Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.

André Luis Gonçalves Carvalhal - Matrícula: 20220318540

Campus Barra World - Desenvolvimento Full-Stack
RPG0016 - BackEnd sem banco não tem – 9001 – 2023.3

Objetivo 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.**

Link GitHub
https://github.com/ANDREC1986/RPG0016-202203185403

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Pessoa.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrodb.model;

/**
 *
 * @author andre
 */
public class Pessoa {
    private int id;
    private String nome, logradouro, cidade, estado, telefone, email;

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

    public void exibir() {
    System.out.println("id: "+this.id);
    System.out.println("nome: "+this.nome);
    System.out.println("logradouro: "+this.logradouro);
    System.out.println("cidade: "+this.cidade);
    System.out.println("estado: "+this.estado);
    System.out.println("telefone: "+this.telefone);
    System.out.println("email: "+this.email);

    }

    public int getId() {
        return id;
    }

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

```
}

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 getTelefone() {
    return telefone;
}

public void setTelefone(String telefone) {
    this.telefone = telefone;
}

public String getEmail() {
    return email;
}

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

```
    }

}
```

PessoaFisica.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrodb.model;

/**
 *
 * @author andre
 */
public class PessoaFisica extends Pessoa {
    private String cpf;

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

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF :"+this.cpf);
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

}
```

PessoaJuridica.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrodb.model;

/**
 *
 * @author andre
 */
public class PessoaJuridica extends Pessoa {
    private String cnpj;

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

    @Override
    public void exibir(){
        super.exibir();
        System.out.println("CNPJ: "+this.cnpj);
    }

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

}
```

PessoaFisicaDAO.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrodb.model;

import cadastrodb.model.util.ConectorDB;
```

```
import cadastrodb.model.util.SequenceManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author andre
 */
public class PessoaFisicaDAO {

    public static PessoaFisica getPessoa(int id){
        try {
            Connection con = ConectorDB.getConnection();
            PreparedStatement verify = ConectorDB.getPrepared(con, "SELECT * FROM pessoa_fisica
WHERE idPessoa = ?");
            verify.setInt(1, id);
            ResultSet cpf = ConectorDB.getSelect(verify);
            if(!cpf.next()) {
                ConectorDB.close(cpf, verify, con);
            } else {
                PreparedStatement pessoa = ConectorDB.getPrepared(con, "SELECT * FROM pessoa
WHERE idPessoa = ?");
                pessoa.setInt(1, id);
                ResultSet pessoaDados = ConectorDB.getSelect(pessoa);
                pessoaDados.next();
                PessoaFisica getPessoa = new
PessoaFisica(cpf.getString("cpf"),pessoaDados.getInt("idPessoa"),pessoaDados.getString("nome"),
pessoaDados.getString("logradouro"),pessoaDados.getString("cidade"),pessoaDados.getString("est
ado"),pessoaDados.getString("telefone"),pessoaDados.getString("email"));
                ConectorDB.close(pessoaDados);
                ConectorDB.close(pessoa);
                ConectorDB.close(cpf, verify, con);
                return getPessoa;
            }

        } catch (SQLException ex) {
            System.out.println("Nao foi possivel conectar!");
        }
        return new PessoaFisica(null,0,null,null,null,null,null,null) {
        @Override
        public void exibir() {
            System.out.println("Pessoa Fisica nao encontrada!");
        }
```

```
        };
    }

    public static ArrayList<PessoaFisica> getPessoas() {
        ArrayList<PessoaFisica> getPessoas = new ArrayList();
        try {
            Connection con = ConectorDB.getConnection();
            PreparedStatement smt = ConectorDB.getPrepared(con, "SELECT idPessoa FROM
pessoa_fisica");
            ResultSet idPessoas = ConectorDB.getSelect(smt);
            while(idPessoas.next()){
                getPessoas.add(getPessoa(idPessoas.getInt("idPessoa")));
            }
            ConectorDB.close(idPessoas, smt, con);

        } catch (SQLException ex) {
            System.out.println("Nao foi possivel conectar");
        }
        return getPessoas;
    }

    public static void incluir(PessoaFisica pessoaObj){
        try {
            int pessoaID = SequenceManager.getValue();
            Connection con = ConectorDB.getConnection();
            con.setAutoCommit(false);

            PreparedStatement pessoa = ConectorDB.getPrepared(con, "INSERT INTO pessoa VALUES
(?,?,?,?,?,?,?)");
            PreparedStatement pessoafisica = ConectorDB.getPrepared(con, "INSERT INTO
pessoa_fisica VALUES (?,?)");
            pessoa.setInt(1, pessoaID);
            pessoa.setString(2, pessoaObj.getNome());
            pessoa.setString(3, pessoaObj.getLogradouro());
            pessoa.setString(4, pessoaObj.getCidade());
            pessoa.setString(5, pessoaObj.getEstado());
            pessoa.setString(6, pessoaObj.getTelefone());
            pessoa.setString(7, pessoaObj.getEmail());
            pessoafisica.setInt(1, pessoaID);
            pessoafisica.setString(2, pessoaObj.getCpf());
            try {
                pessoa.execute();
                pessoafisica.execute();
                con.commit();
                System.out.println("Inserido com sucesso!");
            } catch (SQLException ex) {
                con.rollback();
                System.out.println("Falha ao inserir dados!");
```

```
            } finally {
                ConectorDB.close(pessoafisica);
                ConectorDB.close(pessoa);
                ConectorDB.close(con);
            }

        } catch (SQLException ex) {
            System.out.println("Nao foi possivel conectar.");
        }



    }

    public static void alterar(int id, String param, String value) {
        try {
            Connection con = ConectorDB.getConnection();
            String sql = "";
            if("cpf" == param){
            sql = "UPDATE pessoa_fisica SET ?=? WHERE idPessoa=?";
            } else { sql = "UPDATE pessoa SET "+param+"=? WHERE idPessoa=?";}
            PreparedStatement smt = ConectorDB.getPrepared(con, sql);
            smt.setString(1, value);
            smt.setInt(2, id);
            try {
                smt.executeUpdate();
                con.commit();
                System.out.println("Atualizado com sucesso!");
            } catch (SQLException ex) {
                System.out.println("Nao foi possivel atualizar!");
            } finally {
                ConectorDB.close(smt);
                ConectorDB.close(con);
            }


        } catch (SQLException ex) {
            Logger.getLogger(PessoaFisicaDAO.class.getName()).log(Level.SEVERE, null, ex);
            System.out.println("Nao foi possivel conectar!");
        }
    }

    public static void excluir(int id){
        try {
            Connection con = ConectorDB.getConnection();
            con.setAutoCommit(false);
            PreparedStatement delete_pessoa = ConectorDB.getPrepared(con, "DELETE FROM pessoa
WHERE idPessoa=?");
```

```
            PreparedStatement delete_cpf = ConectorDB.getPrepared(con, "DELETE FROM
pessoa_fisica WHERE idPessoa=?");
            delete_pessoa.setInt(1, id);
            delete_cpf.setInt(1, id);
            try {
                delete_cpf.executeUpdate();
                delete_pessoa.executeUpdate();
                con.commit();
                System.out.println("Excluido com Sucesso!");
            } catch (SQLException ex) {
                System.out.println("Nao foi possivel deletar! Verifique a ID!");
            } finally {
                ConectorDB.close(delete_cpf);
                ConectorDB.close(delete_pessoa);
                ConectorDB.close(con);
            }
        } catch (SQLException ex) {
            System.out.println("Nao foi possivel conectar!");
        }


    }

    public static void main(String[] args) {
        getPessoas().forEach((e) -> e.exibir());

    }
}
```

