**Jonathan Domingos Carneiro da Silva**
**matricula: 2023.04.12744-1**
**3274 POLO CENTRO - ITAITINGA – CE**

# Nível 3: Back-end Sem Banco Não Tem

**– DESENVOLVIMENTO FULL STACK 2023.1 – 3º Semestre – 2024.2**

**Criação de aplicativo Java, com acesso ao banco de dados SQL Server através do middleware JDBC.**

**Códigos usados nesse primeiro procedimento:**

**CadastroBD**

**Cadastro.BD.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 cadastrobd;



import
java.sql.SQLException;import
java.util.ArrayList;


import cadastrobd.model.Pessoa;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.util.ConectorBD;
```

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

/**
 *
 * @author Usuario
 */
public class CadastroBD {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws SQLException
        {PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();

        PessoaFisica PessoaF = new PessoaFisica("11122233388", 1, "Ivan dos Santos","Rua
        das Flores", "Quissama", "RJ", "1137968555", "ivansantos@ivan.com.br");
        pfDAO.incluir(PessoaF);



    }

}
```

**CadastroBDTeste.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 cadastrobd;
```

```
import java.sql.*;

import  cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;
import cadastrobd.model.util.SequenceManager;
import java.util.ArrayList;

/**
 *
 * @author Usuario
 */
public class CadastroBDTeste {

    public static void main(String[] args) throws SQLException {
        PessoaFisicaDAO PFisicaDao = new PessoaFisicaDAO();
        SequenceManager sequenceCode = new SequenceManager();int
        nextValue = sequenceCode.getValue("CodigoPessoa");

        //Instanciar uma pessoa física e persistir no banco de dados.
        PessoaFisica PessoaF = new PessoaFisica("00099900099", nextValue, "Pedro de Souza",
                "Rua das Flores", "Quissama", "RJ", "2512968000", "pedro@gmail.com");
        PFisicaDao.incluir(PessoaF);

        //Alterar os dados da pessoa física no banco.
        PessoaF.setCidade("Macae");
        PessoaF.setEstado("RJ");
        PFisicaDao.alterar(PessoaF);
```

```
        //Consultar todas as pessoas físicas do banco de dados e listar no console.
        ArrayList<PessoaFisica> resultadoP = PFisicaDao.getPessoas();
        resultadoP.forEach(item -> item.exibir());

        //Excluir a pessoa física criada anteriormente no banco.
        PFisicaDao.excluir(PessoaF);
        PFisicaDao.close();

        PessoaJuridicaDAO PJuridicaDao = new PessoaJuridicaDAO();
        nextValue = sequenceCode.getValue("CodigoPessoa");
        //Instanciar uma pessoa jurídica e persistir no banco de dados.
        PessoaJuridica PessoaJ = new PessoaJuridica("00333999000199", nextValue, "HappySalad
ltda", "Rua Mendes n003",
                "Macae", "RJ", "6540662500", "happysalad@happysalas.com.br");
        PJuridicaDao.incluir(PessoaJ);

        //Alterar os dados da pessoa jurídica no banco.
        PessoaJ.setLogradouro("Avenida Abelardo, Vila Rosali nº 1997");
        PJuridicaDao.alterar(PessoaJ);

        //Consultar todas as pessoas jurídicas do banco e listar no console.
        ArrayList<PessoaJuridica> resultadoJ = PJuridicaDao.getPessoas();
        resultadoJ.forEach(item -> item.exibir());

