Nome: **Ariel Freitas dos Santos**
Matrícula: **202208291694**
Curso: **Desenvolvimento Full Stack**
Turma : **2023.3**
Campus: **POLO ST MORADA DO SOL - GOIÂNIA - GO**
Github: **https://github.com/ArielFSantos/SQL_Estacio_Nivel_3**

Missão Pratica | Mundo 3 | Nível 3

RPG0016 - BackEnd sem banco não tem

Objetivo:

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
6. Server na persistência de dados.

CadastroBD.java

```java
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;
import java.util.Scanner;
import cadastrobd.model.util.ConectorBD;
import cadastrobd.model.util.SequenceManager;
import java.util.List;

public class CadastroBD {
    public static void main(String[] args) {
        ConectorBD conector = new ConectorBD(url:"jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;", user:"loja", password:"loja");
        SequenceManager sequenceManager = new SequenceManager(conector);

        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conector, sequenceManager);
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(conector, sequenceManager);

        Scanner scanner = new Scanner(source: System.in);
        int opcao;

        do {
            System.out.println(x: "Selecione uma opção:");
            System.out.println(x: "1 - Incluir");
            System.out.println(x: "2 - Alterar");
            System.out.println(x: "3 - Excluir");
            System.out.println(x: "4 - Exibir pelo ID");
            System.out.println(x: "5 - Exibir todos");
            System.out.println(x: "0 - Sair");

            opcao = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character

            switch (opcao) {
                case 1 -> {
                    System.out.println(x: "Selecione o tipo (F - Pessoa Fisica, J - Pessoa Juridica):");
                    String tipoCadastro = scanner.nextLine();
                    switch (tipoCadastro) {
                        case "F"  -> {
                                // Cadastro de Pessoa Fisica
                                PessoaFisica pessoaFisica = new PessoaFisica();

                                // Solicitar os dados do usuário via teclado
                                System.out.println(x: "Nome: ");
                                pessoaFisica.setNome(nome: scanner.next());
```

```java
                // Solicitar outros atributos da Pessoa Física (como CPF, logradouro, cidade, etc.)
                System.out.println(x: "CPF: ");
                pessoaFisica.setCpf(cpf:scanner.next());
                System.out.println(x: "Logradouro: ");
                pessoaFisica.setLogradouro(logradouro: scanner.next());
                System.out.println(x: "Cidade: ");
                pessoaFisica.setCidade(cidade: scanner.next());
                System.out.println(x: "Estado: ");
                pessoaFisica.setEstado(estado: scanner.next());
                System.out.println(x: "Telefone: ");
                pessoaFisica.setTelefone(telefone: scanner.next());
                System.out.println(x: "Email: ");
                pessoaFisica.setEmail(email: scanner.next());

                // Chamar o método pessoaFisicaDAO.incluir(objetoPessoaFisica)
                pessoaFisicaDAO.incluir(pessoaFisica);
            }
            case "J" -> {
                // Cadastro de Pessoa Jurídica
                PessoaJuridica pessoaJuridica = new PessoaJuridica();

                // Solicitar os dados do usuário via teclado
                System.out.println(x: "Nome: ");
                pessoaJuridica.setNome(novoNome: scanner.next());

                // Solicitar os demais dados via teclado (por exemplo, CNPJ, telefone, email, estado, cidade, logradouro)
                System.out.println(x: "CNPJ: ");
                pessoaJuridica.setCnpj(cnpj: scanner.next());

                System.out.println(x: "Telefone: ");
                pessoaJuridica.setTelefone(telefone: scanner.next());

                System.out.println(x: "Email: ");
                pessoaJuridica.setEmail(email: scanner.next());

                System.out.println(x: "Estado: ");
                pessoaJuridica.setEstado(estado: scanner.next());

                System.out.println(x: "Cidade: ");
                pessoaJuridica.setCidade(cidade: scanner.next());

                System.out.println(x: "Logradouro: ");
                pessoaJuridica.setLogradouro(logradouro: scanner.next());
```

```
            // Chamar o método pessoaJuridicaDAO.incluir(objetoPessoaJuridica)
            pessoaJuridicaDAO.incluir(pessoaJuridica);
        }
        default -> System.out.println(x: "Opção invalida.");
    }
}
case 2 -> {
    System.out.println(x: "Selecione o tipo (F - Pessoa Fisica, J - Pessoa Juridica):");
    String tipoAlteracao = scanner.nextLine();
    switch (tipoAlteracao) {
        case "F" -> {
            // Implementar opção para alterar Pessoa Física
            System.out.println(x: "ID da Pessoa Fisica a ser alterada: ");
            int id = scanner.nextInt();
            PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);

            if (pessoaFisica != null) {
                // Apresentar os dados atuais
                System.out.println(x: "Dados atuais:");
                System.out.println(x: pessoaFisica);

                // Solicitar os novos dados
                System.out.println(x: "Novo nome: ");
                pessoaFisica.setNome(nome: scanner.next());

                System.out.println(x: "Novo CPF: ");
                pessoaFisica.setCpf(cpf:scanner.next());

                System.out.println(x: "Novo Logradouro: ");
                pessoaFisica.setLogradouro(logradouro: scanner.next());

                System.out.println(x: "Nova Cidade: ");
                pessoaFisica.setCidade(cidade: scanner.next());

                System.out.println(x: "Novo Estado: ");
                pessoaFisica.setEstado(estado: scanner.next());

                System.out.println(x: "Novo Telefone: ");
                pessoaFisica.setTelefone(telefone: scanner.next());

                System.out.println(x: "Novo Email: ");
                pessoaFisica.setEmail(email: scanner.next());

                // Chamar o método pessoaFisicaDAO.alterar(pessoaFisica)
                pessoaFisicaDAO.alterar(pessoaFisica);