PessoaJuridicaDAO.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrodb.model;

import cadastrodb.model.util.ConectorDB;
import cadastrodb.model.util.SequenceManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
/**
 *
 * @author andre
 */
public class PessoaJuridicaDAO {

       public static PessoaJuridica getPessoa(int id){
       try {
          Connection con = ConectorDB.getConnection();
          PreparedStatement verify = ConectorDB.getPrepared(con, "SELECT * FROM
pessoa_juridica WHERE idPessoa = ?");
          verify.setInt(1, id);
          ResultSet cnpj = ConectorDB.getSelect(verify);
          if(!cnpj.next()) {
             ConectorDB.close(cnpj, verify, con);
          } else {
             PreparedStatement pessoa = ConectorDB.getPrepared(con, "SELECT * FROM pessoa
WHERE idPessoa = ?");
             pessoa.setInt(1, id);
             ResultSet pessoaDados = ConectorDB.getSelect(pessoa);
             pessoaDados.next();
             PessoaJuridica getPessoa = new
PessoaJuridica(cnpj.getString("cnpj"),pessoaDados.getInt("idPessoa"),pessoaDados.getString("no
me"),pessoaDados.getString("logradouro"),pessoaDados.getString("cidade"),pessoaDados.getStrin
g("estado"),pessoaDados.getString("telefone"),pessoaDados.getString("email"));
             ConectorDB.close(pessoaDados);
             ConectorDB.close(pessoa);
             ConectorDB.close(cnpj, verify, con);
             return getPessoa;
          }

       } catch (SQLException ex) {
          System.out.println("Nao foi possivel conectar!");
       }
       return new PessoaJuridica(null,0,null,null,null,null,null,null) {
       @Override
       public void exibir() {
          System.out.println("Pessoa Juridica nao encontrada!");
       }
       };
    }

    public static ArrayList<PessoaJuridica> getPessoas() {
       ArrayList<PessoaJuridica> getPessoas = new ArrayList();
       try {
          Connection con = ConectorDB.getConnection();
          PreparedStatement smt = ConectorDB.getPrepared(con, "SELECT idPessoa FROM
```

```
pessoa_juridica");
          ResultSet idPessoas = ConectorDB.getSelect(smt);
          while(idPessoas.next()){
             getPessoas.add(getPessoa(idPessoas.getInt("idPessoa")));
          }
          ConectorDB.close(idPessoas, smt, con);