        //Excluir a pessoa jurídica criada anteriormente no banco.
        PJuridicaDao.excluir(PessoaJ);
        PJuridicaDao.close();
    }
}
```

**Cadastrobd.model**

**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 cadastrobd.model;


/**
 *
 * @author Usuario
 */
public class Pessoa {
    private Integer id;
    private String nome;
    private String
    logradouro;private String
    cidade; private String
    estado; private String
    telefone; private String
    email;


    Pessoa(){


    }


    Pessoa(Integer _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;
}



void exibir(){
    System.out.println("id: " + this.getId().toString());
    System.out.println("nome: " + this.getNome());
    System.out.println("logradouro: " + this.getLogradouro());
    System.out.println("cidade: " + this.getCidade());
    System.out.println("estado: " + this.getEstado());
    System.out.println("telefone: " + this.getTelefone());
    System.out.println("email: " + this.getEmail());
}

/**
 * @return the id
 */
public Integer getId()
    {return id;
}

/**
```

```
 * @param id the id to set
 */
public void setId(Integer id)
    {this.id = id;
}

/**
 * @return the nome
 */
public String getNome()
    {return nome;
}

/**
 * @param nome the nome to set
 */
public void setNome(String nome)
    {this.nome = nome;
}

/**
 * @return the logradouro
 */
public String getLogradouro()
    {return logradouro;
}

/**
 * @param logradouro the logradouro to set
 */
```

```
public void setLogradouro(String logradouro)
    {this.logradouro = logradouro;
}
/**
 * @return the cidade
 */
public String getCidade()
    {return cidade;
}

/**
 * @param cidade the cidade to set
 */
public void setCidade(String cidade) {
    this.cidade = cidade;
}

/**
 * @return the estado
 */
public String getEstado()
    {return estado;
}

/**
 * @param estado the estado to set
 */
public void setEstado(String estado) {
    this.estado = estado;
}
```

```
    /**
     * @return the telefone
     */
    public String getTelefone()
        {return telefone;
    }

    /**
     * @param telefone the telefone to set
     */
    public void setTelefone(String telefone)
        {this.telefone = telefone;
    }

    /**
     * @return the email
     */
    public String getEmail()
        {return email;
    }

    /**
     * @param email the email to set
     */
    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 cadastrobd.model;

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

    public PessoaFisica() {
        super();
    }

    public PessoaFisica(String cpf, Integer _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.getCpf().toString());
    }

    /**
     * @return the cpf
     */
    public String getCpf()
        {return cpf;
    }

    /**
     * @param cpf the cpf to set
     */
    public void setCpf(String cpf)
        {this.cpf = cpf;
    }

}
```

**PessoaFisicaDAO.java**

```
package cadastrobd.model;

import java.sql.ResultSet;
import
java.sql.SQLException;import
java.util.ArrayList;

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

```
/*
 * 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
 */
/**
 *
 * @author Usuario
 */
public class PessoaFisicaDAO {

    ConectorBD cnx = new ConectorBD();
    SequenceManager sequenceCode = new SequenceManager();

    public PessoaFisica getPessoa(Integer id) throws SQLException {
        ResultSet rs = cnx.getSelect("select \n"
                + //
                "\tpf.idPessoa_Fisica as id,\n"
                + //
                "\tpf.cpf,\n
                "
                + //
                "\tp.nome,\n
                "
                + //
                "\tp.logradouro,\n
                "
                + //
                "\tp.cidade,\n
```

```
        "
        + //
        "\tp.estado,\n"
        + //
        "\tp.telefone,\n
        "
        + //
        "\tp.email\n
        "
        + //
        "from Pessoa_Fisica pf\n"
        + //
        "inner join pessoa p on pf.idPessoa  = p.id_Pessoa \n"
        + //
        "\n
        "
        + //
        "where pf.idPessoa_Fisica  = " + id.toString());

rs.next();
PessoaFisica p = new PessoaFisica(
        rs.getString("cpf"),
        rs.getInt("id"),
        rs.getString("nome"),
        rs.getString("logradouro"),
        rs.getString("cidade"),
        rs.getString("estado"),
        rs.getString("telefone"),
        rs.getString("email")
```

```
    );
    p.exibir();
    cnx.close();
    return p;
}

public ArrayList<PessoaFisica> getPessoas() throws SQLException {
    ArrayList<PessoaFisica> list = new ArrayList<PessoaFisica>();

    ResultSet rs = cnx.getSelect("select \n"
            + //
            "\tpf.idPessoa_Fisica as id,\n"
            + //
            "\tpf.cpf,\n
            "
            + //
            "\tp.nome,\n
            "
            + //
            "\tp.logradouro,\n
            "
            + //
            "\tp.cidade,\n
            "
            + //
            "\tp.estado,\n"
            + //
            "\tp.telefone,\n
            "
```