                System.out.println(x: "Pessoa Fisica alterada com sucesso.");
```

```
        } else {
            System.out.println(x: "Pessoa Fisica não encontrada.");
        }
    }


    case "J" -> {
        // Implementar opção para alterar Pessoa Jurídica
        System.out.println(x: "ID da Pessoa Juridica a ser alterada: ");
        int id = scanner.nextInt();
        PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);

        if (pessoaJuridica != null) {
            // Apresentar os dados atuais
            System.out.println(x: "Dados atuais:");
            System.out.println(x: pessoaJuridica);

            // Solicitar os novos dados
            System.out.println(x: "Novo nome: ");
            pessoaJuridica.setNome(novoNome: scanner.next());

            System.out.println(x: "Novo CNPJ: ");
            pessoaJuridica.setCnpj(cnpj: scanner.next());

            System.out.println(x: "Novo Logradouro: ");
            pessoaJuridica.setLogradouro(logradouro: scanner.next());

            System.out.println(x: "Nova Cidade: ");
            pessoaJuridica.setCidade(cidade: scanner.next());

            System.out.println(x: "Novo Estado: ");
            pessoaJuridica.setEstado(estado: scanner.next());

            System.out.println(x: "Novo Telefone: ");
            pessoaJuridica.setTelefone(telefone: scanner.next());

            System.out.println(x: "Novo Email: ");
            pessoaJuridica.setEmail(email: scanner.next());

            // Chamar o método pessoaJuridicaDAO.alterar(pessoaJuridica)
            pessoaJuridicaDAO.alterar(pessoaJuridica);
            System.out.println(x: "Pessoa Juridica alterada com sucesso.");
        } else {
            System.out.println(x: "Pessoa Juridica não encontrada.");
        }
    }
```

```
                    default -> System.out.println(x: "Opção invalida.");
                }
            }
            case 3 -> {
System.out.println(x: "Selecione o tipo (F - Pessoa Fisica, J - Pessoa Juridica):");
String tipoExclusao;
                tipoExclusao = scanner.nextLine();

switch (tipoExclusao) {
    case "F" -> {
        System.out.println(x: "ID da Pessoa Fisica a ser excluida: ");
        int id = scanner.nextInt();
        PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);

        if (pessoaFisica != null) {
            System.out.println(x: "Dados da Pessoa Fisica a ser excluida:");
            System.out.println(x: pessoaFisica);

            System.out.println(x: "Tem certeza de que deseja excluir esta Pessoa Fisica? (S/N)");
            String confirmacao = scanner.next();

            if (confirmacao.equalsIgnoreCase(anotherString: "S")) {
                pessoaFisicaDAO.excluir(id);
                System.out.println(x: "Pessoa Fisica excluida com sucesso.");
            } else {
                System.out.println(x: "Operação de exclusão cancelada.");
            }
        } else {
            System.out.println(x: "Pessoa Fisica não encontrada.");
        }
    }

    case "J" -> {
        System.out.println(x: "ID da Pessoa Juridica a ser excluida: ");
        int id = scanner.nextInt();
        PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);

        if (pessoaJuridica != null) {
            System.out.println(x: "Dados da Pessoa Juridica a ser excluida:");
            System.out.println(x: pessoaJuridica);

            System.out.println(x: "Tem certeza de que deseja excluir esta Pessoa Juridica? (S/N)");
            String confirmacao = scanner.next();
```

```
            if (confirmacao.equalsIgnoreCase(anotherString: "S")) {
                pessoaJuridicaDAO.excluir(id);
                System.out.println(x: "Pessoa Juridica excluida com sucesso.");
            } else {
                System.out.println(x: "Operação de exclusão cancelada.");
            }
        } else {
            System.out.println(x: "Pessoa Juridica não encontrada.");
        }
    }

    default -> System.out.println(x: "Opção invalida.");
}


            case 4 -> {
System.out.println(x: "Selecione o tipo (F - Pessoa Fisica, J - Pessoa Jurídica):");
String tipoExibirPorID = scanner.nextLine();

switch (tipoExibirPorID) {
    case "F" -> {
        System.out.println(x: "ID da Pessoa Fisica a ser exibida: ");
        int id = scanner.nextInt();
        PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);

        if (pessoaFisica != null) {
            System.out.println(x: "Dados da Pessoa Fisica:");
            System.out.println(x: pessoaFisica);
        } else {
            System.out.println(x: "Pessoa Fisica não encontrada.");
        }
    }

    case "J" -> {
        System.out.println(x: "ID da Pessoa Juridica a ser exibida: ");
        int id = scanner.nextInt();
        PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);

        if (pessoaJuridica != null) {
            System.out.println(x: "Dados da Pessoa Juridica:");
            System.out.println(x: pessoaJuridica);
        } else {
            System.out.println(x: "Pessoa Juridica não encontrada.");
        }
    }
```

```
        default -> System.out.println("Opção invalida.");
    }
}
                case 5 -> {
                    System.out.println("Selecione o tipo (F - Pessoa Fisica, J - Pessoa Juridica):");
                    String tipoExibirTodos = scanner.nextLine();

                    switch (tipoExibirTodos) {
                        case "F" ->  {
                            System.out.println("Pessoas Fisicas no Banco:");
                            List<PessoaFisica> pessoasFisicas = (List<PessoaFisica>) pessoaFisicaDAO.getPessoas();
                            for (PessoaFisica pf : pessoasFisicas) {
                                System.out.println(pf);
                            }
                        }

                        case "J" -> {
                            System.out.println("Pessoas Juridicas no Banco:");
                            pessoaJuridicaDAO.getPessoas().forEach(System.out::println);
                        }

                        default -> System.out.println("Opção invalida.");
                    }
                }
                case 0 -> System.out.println("Saindo...");
                default -> System.out.println("Opção invalida.");
            }

                } while (opcao != 0);
    }
}
```