       } catch (SQLException ex) {
          System.out.println("Nao foi possivel conectar");
       }
       return getPessoas;
    }

    public static void incluir(PessoaJuridica pessoaObj){
       try {
          int pessoaID = SequenceManager.getValue();
          Connection con = ConectorDB.getConnection();
          con.setAutoCommit(false);
          PreparedStatement pessoa = ConectorDB.getPrepared(con, "INSERT INTO pessoa VALUES
(?,?,?,?,?,?,?)");
          PreparedStatement PessoaJuridica = ConectorDB.getPrepared(con, "INSERT INTO
pessoa_juridica VALUES (?,?)");
          pessoa.setInt(1, pessoaID);
          pessoa.setString(2, pessoaObj.getNome());
          pessoa.setString(3, pessoaObj.getLogradouro());
          pessoa.setString(4, pessoaObj.getCidade());
          pessoa.setString(5, pessoaObj.getEstado());
          pessoa.setString(6, pessoaObj.getTelefone());
          pessoa.setString(7, pessoaObj.getEmail());
          PessoaJuridica.setInt(1, pessoaID);
          PessoaJuridica.setString(2, pessoaObj.getCnpj());
          try {
             pessoa.execute();
             PessoaJuridica.execute();
             con.commit();
             System.out.println("Inserido com sucesso!");
          } catch (SQLException ex) {
             con.rollback();
             System.out.println("Falha ao inserir dados!");
          } finally {
             ConectorDB.close(PessoaJuridica);
             ConectorDB.close(pessoa);
             ConectorDB.close(con);
          }

       } catch (SQLException ex) {
          Logger.getLogger(PessoaJuridicaDAO.class.getName()).log(Level.SEVERE, null, ex);
       }
```

```
    }

    public static void alterar(int id, String param, String value) {
        try {
            Connection con = ConectorDB.getConnection();
            String sql = "";
            if("cnpj" == param){
            sql = "UPDATE pessoa_juridica SET ?=? WHERE idPessoa=?";
            } else { sql = "UPDATE pessoa SET "+param+"=? WHERE idPessoa=?";}
            PreparedStatement smt = ConectorDB.getPrepared(con, sql);
            smt.setString(1, value);
            smt.setInt(2, id);
            try {
                smt.executeUpdate();
                con.commit();
                System.out.println("Atualizado com sucesso!");
            } catch (SQLException ex) {
                System.out.println("Nao foi possivel atualizar!");
            } finally {
                ConectorDB.close(smt);
                ConectorDB.close(con);
            }


        } catch (SQLException ex) {
            Logger.getLogger(PessoaJuridicaDAO.class.getName()).log(Level.SEVERE, null, ex);
            System.out.println("Nao foi possivel conectar!");
        }
    }

    public static void excluir(int id){
        try {
            Connection con = ConectorDB.getConnection();
            con.setAutoCommit(false);
            PreparedStatement delete_pessoa = ConectorDB.getPrepared(con, "DELETE FROM pessoa
WHERE idPessoa=?");
            PreparedStatement delete_cnpj = ConectorDB.getPrepared(con, "DELETE FROM
pessoa_juridica WHERE idPessoa=?");
            delete_pessoa.setInt(1, id);
            delete_cnpj.setInt(1, id);
            try {
                delete_cnpj.executeUpdate();
                delete_pessoa.executeUpdate();
                con.commit();
                System.out.println("Excluido com Sucesso!");
```

```
            } catch (SQLException ex) {
                System.out.println("Nao foi possivel deletar! Verifique a ID!");
            } finally {
                ConectorDB.close(delete_cnpj);
                ConectorDB.close(delete_pessoa);
                ConectorDB.close(con);
            }
        } catch (SQLException ex) {
            System.out.println("Nao foi possivel conectar!");
        }


    }

    public static void main(String[] args) {
        getPessoa(1).exibir();

    }
}
```

ConectorDB.java

```
/* * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrodb.model.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

/**
 *
 * @author andre
 */
```

```
public class ConectorDB {

    public static Connection getConnection() throws SQLException {
        return
DriverManager.getConnection("jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;t
rustServerCertificate=true;","loja","loja");
    }

    public static PreparedStatement getPrepared(Connection con, String sql) throws SQLException{
        PreparedStatement smt = con.prepareStatement(sql);
        return smt;
    }

    public static ResultSet getSelect(PreparedStatement prep) throws SQLException {
        ResultSet result = prep.executeQuery();
        return result;
    }

    public static void close(PreparedStatement smt){
        try {
            smt.close();
        } catch (SQLException e) {
            System.out.print(e);
        }
    }

        public static void close(Connection connection){
        try {
            connection.close();
        } catch (SQLException e) {
            System.out.print(e);
        }
    }

        public static void close(ResultSet result){
        try {
            result.close();
        } catch (SQLException e) {
            System.out.print(e);
        }
    }