```
            + //
            "\tp.email\n
            "
            + //
            "from Pessoa_Fisica pf \n"
            + //
            "inner join pessoa p on pf.idPessoa  = p.id_Pessoa ");

        while (rs.next()) {
            PessoaFisica p = new PessoaFisica(
                    rs.getString("cpf"),
                    rs.getInt("id"),
                    rs.getString("nome"),
                    rs.getString("logradouro"),
                    rs.getString("cidade"),
                    rs.getString("estado"),
                    rs.getString("telefone"),
                    rs.getString("email")
            );
            list.add(p);
        }
        cnx.close();
        return list;
    }

    public void incluir(PessoaFisica p) throws SQLException {
        String sqlInsertPessoa = String.format(
                "insert into pessoa ( nome, logradouro, cidade, estado, telefone, email ) values (  '%s', '%s', '%s', '%s', '%s',  '%s' );",
                p.getNome(), p.getLogradouro(), p.getCidade(), p.getEstado(), p.getTelefone(),
```

```
p.getEmail());
        //System.out.println(sql);

        int idNovaPessoa = cnx.insert(sqlInsertPessoa);
        if (idNovaPessoa == 0) {
            System.out.println("Erro ao criar pessoa");
        } else {

            String sqlInsertPessoaFisica = String.format("insert into Pessoa_Fisica (
idPessoa_Fisica, cpf, idPessoa ) values (%s, '%s', %s);", p.getId(),
                    p.getCpf(), idNovaPessoa);
            //System.out.println(sqlInsertPessoaFisica);
            cnx.insert(sqlInsertPessoaFisica);
        }
    }

    public void alterar(PessoaFisica novaPessoa) throws SQLException {

        String sqlUpdatePessoaFisica = String.format(
                "update Pessoa_Fisica  set cpf= '%s' where Pessoa_fisica.idPessoa_Fisica  = %s;",
                novaPessoa.getCpf(), novaPessoa.getId()
        );
        System.out.println(sqlUpdatePessoaFisica);
        cnx.update(sqlUpdatePessoaFisica);

        ResultSet rs = cnx.getSelect("select idPessoa from Pessoa_Fisica pf  WHERE
pf.idPessoa_Fisica = " + novaPessoa.getId() + ";");
        rs.next();

        int idPessoaAssociadaA_PessoaFisica = rs.getInt(1);
```

```
        String sqlUpdatePessoa = String.format(

                "update pessoa set nome = '%s', logradouro = '%s', cidade = '%s', estado='%s',
telefone = '%s',  email = '%s'  WHERE id_Pessoa = %s;",

                novaPessoa.getNome(), novaPessoa.getLogradouro(), novaPessoa.getCidade(),
novaPessoa.getEstado(), novaPessoa.getTelefone(), novaPessoa.getEmail(),
idPessoaAssociadaA_PessoaFisica

        );


        cnx.update(sqlUpdatePessoa);


    }


    public void excluir(Pessoa p) throws SQLException {

        ResultSet rs = cnx.getSelect("select idPessoa from Pessoa_Fisica pf  WHERE
pf.idPessoa_Fisica = " + p.getId() + ";");

        rs.next();

        int idPessoaAssociadaA_PessoaFisica = rs.getInt(1);

        String sqlDeletePessoaFisica = "Delete from Pessoa_Fisica where idPessoa_Fisica = " +
p.getId() + ";";

        cnx.update(sqlDeletePessoaFisica);
        String sqlDeletePessoa = "Delete from Pessoa where id_Pessoa = " +
idPessoaAssociadaA_PessoaFisica + ";";

        cnx.update(sqlDeletePessoa);

    }


    public void close() throws SQLException

        {cnx.close();

    }

}
```

**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 cadastrobd.model;

/**
 *
 * @author Usuario
 */
public class PessoaJuridica extends Pessoa {

    private String cnpj;

    public PessoaJuridica() {
        super();
    }
    public PessoaJuridica(String cnpj, Integer _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.getCnpj().toString());
    }

    /**
     * @return the cnpj
```

```
     */
    public String getCnpj()
        {return cnpj;
    }

    /**
     * @param cnpj the cnpj to set
     */
    public void setCnpj(String cnpj)
        {this.cnpj = cnpj;
    }

}
```

