])  
            VALUES  
            (  
            ?  
            , ?  
            , ?  
            , ?  
            , ?  
            , ?  
            )";
```

```
, ? );
```

```
INSERT INTO [dbo].[PessoaFisica]
```

```
([Pessoa_idPessoa]
```

```
,[CPF])
```

```
VALUES
```

```
( ?
```

```
, ? );
```

```
""";
```

```
statement = conector.getPrepared( sql, connection );
```

```
statement.setInt(1, pessoa.getId() );
```

```
statement.setString(2, pessoa.getNome());
```

```
statement.setString(3, pessoa.getLogradouro());
```

```
statement.setString(4, pessoa.getCidade());
```

```
statement.setString(5, pessoa.getEstado());
```

```
statement.setString(6, pessoa.getTelefone());
```

```
statement.setString(7, pessoa.getEmail());
```

```
statement.setInt(8, pessoa.getId());
```

```
statement.setInt(9, pessoa.getCpf());
```

```
statement.executeUpdate();
```

```
} catch (SQLException e) {
```

```
    e.printStackTrace();
```

```
    // Tratamento de exceções
```

```
} finally {
```

```
    conector.closeConnection(connection);
```

```
    conector.closeStatement(statement);
```

```
}  
}
```

```
public void alterar( int id , PessoaFisica pessoa) {
```

```
    Connection connection = null;
```

```
    PreparedStatement statement = null;
```

```
    try {
```

```
        connection = conector.getConnection();
```

```
        String sql = ""
```

```
            UPDATE [dbo].[Pessoa]
```

```
            SET
```

```
                nome = ?
```

```
            ,[logradouro] = ?
```

```
            ,[cidade] = ?
```

```
            ,[estado] = ?
```

```
            ,[telefone] = ?
```

```
            ,[email] = ?
```

```
            where idPessoa = ?
```

```
        UPDATE [dbo].[PessoaFisica]
```

```
        SET [CPF] = ?
```

```
        where Pessoa_idPessoa = ?
```

```
        """;
```

```
        statement = conector.getPrepared( sql, connection );
```

```
        statement.setString(1, pessoa.getNome());
```

```
        statement.setString(2, pessoa.getLogradouro());
```

```

        statement.setString(3, pessoa.getCidade());
        statement.setString(4, pessoa.getEstado());
        statement.setString(5, pessoa.getTelefone());
        statement.setString(6, pessoa.getEmail());
        statement.setInt(7, id );
        statement.setInt(8, pessoa.getCpf());
        statement.setInt(9, id );

        statement.executeUpdate();

    } catch (SQLException e) {

        e.printStackTrace();
        // Tratamento de exceções
    } finally {
        conector.closeConnection(connection);
        conector.closeStatement(statement);
    }
}

public void excluir(int id) {
    // Lógica para remoção de uma pessoa física do banco em ambas as tabelas
    Connection connection = null;
    PreparedStatement statement = null;

    try {

        connection = conector.getConnection();

        String sql = ""
            DELETE FROM [Pessoa] WHERE idPessoa = ?

```

```

        DELETE FROM [PessoaFisica] WHERE pessoa_idPessoa = ?
        """";

statement = conector.getPrepared( sql, connection );

statement.setInt(1, id );
statement.setInt(2, id );


statement.executeUpdate();

} catch (SQLException e) {

    e.printStackTrace();
    // Tratamento de exceções
} finally {
    conector.closeConnection(connection);
    conector.closeStatement(statement);
}
}
}

}
}



---


public class PessoaJuridica extends Pessoa {
    private int cnpj;


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



---


package cadastrobd.model;

```

```
import cadastrbd.model.util.ConectorBD;
import cadastrbd.model.util.SequenceManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
```

```
public class PessoaJuridicaDAO {
    private ConectorBD conector;
    private SequenceManager sequenceManager;

    public PessoaJuridicaDAO() {
        this.conector = new ConectorBD();
        this.sequenceManager = new SequenceManager();
    }
}
```

```
public PessoaJuridica getPessoa(int id) {

    Connection connection = null;
    PreparedStatement statement = null;
    ResultSet resultSet = null;

    try {
        connection = conector.getConnection();