        public static void close(ResultSet result, PreparedStatement prep, Connection con){
            close(result);
            close(prep);
            close(con);
        }
}
```

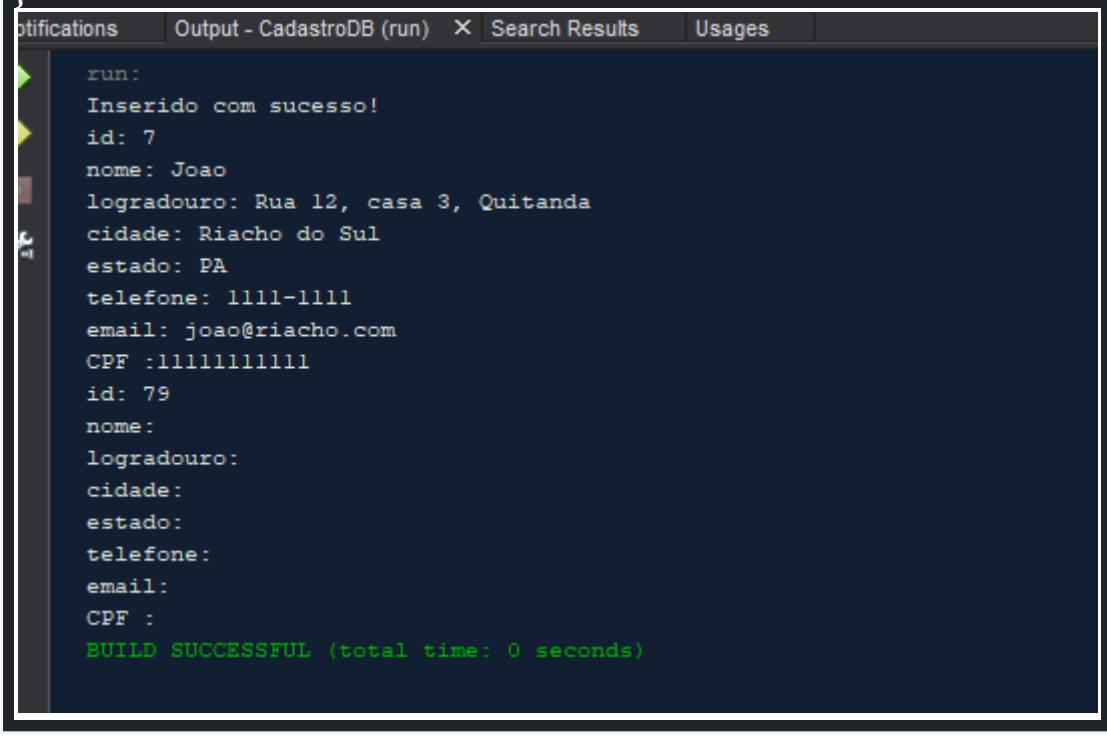
SequenceManager.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrodb.model.util;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

/**
 *
 * @author andre
 */
public class SequenceManager {
    public static int getValue() throws SQLException{
        Connection con = ConectorDB.getConnection();
        PreparedStatement smt = ConectorDB.getPrepared(con,"SELECT NEXT VALUE FOR
idPessoa_SEQ AS VALOR");
        ResultSet result = ConectorDB.getSelect(smt);
        result.next();
        return result.getInt("VALOR");
    }
}
```

CadastroDBTeste.java

```
package cadastrodb;

import cadastrodb.model.PessoaFisica;
impor
t cadastrodb.model.PessoaFisicaDAO;
import cadastrodb.model.PessoaJuridica;
import cadastrodb.model.PessoaJuridicaDAO;
import java.util.ArrayList;

/**
 *
 * @author andre
 */
public class CadastroDBTeste {
        public static void main(String[] args) {
```

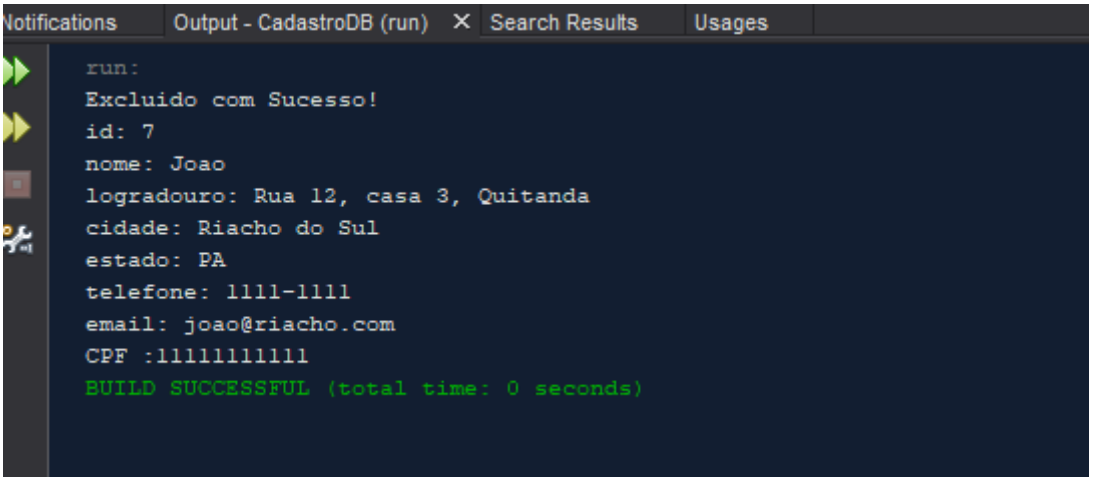
```
        PessoaFisica tester = new PessoaFisica("",0,"","","","","","");
        PessoaFisicaDAO.incluir(tester);
        ArrayList<PessoaFisica> pessoas = PessoaFisicaDAO.getPessoas();
        pessoas.forEach((e) -> e.exibir());
}
```



