Tugas
Praktikum Pemrograman II

UNPAS
Teknik Informatika
Universitas Pasundan

JLaporan Tugas Pertemuan Ke - 10 Study Kasus

Identitas

- NPM : [223040024]
- Nama : [Diaz Alfiari Rachmad]
- Kelas : [A]
- URL Repository Github : [https://github.com/Diazalfiari/pp2_223040024_A/tree/main/Tugas/StudyKasus2/StudyKasus2]

Penjelasan Kode

Jika berkas **tidak memiliki output**, gunakan tabel ini.

mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC" />
            <dataSource type="POOLED">
                <property name="driver"
value="com.mysql.cj.jdbc.Driver" />
                <property name="url"
value="jdbc:mysql://localhost:3306/pp2_sesi11" />
                <property name="username" value="root" />
                <property name="password" value="" />
            </dataSource>
        </environment>
    </environments>
    <mappers>
                <mapper class="model.ProductMapper"/>
        <package name="mapper" />
    </mappers>
</configuration>
```

Penjelasan

[Berkas di atas adalah **konfigurasi MyBatis** yang digunakan untuk mengatur lingkungan dan penghubung (mapper) antara aplikasi dengan database.]

ProductsController.java

```
package controller;

import java.awt.GridLayout;
import model.Product;
import view.ProductsView;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import model.ProductMapper;
/**
 *
 * @author Diaza
 */
public class ProductsController {
    private ProductsView view;
    private ProductMapper mapper;



    public ProductsController(ProductsView view, ProductMapper
mapper) {
        this.view = view;
        this.mapper = mapper;

        view.getBtnAdd().addActionListener(e -> addProduct());
        view.getBtnDelete().addActionListener(e -> deleteProduct());
        view.getBtnRefresh().addActionListener(e -> loadProducts());

        view.getBtnUpdate().addActionListener(e -> {
    int selectedRow = view.getTable().getSelectedRow();
    if (selectedRow == -1) {
```

```
            return;
        }

        int id = (int) view.getTable().getValueAt(selectedRow, 0);
        String currentName = view.getTable().getValueAt(selectedRow,
1).toString();
        String currentBrand = view.getTable().getValueAt(selectedRow,
2).toString();
        double currentPrice =
Double.parseDouble(view.getTable().getValueAt(selectedRow,
3).toString());
        int currentStock =
Integer.parseInt(view.getTable().getValueAt(selectedRow,
4).toString());

        JDialog dialog = new JDialog(view, "Update Product", true);
        dialog.setLayout(new GridLayout(5, 2, 10, 10));
        dialog.setSize(400, 300);

        JTextField tfName = new JTextField(currentName);
        JTextField tfBrand = new JTextField(currentBrand);
        JTextField tfPrice = new
JTextField(String.valueOf(currentPrice));
        JTextField tfStock = new
JTextField(String.valueOf(currentStock));
        JButton btnUpdateDialog = new JButton("Update");

        dialog.add(new JLabel("Name:"));
        dialog.add(tfName);
        dialog.add(new JLabel("Brand:"));
        dialog.add(tfBrand);
        dialog.add(new JLabel("Price:"));
        dialog.add(tfPrice);
        dialog.add(new JLabel("Stock:"));
        dialog.add(tfStock);
        dialog.add(new JLabel()); // untuk jarak
        dialog.add(btnUpdateDialog);

        btnUpdateDialog.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent evt) {
```

```
            Product e = new Product();
            e.setId(id);
            e.setName(tfName.getText());
            e.setBrand(tfBrand.getText());
            e.setPrice(Double.parseDouble(tfPrice.getText()));
            e.setStock(Integer.parseInt(tfStock.getText()));

            mapper.updateProduct(e);
            JOptionPane.showMessageDialog(dialog, "Data berhasil di
update");
            dialog.dispose();
            loadProducts();
        }
    });
        dialog.setLocationRelativeTo(view);
        dialog.setVisible(true);
    });
        loadProducts();
    }

    private void loadProducts() {
        List<Product> products = mapper.getAllProducts();
        DefaultTableModel model = new DefaultTableModel(new
String[]{"ID", "Name", "Brand", "Price", "Stock"}, 0);
        for (Product e : products) {
            model.addRow(new Object[]{e.getId(), e.getName(),
e.getBrand(), e.getPrice(), e.getStock()});
        }
        view.getTable().setModel(model);
    }