        String sql = "SELECT * FROM Pessoa right join PessoaJuridica on idPessoa = Pessoa_idPessoa where idPessoa = ?";

        statement = conector.getPrepared( sql, connection );

        statement.setInt(1, id);
```

```

resultSet = conector.getSelect( statement );

while(resultSet.next())
{

    return new PessoaJuridica(
        resultSet.getInt("idPessoa"),
        resultSet.getString("nome"),
        resultSet.getString("logradouro") ,
        resultSet.getString("cidade") ,
        resultSet.getString("estado") ,
        resultSet.getString("telefone") ,
        resultSet.getString("email") ,
        resultSet.getInt("CNPJ")
    );
}

} catch (SQLException e) {
    e.printStackTrace();
    // Tratamento de exceções
} finally {
    conector.closeAll(connection, statement, resultSet);
}

return null;

}

public ArrayList<PessoaJuridica> getPessoas() {

```

```

    Connection connection = null;
    PreparedStatement statement = null;

```

```

ResultSet resultSet = null;

ArrayList<PessoaJuridica> pessoas = new ArrayList<>();

try {
    connection = conector.getConnection();

    String sql = "SELECT * FROM Pessoa right join PessoaJuridica on idPessoa = Pessoa_idPessoa";

    statement = conector.getPrepared( sql, connection );

    resultSet = conector.getSelect( statement );

    while(resultSet.next())
    {
        pessoas.add( new PessoaJuridica(
            resultSet.getInt("idPessoa"),
            resultSet.getString("nome"),
            resultSet.getString("logradouro") ,
            resultSet.getString("cidade") ,
            resultSet.getString("estado") ,
            resultSet.getString("telefone") ,
            resultSet.getString("email") ,
            resultSet.getInt("CNPJ")
        ));
    }

    return pessoas;

} catch (SQLException e) {
    e.printStackTrace();

    // Tratamento de exceções
} finally {
    conector.closeAll(connection, statement, resultSet);
}

```



```
}  
  
return null;  
  
}
```

```
public void incluir(PessoaJuridica pessoa) {  
  
    Connection connection = null;  
  
    PreparedStatement statement = null;  
  
    try {  
  
        int nextId = sequenceManager.getValue("sequencia_pessoa");  
  
        pessoa.setId(nextId); // Define o ID da pessoa física  
  
  
        connection = conector.getConnection();  
  
  
        String sql = ""  
            INSERT INTO [dbo].[Pessoa]  
            (idPessoa  
            ,nome  
            ,[logradouro]  
            ,[cidade]  
            ,[estado]  
            ,[telefone]  
            ,[email])  
            VALUES  
            ( ?  
            , ?  
            , ?  
            , ?  
            , ?  
            , ?  
            , ? );
```

```
INSERT INTO [dbo].[PessoaJuridica]
([Pessoa_idPessoa]
,[CNPJ])
VALUES
( ?
, ? );
""";
```

```
statement = conector.getPrepared( sql, connection );
```

```
statement.setInt(1, pessoa.getId() );
statement.setString(2, pessoa.getNome());
statement.setString(3, pessoa.getLogradouro());
statement.setString(4, pessoa.getCidade());
statement.setString(5, pessoa.getEstado());
statement.setString(6, pessoa.getTelefone());
statement.setString(7, pessoa.getEmail());
statement.setInt(8, pessoa.getId());
statement.setInt(9, pessoa.getCnpj());
```

```
statement.executeUpdate();
```

```
} catch (SQLException e) {
```

```
    e.printStackTrace();
```

```
    // Tratamento de exceções
```

```
} finally {
```

```
    conector.closeConnection(connection);
```

```
    conector.closeStatement(statement);
```

```
}
```

```
}
```

```
public void alterar( int id , PessoaJuridica pessoa) {
```

```
    Connection connection = null;
```

```
    PreparedStatement statement = null;
```

```
    try {
```

```
        connection = conector.getConnection();
```

```
        String sql = ""
```

```
            UPDATE [dbo].[Pessoa]
```

```
            SET
```

```
                nome = ?
```

```
            ,[logradouro] = ?
```

```
            ,[cidade] = ?
```

```
            ,[estado] = ?
```

```
            ,[telefone] = ?
```

```
            ,[email] = ?
```

```
            where idPessoa = ?
```

```
        UPDATE [dbo].[PessoaJuridica]
```

```
        SET [CNPJ] = ?
```

```
        where Pessoa_idPessoa = ?
```

```
        "";
```

```
        statement = conector.getPrepared( sql, connection );
```

```
        statement.setString(1, pessoa.getNome());
```

```
        statement.setString(2, pessoa.getLogradouro());
```

```
        statement.setString(3, pessoa.getCidade());
```

```

        statement.setString(4, pessoa.getEstado());
        statement.setString(5, pessoa.getTelefone());
        statement.setString(6, pessoa.getEmail());
        statement.setInt(7, id );
        statement.setInt(8, pessoa.getCnpj());
        statement.setInt(9, id );