CadastroDBTeste.java

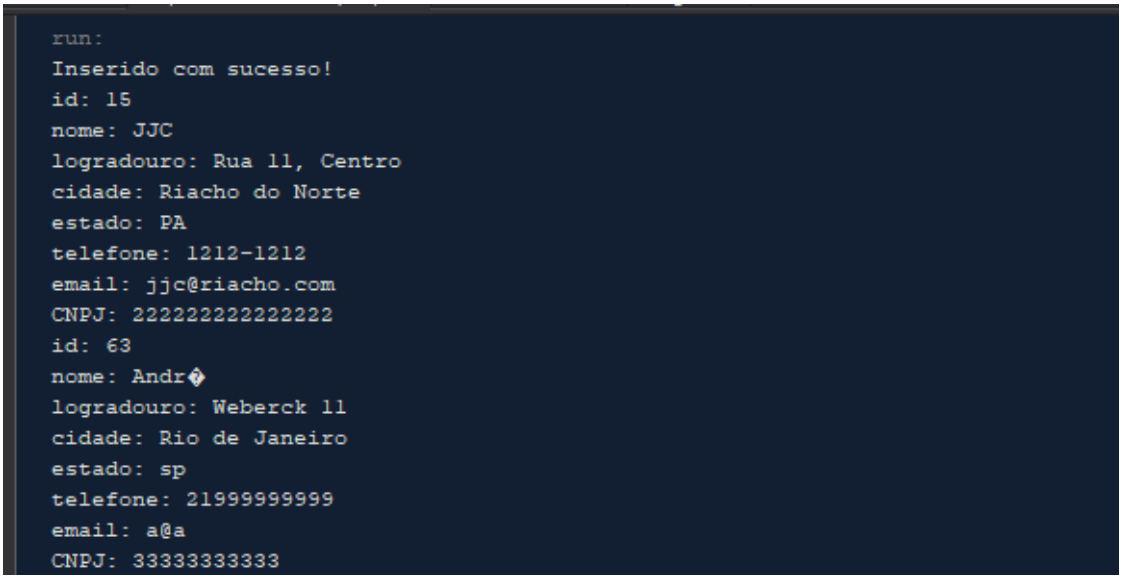
```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastrodb;

import cadastrodb.model.PessoaFisica;
import cadastrodb.model.PessoaFisicaDAO;
import cadastrodb.model.PessoaJuridica;
import cadastrodb.model.PessoaJuridicaDAO;
import java.util.ArrayList;

/**
 *
 * @author andre
 */
public class CadastroDBTeste {
    public static void main(String[] args) {
```

```
        PessoaFisicaDAO.excluir(79);
        ArrayList<PessoaFisica> pessoas = PessoaFisicaDAO.getPessoas();
        pessoas.forEach((e) -> e.exibir());
}
```



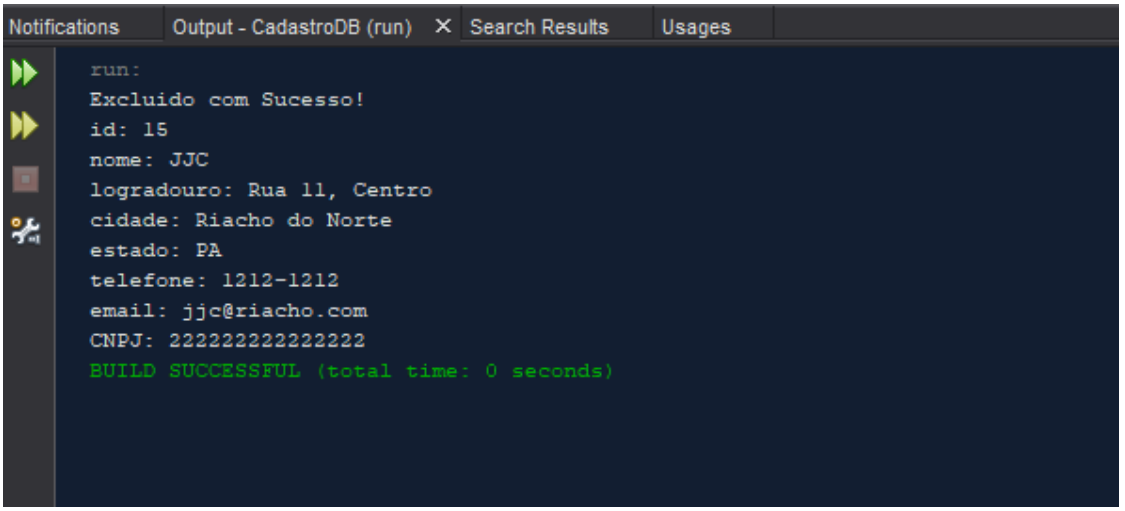
CadastroDBTeste.java

```
package cadastrodb;

import cadastrodb.model.PessoaFisica;
impor
t cadastrodb.model.PessoaFisicaDAO;
import cadastrodb.model.PessoaJuridica;
import cadastrodb.model.PessoaJuridicaDAO;
import java.util.ArrayList;

/**
 *
 * @author andre
 */
public class CadastroDBTeste {
        public static void main(String[] args) {
            PessoaJuridica teste = new PessoaJuridica("33333333333",0,"André","Weberck 11","Rio de
Janeiro","RJ","21999999999","a@a");
            PessoaJuridicaDAO.incluir(teste);
            ArrayList<PessoaJuridica> pessoas = PessoaJuridicaDAO.getPessoas();
            pessoas.forEach((e) -> e.exibir());
}
```