Pessoa.java

```java
package cadastrobd.model;
public class Pessoa {
    int id;
    String nome;
    String logradouro;
    String cidade;
    String estado;
    String telefone;
    String email;

    public Pessoa() {
        // Construtor padrão
    }

    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: " + 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 String getNome() {
        return nome;
    }

   public String getLogradouro() {
     return logradouro;
    }

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

```
    }

    public String getCidade() {
        return cidade;
    }

    public String getEstado() {
        return estado;
    }

    public String getTelefone() {
        return telefone;
    }

    public String getEmail() {
        return email;
    }

    public int getId() {
        return id;
    }

    public void setNome(String novoNome) {
        nome = novoNome;
    }



}
```

PessoaFisica.java

```java
package cadastrobd.model;

public class PessoaFisica extends Pessoa {
    private String cpf;

    public PessoaFisica() {
        // Construtor padrão
    }

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

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

    public String getCpf() {
        return cpf;
    }

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

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

    @Override
    public int getId() {
        return id;
    }


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

```java
    @Override
    public String getLogradouro() {
        return logradouro;
    }


    @Override
    public String getCidade() {
        return cidade;
    }

    @Override
    public String getEstado() {
        return estado;
    }

    @Override
    public String getEmail() {
        return email;
    }

    @Override
    public String getTelefone() {
        return telefone;
    }

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


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

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

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

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

PessoaFisicaDAO.java