        statement.executeUpdate();

    } catch (SQLException e) {

        e.printStackTrace();
        // Tratamento de exceções
    } finally {
        conector.closeConnection(connection);
        conector.closeStatement(statement);
    }
}

public void excluir(int id) {
    // Lógica para remoção de uma pessoa física do banco em ambas as tabelas
    Connection connection = null;
    PreparedStatement statement = null;

    try {

        connection = conector.getConnection();

        String sql = ""
            DELETE FROM [Pessoa] WHERE idPessoa = ?

```

```
DELETE FROM [PessoaJuridica] WHERE pessoa_idPessoa = ?  
""";
```

```
statement = conector.getPrepared( sql, connection );
```

```
statement.setInt(1, id );
```

```
statement.setInt(2, id );
```

```
statement.executeUpdate();
```

```
} catch (SQLException e) {
```

```
    e.printStackTrace();
```

```
    // Tratamento de exceções
```

```
} finally {
```

```
    conector.closeConnection(connection);
```

```
    conector.closeStatement(statement);
```

```
}
```

```
}
```

```
}
```

```
public int getCnpj() {
```

```
    return cnpj;
```

```
}
```

```
public void setCnpj(int cnpj) {
```

```
    this.cnpj = cnpj;
```

```
}
```

```
@Override
```

```
public void exibir() {
```

```

        super.exibir();

        System.out.println("CPF: " + cnpj);
    }
}

```

Instanciar uma pessoa física e persistir no banco de dados.

```

PessoaFisicaDAO pessoaFisicaDao = new PessoaFisicaDAO();

PessoaFisica pessoaFisica = new PessoaFisica( 1, "Ricardo", "Rua 112", "Dunas
milagrosas", "AC", "3213-3231", "fdaff", 555555545 );

pessoaFisicaDao.incluir( pessoaFisica );

```

Tabela pessoa

	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	1	Ricardo	Rua 112	Dunas milagrosas	AC	3213-3231	fdaff

Tabela Pessoa física

	Pessoa_idPessoa	CPF
1	1	555555545

Alterar os dados da pessoa física no banco.

```

PessoaFisica pessoaFisica = new PessoaFisica( 1, "Maria", "Rua 999", "Dunas perigosas",
"SP", "2222-2222", "oi@oooo.com", 999999999 );

pessoaFisicaDao.alterar( 1, pessoaFisica );

```

Tabela pessoa

	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	1	Maria	Rua 999	Dunas perigosas	SP	2222-2222	oi@oooo.com

Tabela pessoa física

	Pessoa_idPessoa	CPF
1	1	999999999

Consultar todas as pessoas físicas do banco de dados e listar no console.

```
ArrayList<PessoaFisica> pessoasFisicas = pessoaFisicaDao.getPessoas( );  
pessoasFisicas.stream().forEach( pessoa -> pessoa.exibir() );
```

```
SQL 1 execution x  CadastroBD (debug) x  Deb  
run:  
ID: 10  
Nome: Joao  
CPF: 111111111  
ID: 1  
Nome: Maria  
CPF: 999999999  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Excluir a pessoa física criada anteriormente no banco.

```
pessoaFisicaDao.excluir( 1 );
```

Resultados		Mensagens							
	idPessoa	nome	logradouro	cidade	estado	telefone	email	Pessoa_idPessoa	CPF
1	10	Joao	Rua 12, Casa 3,Quitanda	Riacho do Sul	PA	1111-1111	joao@riacho.com	10	111111111