**PessoaJuridicaDAO.java**

```
package cadastrobd.model;

import  cadastrobd.model.util.ConectorBD;
import cadastrobd.model.util.SequenceManager;
import java.sql.ResultSet;
import
java.sql.SQLException;import
java.util.ArrayList;


/*
 * 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
 */
/**
 *
```

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

    ConectorBD cnx = new ConectorBD();

    public PessoaJuridica getPessoa(Integer id) throws SQLException {
        ResultSet rs = cnx.getSelect("select\n"
                + "  Pessoa_Juridica.idPessoa_Juridica as id,\n"
                + "  Pessoa_Juridica.cnpj,\n"
                + "  p.nome,\n"
                + "  p.logradouro,\n"
                + "   p.cidade,\n"
                + "   p.estado,\n"
                + "  p.telefone,\n"
                + "  p.email\n"
                + "  from Pessoa_Juridica\n"
                + "INNER JOIN Pessoa as p on Pessoa_Juridica.idPessoa = p.id_Pessoa\n"
                + "WHERE\n"
                + "  Pessoa_Juridica.idPessoa = " + id.toString());

        rs.next();
        PessoaJuridica p = new PessoaJuridica(
                rs.getString("cnpj"),
                rs.getInt("id"),
                rs.getString("nome"),
                rs.getString("logradouro"),
                rs.getString("cidade"),
                rs.getString("estado"),
                rs.getString("telefone"),
```

```
            rs.getString("email")
        );
        p.exibir();
        cnx.close();
        return p;
    }

    public ArrayList<PessoaJuridica> getPessoas() throws SQLException {
        ArrayList<PessoaJuridica> list = new ArrayList<PessoaJuridica>();

        ResultSet rs = cnx.getSelect("select\n"
                + "  Pessoa_Juridica.idPessoa_Juridica as id,\n"
                + "  Pessoa_Juridica.cnpj,\n"
                + "  p.nome,\n"
                + "  p.logradouro,\n"
                + "  p.cidade,\n"
                + "  p.estado,\n"
                + "  p.telefone,\n"
                + "  p.email\n"
                + "  from Pessoa_Juridica\n"
                + "INNER JOIN Pessoa as p on Pessoa_Juridica.idPessoa = p.id_Pessoa;");

        while (rs.next()) {
            PessoaJuridica p = new PessoaJuridica(
                    rs.getString("cnpj"),
                    rs.getInt("id"),
                    rs.getString("nome"),
                    rs.getString("logradouro"),
                    rs.getString("cidade"),
                    rs.getString("estado"),
```

```
                rs.getString("telefone"),
                rs.getString("email")
            );
            list.add(p);
        }
        cnx.close();
        return list;
    }

    public void incluir(PessoaJuridica p) throws SQLException
        {String sqlInsertPessoa = String.format(
                "insert into pessoa ( nome, logradouro, cidade, estado, telefone, email ) values (  '%s', '%s', '%s', '%s', '%s',  '%s' );'',
                p.getNome(), p.getLogradouro(), p.getCidade(), p.getEstado(), p.getTelefone(), p.getEmail());
        System.out.println(sqlInsertPessoa);
        int idNovaPessoa = cnx.insert(sqlInsertPessoa);
        if (idNovaPessoa == 0) {
            System.out.println("Erro ao criar pessoa");
        } else {

            String sqlInsertPessoaJuridica = String.format("insert into Pessoa_Juridica ( idPessoa_Juridica,cnpj,          idPessoa ) values (%s, '%s', %s);'',
                    p.getId(), p.getCnpj(), idNovaPessoa);
            System.out.println(sqlInsertPessoaJuridica);
            cnx.insert(sqlInsertPessoaJuridica);
        }

    }

    public void alterar(PessoaJuridica novaPessoa) throws SQLException {
```

```
        String sqlUpdatePessoaJuridica = String.format(
                "update Pessoa_Juridica  set cnpj = '%s' where Pessoa_Juridica.idPessoa_Juridica  =
%s;",
                                                                                                );
novaPessoa.getCnpj(), novaPessoa.getId()
        System.out.println(sqlUpdatePessoaJuridica);
        cnx.update(sqlUpdatePessoaJuridica);