```java
package cadastrobd.model;

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

public class PessoaFisicaDAO {

    private final ConectorBD conector;
    private final SequenceManager sequenceManager;

    public PessoaFisicaDAO(ConectorBD conector, SequenceManager sequenceManager) {
        this.conector = conector;
        this.sequenceManager = sequenceManager;
    }

   public List<PessoaFisica> getPessoas() {
    List<PessoaFisica> pessoas = new ArrayList<>();
    String sql = "SELECT * FROM Pessoa p INNER JOIN PessoaFisica pf ON p.id = pf.id";

    try {
        ResultSet resultSet = conector.getSelect(sql);

        while (resultSet.next()) {
            PessoaFisica pessoa = new PessoaFisica(
                    id: resultSet.getInt(string: "id"),
                    nome: resultSet.getString(string: "nome"),
                    logradouro: resultSet.getString(string: "logradouro"),
                    cidade: resultSet.getString(string: "cidade"),
                    estado: resultSet.getString(string: "estado"),
                    telefone: resultSet.getString(string: "telefone"),
                    email: resultSet.getString(string: "email"),
                    cpf:resultSet.getString(string: "cpf")
            );
            pessoas.add(e: pessoa);
        }

        conector.close(resultSet);
    } catch (SQLException e) {

        }

        return pessoas;
    }
```

```java
public void incluir(PessoaFisica pessoaFisica) {
    String sqlPessoa = "INSERT INTO Pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)";
    String sqlPessoaFisica = "INSERT INTO PessoaFisica (id, cpf) VALUES (?, ?)";

    int novoId = sequenceManager.getValue(sequenceName: "SequenciaPessoa");

    try {
        conector.getConnection().setAutoCommit(bln:false);

        PreparedStatement preparedStatementPessoa = conector.getPrepared(sql:sqlPessoa);
        preparedStatementPessoa.setString(i: 1, string: pessoaFisica.getNome());
        preparedStatementPessoa.setString(i: 2, string: pessoaFisica.getLogradouro());
        preparedStatementPessoa.setString(i: 3, string: pessoaFisica.getCidade());
        preparedStatementPessoa.setString(i: 4, string: pessoaFisica.getEstado());
        preparedStatementPessoa.setString(i: 5, string: pessoaFisica.getTelefone());
        preparedStatementPessoa.setString(i: 6, string: pessoaFisica.getEmail());
        preparedStatementPessoa.execute();

        PreparedStatement preparedStatementPessoaFisica = conector.getPrepared(sql:sqlPessoaFisica);
        preparedStatementPessoaFisica.setInt(i: 1, il: novoId);
        preparedStatementPessoaFisica.setString(i: 2, string: pessoaFisica.getCpf());
        preparedStatementPessoaFisica.execute();

        conector.getConnection().commit();
        conector.getConnection().setAutoCommit(bln:true);

        conector.close(preparedStatement: preparedStatementPessoa);
        conector.close(preparedStatement: preparedStatementPessoaFisica);
    } catch (SQLException e) {
        try {
            conector.getConnection().rollback();
            conector.getConnection().setAutoCommit(bln:true);
        } catch (SQLException e2) {
            // Lide com a exceção de alguma forma apropriada para a sua aplicação