Instanciar uma pessoa jurídica e persistir no banco de dados.

```
PessoaJuridica pessoaJuridica = new PessoaJuridica( 1 , "PPO", "Rua 1000", "Vila lobos",  
"BA" , "4444-4444", "PPO@EMAIL.COM", 22222222 );
```

```
pessoaJuridicaDAO.incluir( pessoaJuridica );
```

Resultados		Mensagens							
	idPessoa	nome	logradouro	cidade	estado	telefone	email	Pessoa_idPessoa	CNPJ
1	20	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjc@riacho.com	20	22222222222222
2	12	PPO	Rua 1000	Vila lobos	BA	4444-4444	PPO@EMAIL.COM	12	22222222

Alterar os dados da pessoa jurídica no banco.

```
PessoaJuridica pessoaJuridica = new PessoaJuridica( 1 , "AAC", "Rua 2000", "Vila Leão",  
"PA" , "5555-4444", "AAC@EMAIL.COM", 33333333 );
```

```
pessoaJuridicaDAO.alterar( 12, pessoaJuridica );
```

	idPessoa	nome	logradouro	cidade	estado	telefone	email	Pessoa_idPessoa	CNPJ
1	20	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjc@riacho.com	20	22222222222222
2	12	AAC	Rua 2000	Vila Leão	PA	5555-4444	AAC@EMAIL.COM	12	333333333

Consultar todas as pessoas jurídicas do banco e listar no console.

```
ArrayList<PessoaJuridica> pessoaJuridica = pessoaJuridicaDAO.getPessoas( );
pessoaJuridica.stream().forEach( pessoa -> pessoa.exibir() );
```

```
SQL 1 execution x  CadastroBD (debug) x De
run:
ID: 20
Nome: JJC
CPF: 61432718
ID: 12
Nome: AAC
CPF: 333333333
BUILD SUCCESSFUL (total time: 0 seconds)
```

Excluir a pessoa jurídica criada anteriormente no banco.

```
pessoaJuridicaDAO.excluir( 12 );
```

	idPessoa	nome	logradouro	cidade	estado	telefone	email	Pessoa_idPessoa	CNPJ
1	20	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjc@riacho.com	20	22222222222222

A saída do sistema

```
PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

ArrayList<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas( );
pessoasFisicas.stream().forEach( pessoa -> pessoa.exibir() );

ArrayList<PessoaJuridica> pessoaJuridica = pessoaJuridicaDAO.getPessoas( );
pessoaJuridica.stream().forEach( pessoa -> pessoa.exibir() );
```



```
SQL 1 execution x  CadastroBD (debug) x  Debug
Nome: Joao
Logradouro: Rua 12, Casa 3,Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 111111111
ID: 20
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CPF: 61432718
BUILD SUCCESSFUL (total time: 0 seconds)
```

Análise

- a) Qual a importância dos componentes de middleware, como o JDBC?
R: Permite que programas acessem o sistema de gerenciamento de banco de dados.
- b) Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?
R: Statement não aceita parâmetros e o PreparedStatement aceita.
- c) Como o padrão DAO melhora a manutenibilidade do software?
R: separação das regras de negócio das regras de acesso a banco de dados tanto quanto funcionalidades de bancos de dados, tais como obter conexões, mapear objetos para tipos de dados SQL ou executar comandos SQL.
- d) Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?
R: Criamos uma nova tabela, objeto filho, com a chave primária da tabela principal, objeto pai.

Conclusão

O Java possui o facilitador de conexão entre o programa e o banco (JDBC) permitindo fazer consultas e gerenciar o banco através do próprio NetBeans.

O padrão DAO é muito conveniente para manutenção e organização das regras de negocio.