CadastroDBTeste.java

```
package cadastrodb;

import cadastrodb.model.PessoaFisica;
impor
t cadastrodb.model.PessoaFisicaDAO;
import cadastrodb.model.PessoaJuridica;
import cadastrodb.model.PessoaJuridicaDAO;
import java.util.ArrayList;

/**
 *
 * @author andre
 */
public class CadastroDBTeste {
        public static void main(String[] args) {
            PessoaJuridicaDAO.excluir(63);
            ArrayList<PessoaJuridica> pessoas = PessoaJuridicaDAO.getPessoas();
            pessoas.forEach((e) -> e.exibir());
}
```



Conclusão:

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

Um middleware permite a abstração do código, sendo ele responsável a servir de camada intermediária entre processos. No caso do JDBC ele ocupa a função de conectividade com o banco de dados, permitindo o desenvolvimento do sofware que poderar se conectar de forma uniforme com diversos SGDBs.

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

A diferença é que o Statement é usado para executar uma instrução SQL simples, não sendo pré-compilada, tende a ser mais vulnerável a ataques SQL Injection e deve ser utilizada no caso de consultas fixas ao banco. Já um PreparedStatement é pré-compilado, e executa uma instrução parametrizada, onde os atributos são parâmetros fornecidos para o mesmo, sendo está muito mais segura contra ataques SQL Injection, sendo esta recomendada para o uso em consultas dinâmicas.

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

Ele facilita a manutenção do código pois centraliza a lógica de acesso aos dados, organizando-as e permitindo a fácil manutenção e correção de bugs.

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

A herança é refletida na estrutura de modelagem do banco de dados, refletido em uma relação 1x1.

2º Procedimento | Alimentando a Base

CadastroDB.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package cadastrodb;

import cadastrodb.model.PessoaFisica;
import cadastrodb.model.PessoaFisicaDAO;
import cadastrodb.model.PessoaJuridica;
import cadastrodb.model.PessoaJuridicaDAO;
import java.util.ArrayList;
import java.util.Scanner;

/**
 *
 * @author andre
 */
public class CadastroDB {
        private static boolean mainloop = true;
        private static boolean navigate = false;
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        do {
                navigate = true;
                System.out.println("""
                                   ================================
                                   1 - Incluir Pessoa
                                   2 - Alterar Pessoa1
                                   3 - Excluir Pessoa
                                   4 - Buscar pelo Id
                                   5 - Exibir Todos
                                   0 - Finalizar Programa
                                   ================================
                                   """);
                Scanner select = new Scanner(System.in);
                String menu = select.nextLine();
                switch(menu) {
                    case "1" -> {
                    String tipo = tipo();
```