        ResultSet rs = cnx.getSelect("select pj.idPessoa from Pessoa_Juridica pj WHERE
idPessoa_Juridica = " + novaPessoa.getId() + ";");
        rs.next();

        int idPessoaAssociadaA_PessoaJuridica = rs.getInt(1);

        String sqlUpdatePessoa = String.format(
                "update pessoa set nome = '%s', logradouro = '%s', cidade = '%s', estado='%s',
telefone = '%s',  email = '%s'  WHERE id_Pessoa = %s;",
                novaPessoa.getNome(), novaPessoa.getLogradouro(), novaPessoa.getCidade(),
novaPessoa.getEstado(), novaPessoa.getTelefone(),
                novaPessoa.getEmail(), idPessoaAssociadaA_PessoaJuridica
        );
        System.out.println(sqlUpdatePessoa);

        cnx.update(sqlUpdatePessoa);

    }

    public void excluir(Pessoa p) throws SQLException {
        ResultSet rs = cnx.getSelect("select idPessoa from Pessoa_Juridica pj  WHERE
pj.idPessoa_Juridica = " + p.getId() + ";");
```

```
        rs.next();
        int idPessoaAssociadaA_PessoaJuridica = rs.getInt(1);
        String sqlDeletePessoaJuridica = "Delete from Pessoa_Juridica where idPessoa_Juridica = " + p.getId() + ";";
        cnx.update(sqlDeletePessoaJuridica);
        String sqlDeletePessoa = "Delete from Pessoa where id_Pessoa = " + idPessoaAssociadaA_PessoaJuridica + ";";
        cnx.update(sqlDeletePessoa);
    }

    public void close() throws SQLException
        {cnx.close();
    }
}
```

**cadastrobd.model.util**

**ConectorBD.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 cadastrobd.model.util;

import java.sql.*;

/**
 *
 * @author Usuario
 */
public class ConectorBD {
```

```
    ResultSet rs;
    Connection con;
    PreparedStatement stmt;

    Connection getConnection() throws SQLException {
        con =
DriverManager.getConnection("jdbc:sqlserver://192.168.15.5:1433;databaseName=loja;encrypt
=true;trustServerCertificate=true", "sa", "yRjSb2D4G39Z");
        System.out.println("conectou com sucesso!");
        return con;
    }

    PreparedStatement getPrepared(String sql) throws SQLException {
        stmt = getConnection().prepareStatement(sql);
        return stmt;
    }

    public ResultSet getSelect(String sql) throws SQLException
        {rs = getPrepared(sql).executeQuery();
        return rs;
    }

    public int insert(String sql) throws SQLException {

        int primkey = 0;
        PreparedStatement ps = getConnection().prepareStatement(sql + "SELECT
SCOPE_IDENTITY()");
        ps.executeUpdate();
        ResultSet rs = ps.getGeneratedKeys();
        if (rs.next()) {
            primkey = rs.getInt(1);
```

```
        }


        return primkey;
    }

    public boolean update(String sql) throws SQLException {
        PreparedStatement ps = getConnection().prepareStatement(sql);
        ps.executeUpdate();
        boolean executed = ps.execute();
        return executed;
    }
    public void close() throws SQLException
        {rs.close();
        stmt.close();
        con.close();
    }
}
```

**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 cadastrobd.model.util;

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

```
/**
 *
 * @author Usuario
 */
public class SequenceManager {

    public int getValue(String nomeSequencia) throws SQLException {
        ConectorBD conectorBD = new ConectorBD(); ";");
    ResultSet rs = conectorBD.getSelect("SELECT NEXT VALUE FOR " +
        nomeSequencia +


rs.next();
int nextValue = rs.getInt(1);
        return nextValue;


    }
}
```

**Analise e Conclusão:**

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

Os componentes de middleware, como o JDBC, desempenham um papel crítico na eficiência, escalabilidade, segurança e integração de aplicativos empresariais. Eles tornam o desenvolvimento mais eficiente, garantem a interoperabilidade entre sistemas e permitem queas empresas aproveitem os recursos de dados de forma mais eficaz.