2º Procedimento | Alimentando a Base

Todos os códigos solicitados neste roteiro de aula

```
package cadastrobd;
```

```
import cadastrobd.model.PessoaFisica;
```

```
import cadastrobd.model.PessoaJuridica;
```

```
import cadastrobd.model.PessoaFisicaDAO;
```

```
import cadastrobd.model.PessoaJuridicaDAO;
```

```
import java.util.ArrayList;
```

```
/**
```

```
*
```

```
* @author sendi
```

```
*/
```

```
public class CadastroBD {
```

```
    public static void main(String[] args) {
```

```
        try {
```

```
            Menu menu = new Menu();
```

```
            PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO( );
```

```
            PessoaJuridicaDAO pessoaJuridicaDAO= new PessoaJuridicaDAO( );
```

```
            while (true) {
```

```
                menu.telaInicial();
```

```
                menu.setTelaInicialResposta();
```

```
                switch( menu.getTelaInicialResposta() ) {
```

```
                    case "1" -> {
```

```
                        String tipoPessoa = menu.escolhaTipoPessoa();
```

```

        try{
            if( "F".equals(tipoPessoa) )
                pessoaFisicaDAO.incluir(menu.preenchePessoaFisica() );

            if( "J".equals(tipoPessoa) )
                pessoaJuridicaDAO.incluir(menu.preenchePessoaJuridica() );

        }catch (Exception e) {

            System.out.println("O sistema nao pode incluir a pessoa especificada!");

        }
    }

    case "2" -> {

        int id = menu.inputApenasNumeros( "ID" );

        String tipoPessoa = menu.escolhaTipoPessoa();

        try{

            if( "F".equals(tipoPessoa) ) pessoaFisicaDAO.alterar(
                id,menu.preenchePessoaFisica() );

            if( "J".equals(tipoPessoa) ) pessoaJuridicaDAO.alterar(
                id,menu.preenchePessoaJuridica() );

        }catch (Exception e) {

            System.out.println("O sistema nao pode alterar a pessoa especificada!");

        }
    }

    case "3" -> {

        int id = menu.inputApenasNumeros( "ID" );

        String tipoPessoa = menu.escolhaTipoPessoa();

        try{

            if( "F".equals(tipoPessoa) ) pessoaFisicaDAO.excluir( id );

            if( "J".equals(tipoPessoa) ) pessoaJuridicaDAO.excluir( id );

        } catch (Exception e) {

            System.out.println("O sistema nao pode excluir a pessoa especificada!");

        }
    }
}

```

```

case "4" -> {

    int id = menu.inputApenasNumeros( "ID" );

    String tipoPessoa = menu.escolhaTipoPessoa();

    try{

        if( "F".equals(tipoPessoa) ){

            PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id );

            pessoaFisica.exibir();

        }

        if( "J".equals(tipoPessoa) ){

            PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id );

            pessoaJuridica.exibir();

        }

    } catch (Exception e) {

        System.out.println("O sistema nao pode encontrar a pessoa
especificada!");

    }

}

case "5" -> {

    String tipoPessoa = menu.escolhaTipoPessoa();

    try{

        if( "F".equals(tipoPessoa) ){

            ArrayList<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();

            pessoasFisicas.stream().forEach( pessoa -> pessoa.exibir() );

        }

        if( "J".equals(tipoPessoa) ) {

            ArrayList<PessoaJuridica> pessoasJuridicas =
pessoaJuridicaDAO.getPessoas();

            pessoasJuridicas.stream().forEach( pessoa -> pessoa.exibir() );

        }

    }

}

```

```

        } catch (Exception e) {
            System.out.println("O sistema nao pode encontrar todas as pessoa!");
        }
    }

    case "0" -> System.exit(0);
    default -> System.out.println("Opcao invalida!");
}

}

} catch (IllegalStateException e) {
    // System.in has been closed
    System.out.println("System.in was closed; exiting");
}

}

}

```

```

package cadastrobd;

import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import java.util.Scanner;

public class Menu {
    private Scanner scanner;
    private final String NEW_LINE = System.getProperty("line.separator");
    private String telaInicialResposta;

    public Menu () {
        scanner = new Scanner( System.in );
    }
}

```

```

public void telaInicial() {

    System.out.println("===== " + NEW_LINE

        + "1 - Incluir Pessoa" + NEW_LINE

        + "2 - Alterar Pessoa" + NEW_LINE

        + "3 - Excluir Pessoa" + NEW_LINE

        + "4 - Buscar por ID" + NEW_LINE

        + "5 - Exibir Todos" + NEW_LINE

        + "0 - Finalizar Programa" + NEW_LINE

        + "=====");

//    ultimaResposta = scanner.nextLine();
}

public void setTelaInicialResposta( ) {

    telaInicialResposta = scanner.nextLine();
}

public String getTelaInicialResposta ( ) {

    return telaInicialResposta;
}

public String escolhaTipoPessoa() {

    while( true ) {

        System.out.println("F - Pessoa Fisica | J - Pessoa Juridica" );

        String opcao = scanner.nextLine().toUpperCase();

        switch( opcao ) {

            case "J", "F" -> {