```
Scanner pessoa = new Scanner(System.in);
switch(tipo.toLowerCase()) {
   case "f" -> {
      PessoaFisica nova = new PessoaFisica("",0,"","","","","","");
      System.out.println("Informe o Nome");
      nova.setNome(pessoa.nextLine());
      System.out.println("Informe o CPF");
      nova.setCpf(pessoa.nextLine());
      System.out.println("Informe o Logradouro");
      nova.setLogradouro(pessoa.nextLine());
      System.out.println("Informe o Cidade");
      nova.setCidade(pessoa.nextLine());
      System.out.println("Informe o Estado");
      nova.setEstado(pessoa.nextLine());
      System.out.println("Informe o Telefone");
      nova.setTelefone(pessoa.nextLine());
      System.out.println("Informe o E-mail");
      nova.setEmail(pessoa.nextLine());
      PessoaFisicaDAO.incluir(nova);
      pause();
   }
   case "j" -> {
      PessoaJuridica nova = new PessoaJuridica("",0,"","","","","","");
      System.out.println("Informe o Nome");
      nova.setNome(pessoa.nextLine());
      System.out.println("Informe o CPF");
      nova.setCnpj(pessoa.nextLine());
      System.out.println("Informe o Logradouro");
      nova.setLogradouro(pessoa.nextLine());
      System.out.println("Informe o Cidade");
      nova.setCidade(pessoa.nextLine());
      System.out.println("Informe o Estado");
      nova.setEstado(pessoa.nextLine());
      System.out.println("Informe o Telefone");
      nova.setTelefone(pessoa.nextLine());
      System.out.println("Informe o E-mail");
      nova.setEmail(pessoa.nextLine());
      PessoaJuridicaDAO.incluir(nova);
      pause();
   }
}
}
case "2" -> {
   String tipo = tipo();
   Scanner id = new Scanner(System.in);
   switch(tipo.toLowerCase()) {
      case "f" -> {
         System.out.println("Informe o ID");
```

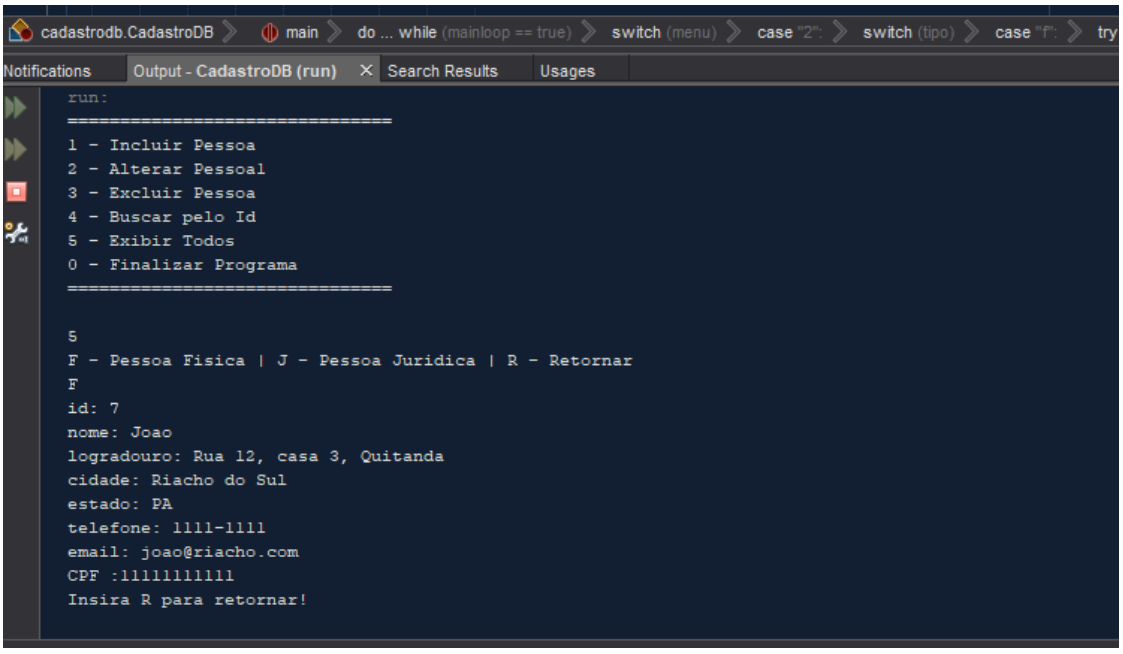
```
                    String pessoaid = id.nextLine();
                    try {
                       Integer.parseInt(pessoaid);
                    } catch(NumberFormatException ex) {
                       System.out.println("ID Invalida");
                       break;
                    }
                    PessoaFisica editar = PessoaFisicaDAO.getPessoa(Integer.parseInt(pessoaid));
                    System.out.println("Qual dado deseja editar? Nome, CPF, Cidade, Estado,
Telefone ou Email");
                    Scanner getParam = new Scanner(System.in);
                    String param = getParam.nextLine().toLowerCase();
                    Scanner getValue = new Scanner(System.in);
                    String value = "";
                    switch(param) {
                       case "nome" -> {
                          System.out.println("Insira o nome:");
                          value = getValue.nextLine();
                       }
                       case "cpf" -> {
                          System.out.println("Insira o CPF (11 Digitos):");
                           value = getValue.nextLine();
                       }
                       case "cidade" -> {
                          System.out.println("Insira a Cidade:");
                           value = getValue.nextLine();
                       }
                       case "estado" -> {
                          System.out.println("Insira o estado (2 Characteres):");
                          value = getValue.nextLine();
                       }
                       case "telefone" -> {
                          System.out.println("Insira a telefone (11 Digitos):");
                          value = getValue.nextLine();
                       }
                       case "email" -> {
                          System.out.println("Insira o Email:");
                          value = getValue.nextLine();
                       }
                       default -> {
                          System.out.println("Dado Invalido!");
                          pause();
                       }
                    }
                    PessoaFisicaDAO.alterar(Integer.parseInt(pessoaid), param, value);
                    pause();
                 }
                 case "j" -> {
```