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

ao usar o JDBC, é altamente recomendável preferir PreparedStatement sempre que possível, devido aos benefícios de segurança, desempenho e usabilidade que ele oferece em relação aos Statement. A menos que haja uma necessidade específica de usar consultas dinâmicas, como em casos de geração dinâmica de SQL, o uso de PreparedStatement é a escolha maissegura e eficiente.

**Como o padrão DAO melhora a manutenibilidade do software?**

O padrão DAO ajuda a organizar o código, facilita a manutenção, promove a reutilização e melhora a flexibilidade do software, tornando-o mais resistente a mudanças futuras.

**Como a herança é refletida no banco de dados, quando lidamos com um modeloestritamente relacional?**

A escolha entre essas abordagens depende das necessidades do sistema, com herança única sendo mais simples, enquanto herança múltipla oferece maior granularidade. Ambas permitem representar hierarquias de classes em um banco de dados relacional.

1. **Importância dos componentes de middleware, como o JDBC:** Os componentes de middleware desempenham um papel crucial na conectividade e na integração de sistemas em uma arquitetura de software empresarial. O JDBC (Java Database Connectivity) é um exemplo significativo nesse contexto, pois fornece uma interface padrão para que aplicativos Java possam interagir com bancos de dados relacionais. Isso permite que os desenvolvedores criem aplicativos robustos e escaláveis que podem acessar e manipular dados de maneira eficiente, independentemente do banco de dados subjacente. Além disso, o JDBC oferece recursos importantes, como transações, controle de concorrência e manipulação de metadados, contribuindo para a eficiência e a confiabilidade das operações de banco de dados.

2. **Diferença entre o uso de Statement ou PreparedStatement:** O Statement e o PreparedStatement são duas maneiras de executar consultas SQL em um banco de dados usando JDBC. O PreparedStatement é preferível ao Statement por vários motivos. Primeiro, o PreparedStatement fornece uma maneira de pré-compilar consultas SQL, o que pode melhorar significativamente o desempenho, especialmente em aplicativos que executam consultas repetidamente. Além disso, o PreparedStatement oferece proteção contra-ataques de injeção de SQL, pois permite a passagem de parâmetros de forma segura, enquanto o Statement concatena diretamente os valores na consulta, o que pode deixar o sistema vulnerável a ataques. Portanto, em termos de segurança, desempenho e usabilidade, é altamente recomendável usar PreparedStatement sempre que possível.

3. **Como o padrão DAO melhora a manutenibilidade do software:** O padrão DAO (Data Access Object) é uma abordagem de design que separa a lógica de acesso a dados da lógica de negócios de um aplicativo. Isso significa que as operações de acesso a dados são encapsuladas em classes específicas (os DAOs), fornecendo uma camada de abstração entre a lógica de negócios e o banco de dados subjacente. Isso melhora a manutenibilidade do software de várias maneiras. Primeiro, facilita a identificação e a modificação de código relacionado ao acesso a dados, pois essas operações são centralizadas em classes DAO dedicadas. Além disso, promove a reutilização de código, uma vez que operações comuns de acesso a dados podem ser encapsuladas em métodos reutilizáveis dentro dos DAOs. Finalmente, o padrão DAO torna o software mais flexível, pois permite a substituição ou atualização do banco de dados subjacente sem afetar a lógica de negócios do aplicativo.

4. **Como a herança é refletida no banco de dados em um modelo estritamente relacional:** Em um modelo de banco de dados estritamente relacional, a herança é implementada de várias maneiras, sendo as mais comuns a abordagem de herança única e a de herança múltipla. Na herança única, cada classe de nível superior no modelo de objeto é mapeada para uma tabela no banco de dados, enquanto as classes filhas são mapeadas para a mesma tabela, com a adição de colunas extras para os atributos exclusivos dessas classes. Por outro lado, na herança múltipla, cada classe do modelo de objeto é mapeada para sua própria tabela no banco de dados, com relações de chave estrangeira entre as tabelas para representar as relações de herança. A escolha entre essas abordagens depende das necessidades específicas do sistema, com a herança única sendo mais simples e a herança múltipla oferecendo maior granularidade na modelagem de hierarquias de classes.