```

```

        return opcao;
    }
    default -> System.out.println("Opcao invalida!");
}
}
}

```

```

public String escolhaPrefixo() {
    System.out.println("Favor Digitar o Prefixo do Arquivo:" );
    return scanner.nextLine();
}

```

```

public PessoaFisica preenchePessoaFisica() {

    System.out.println("Favor Digitar o Nome:" );
    String nome = scanner.nextLine();

    System.out.println("Favor Digitar o Logradouro:" );
    String logradouro = scanner.nextLine();

    System.out.println("Favor Digitar o Cidade:" );
    String cidade = scanner.nextLine();

    System.out.println("Favor Digitar o Estado:" );
    String estado = scanner.nextLine();

    System.out.println("Favor Digitar o Telefone:" );
    String telefone = scanner.nextLine();

    System.out.println("Favor Digitar o Email:" );

```

```
String email = scanner.nextLine();

System.out.println("Favor Digitar o CPF:" );
String cpf = scanner.nextLine();

return new PessoaFisica( 1, nome, logradouro,cidade,estado,telefone,email,cpf );

}
```

```
public PessoaJuridica preenchePessoaJuridica() {
```

```
System.out.println("Favor Digitar o Nome:" );
String nome = scanner.nextLine();
```

```
System.out.println("Favor Digitar o Logradouro:" );
String logradouro = scanner.nextLine();
```

```
System.out.println("Favor Digitar o Cidade:" );
String cidade = scanner.nextLine();
```

```
System.out.println("Favor Digitar o Estado:" );
String estado = scanner.nextLine();
```

```
System.out.println("Favor Digitar o Telefone:" );
String telefone = scanner.nextLine();
```

```
System.out.println("Favor Digitar o Email:" );
String email = scanner.nextLine();
```

```
System.out.println("Favor Digitar o CNPJ:" );
String CNPJ = scanner.nextLine();
```



```
        return new PessoaJuridica(2, nome, logradouro,cidade,estado,telefone,email, CNPJ );
    }
}
```

```
public int inputApenasNumeros( String nomeInput ){
    System.out.println("Favor Digitar o " + nomeInput + " (Somente Numeros):" );

    int reloop = 0;
    int result = 0;
    do {
        try {
            String input = scanner.nextLine(); // Scan the next line from System.in
            result = Integer.parseInt(input); // Try to parse it as an int
            reloop++;
        } catch (Exception e) {
            System.out.println("Por favor digite um NUMERO!");
        }
    } while (reloop == 0);

    return result;
}
}
```

```
public final static void limparConsole()
{
    try
    {
        final String os = System.getProperty("os.name");

        if (os.contains("Windows"))
        {
            Runtime.getRuntime().exec("cls");
        }
    }
}
```

```

    }

    else

    {

        Runtime.getRuntime().exec("clear");

    }

}

catch (final Exception e)

{

    // Handle any exceptions.

}

}

}

```

Métodos de Pessoa física

Incluir Pessoa

```

=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar por ID
5 - Exibir Todos
0 - Finalizar Programa
=====
1
F - Pessoa Fisica | J - Pessoa Juridica
f
Favor Digitar o Nome:
fabio
Favor Digitar o Logradouro:
lugar
Favor Digitar o Cidade:
cidade boa
Favor Digitar o Estado:
sp
Favor Digitar o Telefone:
5555-5555
Favor Digitar o Email:
joao@email.com
Favor Digitar o CPF:
9999999999
=====

```

```
=====
5
F - Pessoa Fisica | J - Pessoa Juridica
f
ID: 10
Nome: Joao
Logradouro: Rua 12, Casa 3,Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 111111111
ID: 1
Nome: fabio
Logradouro: lugar
Cidade: cidade boa
Estado: sp
Telefone: 5555-5555
Email: joao@email.com
CPF: 9999999999
=====
```