        }
    }
}
```

```java
public PessoaFisica getPessoa(int id) {
    PessoaFisica pessoa = null;
    String sql = "SELECT * FROM Pessoa p INNER JOIN PessoaFisica pf ON p.id = pf.id WHERE p.id = ?";

    try {
        PreparedStatement preparedStatement = conector.getPrepared(sql);
        preparedStatement.setInt(i: 1, il: id);
        ResultSet resultSet = preparedStatement.executeQuery();

        if (resultSet.next()) {
            pessoa = new PessoaFisica(
                    id: resultSet.getInt(string: "id"),
                    nome:resultSet.getString(string: "nome"),
                    logradouro: resultSet.getString(string: "logradouro"),
                    cidade: resultSet.getString(string: "cidade"),
                    estado: resultSet.getString(string: "estado"),
                    telefone:resultSet.getString(string: "telefone"),
                    email: resultSet.getString(string: "email"),
                    cpf:resultSet.getString(string: "cpf")
            );
        }

        conector.close(resultSet);
        conector.close(preparedStatement);
    } catch (SQLException e) {
        // Lide com a exceção de alguma forma apropriada para a sua aplicação

    }

    return pessoa;
}

public void alterar(PessoaFisica pessoaFisica) {
    String sqlPessoa = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE id = ?";
    String sqlPessoaFisica = "UPDATE PessoaFisica SET cpf = ? WHERE id = ?";

    try {
        conector.getConnection().setAutoCommit(bln:false);

        PreparedStatement preparedStatementPessoa = conector.getPrepared(sql:sqlPessoa);
        preparedStatementPessoa.setString(i: 1, string: pessoaFisica.getNome());
        preparedStatementPessoa.setString(i: 2, string: pessoaFisica.getLogradouro());
        preparedStatementPessoa.setString(i: 3, string: pessoaFisica.getCidade());
        preparedStatementPessoa.setString(i: 4, string: pessoaFisica.getEstado());
        preparedStatementPessoa.setString(i: 5, string: pessoaFisica.getTelefone());
        preparedStatementPessoa.setString(i: 6, string: pessoaFisica.getEmail());
        preparedStatementPessoa.setInt(i: 7, il: pessoaFisica.getId());
        preparedStatementPessoa.execute();
```

```
        PreparedStatement preparedStatementPessoaFisica = conector.getPrepared(sql:sqlPessoaFisica);
        preparedStatementPessoaFisica.setString(i: 1, string: pessoaFisica.getCpf());
        preparedStatementPessoaFisica.setInt(i: 2, il: pessoaFisica.getId());
        preparedStatementPessoaFisica.execute();

        conector.getConnection().commit();
        conector.getConnection().setAutoCommit(bln:true);

        conector.close(preparedStatement: preparedStatementPessoa);
        conector.close(preparedStatement: preparedStatementPessoaFisica);
    } catch (SQLException e) {
        try {
            conector.getConnection().rollback();
            conector.getConnection().setAutoCommit(bln:true);
        } catch (SQLException e2) {
            e2.printStackTrace();
            // Lide com a exceção de alguma forma apropriada para a sua aplicação
        }
    }
}

public void excluir(int id) {
    String sqlPessoaFisica = "DELETE FROM PessoaFisica WHERE id = ?";
    String sqlPessoa = "DELETE FROM Pessoa WHERE id = ?";

    try {
        conector.getConnection().setAutoCommit(bln:false);

        PreparedStatement preparedStatementPessoaFisica = conector.getPrepared(sql:sqlPessoaFisica);
        preparedStatementPessoaFisica.setInt(i: 1, il: id);
        preparedStatementPessoaFisica.execute();

        PreparedStatement preparedStatementPessoa = conector.getPrepared(sql:sqlPessoa);
        preparedStatementPessoa.setInt(i: 1, il: id);
        preparedStatementPessoa.execute();

        conector.getConnection().commit();
        conector.getConnection().setAutoCommit(bln:true);

        conector.close(preparedStatement: preparedStatementPessoaFisica);
        conector.close(preparedStatement: preparedStatementPessoa);
    } catch (SQLException e) {
        try {
            conector.getConnection().rollback();
            conector.getConnection().setAutoCommit(bln:true);
        } catch (SQLException e2) {
            // Lide com a exceção de alguma forma apropriada para a sua aplicação
```

```
            PreparedStatement preparedStatementPessoaFisica = conector.getPrepared(sql:sqlPessoaFisica);
            preparedStatementPessoaFisica.setInt(i: 1, il: id);
            preparedStatementPessoaFisica.execute();