```
                    System.out.println("Informe o ID");
                    String pessoaid = id.nextLine();
                    try {
                       Integer.parseInt(pessoaid);
                    } catch(NumberFormatException ex) {
                       System.out.println("ID Invalida");
                       break;
                    }
                    PessoaFisica editar = PessoaFisicaDAO.getPessoa(Integer.parseInt(pessoaid));
                    System.out.println("Qual dado deseja editar? Nome, CNPJ, Cidade, Estado,
Telefone ou Email");
                    Scanner getParam = new Scanner(System.in);
                    String param = getParam.nextLine().toLowerCase();
                    Scanner getValue = new Scanner(System.in);
                    String value = "";
                    switch(param) {
                       case "nome" -> {
                          System.out.println("Insira o nome:");
                          value = getValue.nextLine();
                       }
                       case "cpf" -> {
                          System.out.println("Insira o CNPJ (15 Digitos):");
                           value = getValue.nextLine();
                       }
                       case "cidade" -> {
                          System.out.println("Insira a Cidade:");
                           value = getValue.nextLine();
                       }
                       case "estado" -> {
                          System.out.println("Insira o estado (2 Characteres):");
                          value = getValue.nextLine();
                       }
                       case "telefone" -> {
                          System.out.println("Insira a telefone (11 Digitos):");
                          value = getValue.nextLine();
                       }
                       case "email" -> {
                          System.out.println("Insira o Email:");
                          value = getValue.nextLine();
                       }
                       default -> {
                          System.out.println("Dado Invalido!");
                          pause();
                       }
                    }
                    PessoaFisicaDAO.alterar(Integer.parseInt(pessoaid), param, value);
                    pause();
                 }
```

```
    }
}
case "3" -> {
String tipo = tipo();
System.out.println("Informe o ID");
Scanner getid = new Scanner(System.in);
String id = getid.nextLine();
switch(tipo.toLowerCase()){
    case "f" -> {
        try {
            Integer.parseInt(id);
            } catch(NumberFormatException ex) {
                System.out.println("ID Invalida");
                break;
            }
        PessoaFisicaDAO.excluir(Integer.parseInt(id));
        pause();
    }
    case "j" -> {
        try {
                Integer.parseInt(id);
            } catch(NumberFormatException ex) {
                System.out.println("ID Invalida");
                break;
            }
        PessoaJuridicaDAO.excluir(Integer.parseInt(id));
        pause();
    }
}
}
case "4" -> {
    String tipo = tipo();
    Scanner getid = new Scanner(System.in);
    System.out.println("Informe o ID:");
    String id = getid.nextLine();
        try {
            Integer.parseInt(id);
            } catch(NumberFormatException ex) {
                System.out.println("ID Invalida");
                break;
            }
    switch(tipo) {
        case "f" -> {
             PessoaFisica alvo = PessoaFisicaDAO.getPessoa(Integer.parseInt(id));
             alvo.exibir();
             pause();
        }
        case "j" -> {
```

```
                    PessoaJuridica alvo = PessoaJuridicaDAO.getPessoa(Integer.parseInt(id));
                    alvo.exibir();
                    pause();
                }

            }

        }
        case "5" -> {
        String tipo = tipo();
        switch(tipo.toLowerCase()) {
            case "f" -> {
            ArrayList<PessoaFisica> pessoasfisicas = PessoaFisicaDAO.getPessoas();
            pessoasfisicas.forEach((e) -> e.exibir());
            pause();
            }
            case "j" -> {
            ArrayList<PessoaJuridica> pessoasjuridicas = PessoaJuridicaDAO.getPessoas();
            pessoasjuridicas.forEach((e) -> e.exibir());
            pause();
            }
        }
        }
        case "0" -> {
        mainloop = false;
        }

      }
   } while (mainloop == true);
}



public static String tipo(){
   String tipo = null;
   System.out.println("F - Pessoa Fisica | J - Pessoa Juridica | R - Retornar");
   while(tipo == null) {
      Scanner select = new Scanner(System.in);
      String opt = select.nextLine();
      switch(opt) {
         case "f" -> {
         tipo = "f";
         }
         case "j" -> {
         tipo = "j";
         }
         case "r" ->{
         navigate = false;
```

```
            tipo = "quit";
            }
            default -> {
            System.out.println("Tipo invalido!");
            }
        }
    }
    return tipo;
  }

  public static void pause() {
    System.out.println("Insira R para retornar!");
    Scanner imput = new Scanner(System.in);
    while (navigate == true)
    {
      if("r".equals(imput.nextLine().toLowerCase())) {
      navigate = false;
      }
    }
  }
}
```



Conclusão:

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

As principais diferenças são a forma de armazenamento, segurança, simplicidade e escalabilidade. Sendo que o armazenamento em banco de dados permite fácil escalabilidade do projeto, enquanto em arquivo está é muito mais limitada.

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

O operador lambda permite a execução de uma mesma função anônima para todos os dados de um array/arraylist. Por exemplo no projeto, é obtido um ArrayList de PessoasFisicas/Juridicas e é utilizado o lambda, para que seja feita a exibição dos dados de cada objeto no array.

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

Por que o static é o marcador para que o compilador do java saiba que o determinado método pode ser invocado sem que um objeto da classe tenha sido instanciado.