Alterar

```
-----
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar por ID
5 - Exibir Todos
0 - Finalizar Programa
=====
2
Favor Digitar o ID (Somente Numeros):
1
F - Pessoa Fisica | J - Pessoa Juridica
f
Favor Digitar o Nome:
paulo
Favor Digitar o Logradouro:
lugar nenhum\
Favor Digitar o Cidade:
cidade norte
Favor Digitar o Estado:
PA
Favor Digitar o Telefone:
6666-6666
Favor Digitar o Email:
paulao@email.com
Favor Digitar o CPF:
11111111111
```

```
=====
5
F - Pessoa Fisica | J - Pessoa Juridica
f
ID: 10
Nome: Joao
Logradouro: Rua 12, Casa 3,Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 111111111
ID: 1
Nome: paulo
Logradouro: lugar nenhum\
Cidade: cidade norte
Estado: PA
Telefone: 6666-6666
Email: paulao@email.com
CPF: 11111111111
=====
```

Excluir

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar por ID
5 - Exibir Todos
0 - Finalizar Programa
=====
3
Favor Digitar o ID (Somente Numeros):
1
F - Pessoa Fisica | J - Pessoa Juridica
f
=====
=====
5
F - Pessoa Fisica | J - Pessoa Juridica
f
ID: 10
Nome: Joao
Logradouro: Rua 12, Casa 3,Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 111111111
=====
```

Buscar por ID

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar por ID
5 - Exibir Todos
0 - Finalizar Programa
=====
4
Favor Digitar o ID (Somente Numeros):
10
F - Pessoa Fisica | J - Pessoa Juridica
f
ID: 10
Nome: Joao
Logradouro: Rua 12, Casa 3,Quitanda
Cidade: Riacho do Sul
Estado: PA
Telefone: 1111-1111
Email: joao@riacho.com
CPF: 111111111
=====
```

Métodos de Pessoa Jurídica

Incluir

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar por ID
5 - Exibir Todos
0 - Finalizar Programa
=====
1
F - Pessoa Fisica | J - Pessoa Juridica
j
Favor Digitar o Nome:
LLO
Favor Digitar o Logradouro:
Rua das Nozes
Favor Digitar o Cidade:
Pipoqueira Gigante
Favor Digitar o Estado:
AC
Favor Digitar o Telefone:
8888-8888
Favor Digitar o Email:
LLO@EMAIL.COM
Favor Digitar o CNPJ:
66666666666666
=====
```

Alterar

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar por ID
5 - Exibir Todos
0 - Finalizar Programa
=====

2
Favor Digitar o ID (Somente Numeros):
1
F - Pessoa Fisica | J - Pessoa Juridica
j
Favor Digitar o Nome:
IIO
Favor Digitar o Logradouro:
PALMES
Favor Digitar o Cidade:
Cidade dos loucos
Favor Digitar o Estado:
AC
Favor Digitar o Telefone:
1234-4321
Favor Digitar o Email:
IIO@EMAIL.COM
Favor Digitar o CNPJ:
44444444444444
=====

=====

5
F - Pessoa Fisica | J - Pessoa Juridica
j
ID: 20
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CPF: 22222222222222
ID: 1
Nome: IIO
Logradouro: PALMES
Cidade: Cidade dos loucos
Estado: AC
Telefone: 1234-4321
Email: IIO@EMAIL.COM
CPF: 44444444444444
=====
```

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar por ID
5 - Exibir Todos
0 - Finalizar Programa
=====
3
Favor Digitar o ID (Somente Numeros):
1
F - Pessoa Fisica | J - Pessoa Juridica
j
=====
```

```
=====
5
F - Pessoa Fisica | J - Pessoa Juridica
j
ID: 20
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CPF: 222222222222
=====
```

Buscar por ID

```
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar por ID
5 - Exibir Todos
0 - Finalizar Programa
=====
4
Favor Digitar o ID (Somente Numeros):
20
F - Pessoa Fisica | J - Pessoa Juridica
j
ID: 20
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CPF: 222222222222
=====
```

*Exibir todos estão nos métodos.

Análise

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

R: A persistência de dados converte os dados para um arquivo não estruturado, não permitindo análises ou comparações, sendo apenas dados brutos.

A persistência em banco de dados insere os dados em tabelas segmentando-os em cada coluna deixando-as organizadas. Com as tabelas podemos realizar pesquisas, análises e comparações permitindo que os dados transformam em informações.

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

R: Permitindo que não precise de chamada convencional de um método, em uma linha já se obtém o parâmetro, a função e o retorno.

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

R: O atributo Static permite a chamada de métodos sem precisar instanciar a classe.