            PreparedStatement preparedStatementPessoa = conector.getPrepared(sql:sqlPessoa);
            preparedStatementPessoa.setInt(i: 1, il: id);
            preparedStatementPessoa.execute();

            conector.getConnection().commit();
            conector.getConnection().setAutoCommit(bln:true);

            conector.close(preparedStatement: preparedStatementPessoaFisica);
            conector.close(preparedStatement: preparedStatementPessoa);
        } catch (SQLException e) {
            try {
                conector.getConnection().rollback();
                conector.getConnection().setAutoCommit(bln:true);
            } catch (SQLException e2) {
                // Lide com a exceção de alguma forma apropriada para a sua aplicação

            }
        }
    }
}
```

PessoaJuridica.java

```java
package cadastrobd.model;

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

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

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

    /**
     *
     * @return
     */
    @Override
    public String getNome() {
        return super.getNome();
}

    /**
     *
     * @return
     */
    @Override
    public String getLogradouro() {
        return super.getLogradouro();
}


    @Override
    public String getCidade() {
        return super.getCidade();
}
```

```java
    @Override
    public String getEstado() {
        return super.getEstado();
}


    @Override
   public String getTelefone() {
        return super.getTelefone();
}

    @Override
    public String getEmail() {
        return super.getEmail();
    }

    public String getCnpj() {
        return cnpj;
    }

    @Override
    public int getId() {
        return super.getId();
    }

    /**
     *
     * @param novoNome
     */
    @Override
    public void setNome(String novoNome) {
        super.setNome(novoNome);
    }

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

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

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

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

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

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


}
```

PerssoaJuridicaDAO.java

```java
package cadastrobd.model;


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

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

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

    public PessoaJuridica getPessoa(int id) {
        PessoaJuridica pessoa = null;
        String sql = "SELECT * FROM Pessoa p INNER JOIN PessoaJuridica pj ON p.id = pj.id WHERE p.id = ?";

        try {
            PreparedStatement preparedStatement = conector.getPrepared(sql);
            preparedStatement.setInt(i: 1, il: id);
            ResultSet resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                pessoa = new PessoaJuridica(
                    id: resultSet.getInt(string: "id"),
                    nome: resultSet.getString(string: "nome"),
                    logradouro: resultSet.getString(string: "logradouro"),
                    cidade: resultSet.getString(string: "cidade"),
                    estado: resultSet.getString(string: "estado"),
                    telefone: resultSet.getString(string: "telefone"),
                    email: resultSet.getString(string: "email"),
                    cnpj: resultSet.getString(string: "cnpj")
                );
            }

            conector.close(resultSet);
            conector.close(preparedStatement);
        } catch (SQLException e) {
        }
```

```
        return pessoa;
    }

    public List<PessoaJuridica> getPessoas() {
        List<PessoaJuridica> pessoas = new ArrayList<>();
        String sql = "SELECT * FROM Pessoa p INNER JOIN PessoaJuridica pj ON p.id = pj.id";

        try {
            ResultSet resultSet = conector.getSelect(sql);

            while (resultSet.next()) {
                PessoaJuridica pessoa = new PessoaJuridica(
                    id: resultSet.getInt(string: "id"),
                    nome: resultSet.getString(string: "nome"),
                    logradouro: resultSet.getString(string: "logradouro"),
                    cidade: resultSet.getString(string: "cidade"),
                    estado: resultSet.getString(string: "estado"),
                    telefone: resultSet.getString(string: "telefone"),
                    email: resultSet.getString(string: "email"),
                    cnpj: resultSet.getString(string: "cnpj")
                );
                pessoas.add(e: pessoa);
            }

            conector.close(resultSet);
        } catch (SQLException e) {
        }

        return pessoas;
    }

    public void incluir(PessoaJuridica pessoaJuridica) {
        String sqlPessoa = "INSERT INTO Pessoa (nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)";
        String sqlPessoaJuridica = "INSERT INTO PessoaJuridica (id, cnpj) VALUES (?, ?)";

        int novoId = sequenceManager.getValue(sequenceName: "nome_da_sua_sequencia"); // Substitua pelo nome correto da sequência

        try {
            conector.getConnection().setAutoCommit(bln:false);

