

**UNIFASIPE CAMPUS AQUARELA**

**TECNÓLOGO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**TRABALHO N3**

# Programação Orientada a Objeto I

**Aluno(s): Marcelo Marques Meyer**

**Mateus Lara**

**SINOP - MT**

**JULHO / 2023**

TRABALHO POO

Oque é POO - Programação orientada a objetos é um paradigma de programação baseado no conceito de "objetos", que podem conter dados na forma de campos, também conhecidos como atributos, e códigos, na forma de procedimentos, também conhecidos como métodos.

Qual sua função - ORM é uma ferramenta que utiliza mecanismos que possibilitam a manipulação dos objetos por meio do mapeamento entre sistemas orientados a objetos e banco de dados relacionais. Ou seja, ORM é um mecanismo de mapeamento que viabiliza a relação dos objetos com os dados que eles representam.

Oque é - Mapeamento objeto-relacional é uma técnica de desenvolvimento utilizada para reduzir a impedância da programação orientada aos objetos utilizando bancos de dados relacionais.

Em Java, a Programação Orientada a Objetos é um dos pilares fundamentais da linguagem. Aqui está uma descrição dos principais conceitos de POO em Java:

Classe: Em Java, uma classe é uma estrutura que define as características e comportamentos de um objeto. Ela serve como um modelo ou molde para criar instâncias de objetos. As classes em Java são declaradas usando a palavra-chave "class".

Encapsulamento: O encapsulamento é um mecanismo que garante que os atributos e métodos internos de uma classe sejam acessados somente por meio de uma interface definida. Em Java, isso é alcançado usando modificadores de acesso, como public, private e protected. Os atributos geralmente são definidos como private e o acesso a eles é fornecido por meio de métodos getters e setters (métodos de acesso).

Nosso trabalho consistem em um cadastro de automóveis ainda na distribuição por montadora, com os seguintes dados de informação: fabricante, nome, tipo e quantidade. O sistema utiliza ORM funcional um mecanismo que manipula os objetos por mapeamento entre sistemas orientados a objeto.

Inclui um Scanner para ter questionamento e ter o mínimo de aplicabilidade

Criei um menu com do while

do {

            System.out.println("Escolha uma opção:"); // Imprimir o menu na tela

            System.out.println("1 - C\_reate ");

            System.out.println("2 - R\_ead ");

            System.out.println("3 - U\_pdate ");

            System.out.println("4 - D\_elete ");

            System.out.println("0 - S\_air");

            escolha = scan.nextInt();

            if (escolha == 1) {

No momento do **Create** menu 1 as informações digitadas são inseridas através da instancia do objeto da classe Produto

Produto produto1 = new Produto();

Demais linhas do create

                produto1.id = contador;

                System.out.println("Digite as informações dos veiculos");

                System.out.println("");

                System.out.print("Fabricante : ");

                fabricante = scan.nextLine();

                produto1.fabricante = fabricante;

                System.out.print("Nome : ");

                nome = scan.nextLine();

                produto1.nome = nome;

                System.out.print("Tipo : ");

                tipo = scan.nextLine();

                produto1.tipo = tipo;

                System.out.print("Quantidade : ");

                quantidade = scan.nextInt();

                produto1.quantidade = quantidade;

As informações digitadas são inseridas no banco através dos fragmentos de código a seguir executeSQL(String sql) e inserirRegistro

    public static boolean executeSQL(String sql) {

        boolean ok = false;

        Connection currentConnection = connect();

E

    }

    public static boolean inserirRegistro(SQLClass registro) {

        return executeSQL(registro.insertSQL());

    }

para ter certeza que os dados digitados foram inseridos criamos uma variável Boolean retorno que recebe informações true concluído ou false.

                boolean retorno = Database.inserirRegistro(produto1);

                System.out.println(retorno); // retorna true ou false

                contador = contador + 1;

No **Read** obtemos a informação do banco através do código

                boolean retorno = Database.abrirID(produto1, ler);

                System.out.println(

                        produto1.fabricante + " " + produto1.nome + " " + produto1.tipo + " " + produto1.quantidade);

                        System.out.println("");

O código responsável por buscar as informações no banco são

 public static boolean abrirID(SQLClass registro, int id) {

        Field[] fields = registro.getClass().getDeclaredFields();

        boolean ok = false;

        Connection currentConnection = connect();

        try {

            PreparedStatement stmt = currentConnection.prepareStatement(registro.selectSQL() + " where id = " + id);

            ResultSet resultSet = stmt.executeQuery();

            for (Field field : fields) {

                if (field.getType() == String.class) {

                    field.set(registro, resultSet.getString(field.getName()));

                } else {

                    field.set(registro, resultSet.getInt(field.getName()));

                }

            }

            ok = true;

            currentConnection.close();

        } catch (Exception e) {

            e.printStackTrace();

            ok = false;

        }

        return ok;

No **Update** alteramos as informações através da escolha do ID

Inserindo novos dados o banco é atualizado.

    }

    public static boolean atualizarRegistro(SQLClass registro) {

        return executeSQL(registro.updateSQL());

    }

No **Delete** também escolhemos o item do banco a ser deletado pelo ID

                System.out.println("QUAL ITEM DESEJA EXCLUIR ? :");

                excluir = scan.nextInt();

                produto1.id = excluir;

                boolean retorno = Database.deletarRegistro(produto1);

                System.out.println(retorno);