            PreparedStatement preparedStatementPessoa = conector.getPrepared(sql:sqlPessoa);
            preparedStatementPessoa.setString(i: 1, string: pessoaJuridica.getNome());
            preparedStatementPessoa.setString(i: 2, string: pessoaJuridica.getLogradouro());
            preparedStatementPessoa.setString(i: 3, string: pessoaJuridica.getCidade());
            preparedStatementPessoa.setString(i: 4, string: pessoaJuridica.getEstado());
            preparedStatementPessoa.setString(i: 5, string: pessoaJuridica.getTelefone());
            preparedStatementPessoa.setString(i: 6, string: pessoaJuridica.getEmail());
            preparedStatementPessoa.execute();
```

```
            conector.getConnection().commit();
            conector.getConnection().setAutoCommit(bln:true);

            conector.close(preparedStatement: preparedStatementPessoa);
            conector.close(preparedStatement: preparedStatementPessoaJuridica);
        } catch (SQLException e) {
            try {
                conector.getConnection().rollback();
                conector.getConnection().setAutoCommit(bln:true);
            } catch (SQLException e2) {
            }
        }
    }

    public void alterar(PessoaJuridica pessoaJuridica) {
        String sqlPessoa = "UPDATE Pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE id = ?";
        String sqlPessoaJuridica = "UPDATE PessoaJuridica SET cnpj = ? WHERE id = ?";

        try {
            conector.getConnection().setAutoCommit(bln:false);

            PreparedStatement preparedStatementPessoa = conector.getPrepared(sql:sqlPessoa);
            preparedStatementPessoa.setString(i: 1, string: pessoaJuridica.getNome());
            preparedStatementPessoa.setString(i: 2, string: pessoaJuridica.getLogradouro());
            preparedStatementPessoa.setString(i: 3, string: pessoaJuridica.getCidade());
            preparedStatementPessoa.setString(i: 4, string: pessoaJuridica.getEstado());
            preparedStatementPessoa.setString(i: 5, string: pessoaJuridica.getTelefone());
            preparedStatementPessoa.setString(i: 6, string: pessoaJuridica.getEmail());
            preparedStatementPessoa.setInt(i: 7, i1: pessoaJuridica.getId());
            preparedStatementPessoa.execute();

            PreparedStatement preparedStatementPessoaJuridica = conector.getPrepared(sql:sqlPessoaJuridica);
            preparedStatementPessoaJuridica.setString(i: 1, string: pessoaJuridica.getCnpj());
            preparedStatementPessoaJuridica.setInt(i: 2, i1: pessoaJuridica.getId());
            preparedStatementPessoaJuridica.execute();

            conector.getConnection().commit();
            conector.getConnection().setAutoCommit(bln:true);

            conector.close(preparedStatement: preparedStatementPessoa);
            conector.close(preparedStatement: preparedStatementPessoa);
            conector.close(preparedStatement: preparedStatementPessoaJuridica);
        } catch (SQLException e) {
            try {
                conector.getConnection().rollback();
                conector.getConnection().setAutoCommit(bln:true);
            } catch (SQLException e2) {
            }
        }
```

```java
public void excluir(int id) {
    String sqlPessoaJuridica = "DELETE FROM PessoaJuridica WHERE id = ?";
    String sqlPessoa = "DELETE FROM Pessoa WHERE id = ?";

    try {
        conector.getConnection().setAutoCommit(bln:false);

        PreparedStatement preparedStatementPessoaJuridica = conector.getPrepared(sql:sqlPessoaJuridica);
        preparedStatementPessoaJuridica.setInt(i: 1, i1: id);
        preparedStatementPessoaJuridica.execute();

        PreparedStatement preparedStatementPessoa = conector.getPrepared(sql:sqlPessoa);
        preparedStatementPessoa.setInt(i: 1, i1: id);
        preparedStatementPessoa.execute();

        conector.getConnection().commit();
        conector.getConnection().setAutoCommit(bln:true);

        conector.close(preparedStatement: preparedStatementPessoaJuridica);
        conector.close(preparedStatement: preparedStatementPessoa);
    } catch (SQLException e) {
        try {
            conector.getConnection().rollback();
            conector.getConnection().setAutoCommit(bln:true);
        } catch (SQLException e2) {
        }
    }
}
```

ConectorBD.java

```java
package cadastrobd.model.util;

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

public class ConectorBD {
    private Connection connection;

    public ConectorBD(String url, String user, String password) {
        try {
            connection = DriverManager.getConnection(url, user, password);
            System.out.println(x: "Conexao estabelecida com sucesso.");
        } catch (SQLException e) {
            System.err.println("Erro ao estabelecer a conexao com o banco de dados: " + e.getMessage());
        }
    }

    public Connection getConnection() {
        return connection;
    }

    public PreparedStatement getPrepared(String sql) throws SQLException {
        return connection.prepareStatement(string: sql);
    }

    public ResultSet getSelect(String sql) throws SQLException {
        return connection.createStatement().executeQuery(string: sql);
    }

    public void close(PreparedStatement preparedStatement) {
        try {
            if (preparedStatement != null) {
                preparedStatement.close();
            }
        } catch (SQLException e) {
            System.err.println("Erro ao fechar o PreparedStatement: " + e.getMessage());
        }
    }

  public void close(ResultSet resultSet) {
    try {
        if (resultSet != null) {
            resultSet.close();
        }
```

```
    } catch (SQLException e) {
        System.err.println("Erro ao fechar o ResultSet: " + e.getMessage());
    }
}




    public void close() {
        try {
            if (connection != null) {
                connection.close();
                System.out.println(x: "Conexao fechada com sucesso.");
            }
        } catch (SQLException e) {
            System.err.println("Erro ao fechar a conexao: " + e.getMessage());
        }
    }
}
```

SequenceManager.Java

```java
package cadastrobd.model.util;
import java.sql.ResultSet;
import java.sql.SQLException;

/**
 *
 * @author ariel
 */
public class SequenceManager {
    private final ConectorBD conector;

    public SequenceManager(ConectorBD conector) {
        this.conector = conector;
    }

    public int getValue(String sequenceName) {
        int nextValue = 0;
        String sql = "SELECT nextval('" + sequenceName + "')";

        try {
            ResultSet resultSet = conector.getSelect(sql);
            if (resultSet.next()) {
                nextValue = resultSet.getInt(i: 1);
            }
            conector.close(resultSet);
        } catch (SQLException e) {
        }

        return nextValue;
    }
}
```

Análise e Conclusão:

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

O JDBC é uma API do Java que permite que uma aplicação construída na linguagem consiga acessar um banco de dados configurado local ou remotamente. O middleware é um software que atua como intermediário entre diferentes sistemas, permitindo que eles se comuniquem e troquem informações. O JDBC é um exemplo de middleware de banco de dados, que permite a interação entre aplicativos e bancos de dados.

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

O Statement é usado para executar instruções SQL fixas, ou seja, instruções de texto puro. Já o PreparedStatement é usado para executar instruções SQL parametrizadas, que possibilitam que você especifique o tipo do parâmetro passado como Int, String, Float etc. Além disso, o PreparedStatement pré-executa os passos de interpretação, compilação e planejamento da consulta SQL, enquanto o Statement passa por esses passos para cada consulta SQL enviada para o banco

Como o padrão DAO melhora a manutenibilidade do software?

O padrão DAO (Data Access Object) é um padrão de projeto que separa a lógica de negócios da lógica de acesso a dados. Ele permite que os desenvolvedores criem uma camada de abstração entre a aplicação e o banco de dados, tornando o código mais modular e fácil de manter. Com o DAO, as operações de leitura e gravação de dados são encapsuladas em uma classe separada, que pode ser facilmente substituída ou atualizada sem afetar o restante do código. Isso torna o software mais flexível e menos propenso a erros . Em resumo, o padrão DAO melhora a manutenibilidade do software, tornando-o mais modular, flexível e fácil de manter.

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

A herança em bancos de dados relacionais é um desafio comum ao modelar sistemas orientados a objetos. A melhor maneira de representar hierarquias de objetos no banco de dados relacional depende do conjunto de dados em cada objeto e da quantidade de campos comuns versus campos específicos, tamanho da hierarquia, quantidade de dados, tipos de consultas que serão efetuadas contra os dados, etc. Uma das estratégias é criar uma tabela para cada entidade, onde cada tabela conterá não só os dados da classe filha como os dados da classe pai . No entanto, essa abordagem pode não ser a melhor em todos os casos, e outras estratégias podem ser mais adequadas, dependendo do caso em questão .