    private void addProduct() {
        Product e = new Product();
        e.setName(view.getTfName().getText());
        e.setBrand(view.getTfBrand().getText());
        e.setPrice(Double.parseDouble(view.getTfPrice().getText()));
        e.setStock(Integer.parseInt(view.getTfStock().getText()));
        mapper.insertProduct(e);
        loadProducts();
    }

    private void updateProduct() {
```

```
        int selectedRow = view.getTable().getSelectedRow();
        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(view, "Pilih baris yang
ingin di update");
            return;
        }
        Product e = new Product();
    }

    private void deleteProduct() {
        int selectedRow = view.getTable().getSelectedRow();
        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(view, "Pilih baris yang
ingin di hapus");
            return;
        }
        int id = (int) view.getTable().getValueAt(selectedRow, 0);
        mapper.deleteProduct(id);
        loadProducts();
    }
}
```

Penjelasan

Codingan di atas merupakan implementasi controller dalam pola arsitektur Model-View-Controller (MVC) yang bertanggung jawab untuk mengatur logika aplikasi dan menghubungkan view (antarmuka pengguna) dengan model (data).

MyBatisUtil.java

```
package model;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.*;

import java.io.IOException;
/**
 *
 * @author Diaza
 */
```

```
public class MyBatisUtil {
    private static SqlSessionFactory sqlSessionFactory;

    static {
            try {
                sqlSessionFactory = new SqlSessionFactoryBuilder()
                    .build(Resources.getResourceAsStream
("mybatis-config.xml"));
            } catch (IOException e) {
                e.printStackTrace();
            }
    }

    public static SqlSession getSqlSession() {
        return sqlSessionFactory.openSession(true);
    }
}
```

Penjelasan

Code ini adalah implementasi utilitas MyBatis untuk mempermudah pengelolaan koneksi ke database.

Product.java

```
package model;

/**
 *
 * @author Diaza
 */
public class Product {
    private int id;
    private String name;
    private String brand;
    private double price;
    private int stock;

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

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

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getBrand() { return brand; }
    public void setBrand(String brand) { this.brand = brand; }

    public double getPrice() { return price; }
    public void setPrice(double price) { this.price = price; }

    public int getStock() { return stock; }
    public void setStock(int stock) { this.stock = stock; }
}
```

Penjelasan

Berkas ini berisi kelas model bernama Product, yang merupakan representasi objek data produk dalam aplikasi. Kelas ini digunakan untuk menyimpan data produk yang akan diolah atau disimpan dalam database.

ProductMapper.java

```
package model;

import org.apache.ibatis.annotations.*;
import java.util.List;
/**
 *
 * @author Diaza
 */
public interface ProductMapper {
    @Select("SELECT * FROM products")
    List<Product> getAllProducts();

    @Insert("INSERT INTO products (name, brand, price, stock) VALUES
(#{name}, #{brand}, #{price}, #{stock})")
    void insertProduct(Product product);

    @Update("UPDATE products SET name = #{name}, brand = #{brand},
```

```
price = #{price}, stock = #{stock} WHERE id = #{id}")
    void updateProduct(Product product);

    @Delete("DELETE FROM products WHERE id = #{id}")
    void deleteProduct(int id);
}
```

Penjelasan

Berkas ini mendefinisikan interface bernama ProductMapper, yang merupakan bagian dari MyBatis untuk memetakan operasi SQL ke metode Java. Interface ini digunakan untuk menghubungkan aplikasi dengan database tanpa perlu menulis kode SQL secara manual di tempat lain.

ProductsView.java

```
package view;

import javax.swing.*;
import java.awt.*;
/**
 *
 * @author Diaza
 */
public class ProductsView extends JFrame {
    private JTable table;
    private JTextField tfName, tfBrand, tfPrice, tfStock;
    private JButton btnAdd, btnUpdate, btnDelete, btnRefresh;

    public ProductsView() {
        setTitle("Products Data Management");
        setSize(800, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Table
        table = new JTable();
        add(new JScrollPane(table), BorderLayout.CENTER);

        // Input Panel
        JPanel inputPanel = new JPanel(new GridLayout(5, 2));
```

```
        inputPanel.add(new JLabel("Name:"));
        tfName = new JTextField();
        inputPanel.add(tfName);
        inputPanel.add(new JLabel("Brand:"));
        tfBrand = new JTextField();
        inputPanel.add(tfBrand);
        inputPanel.add(new JLabel("Price:"));
        tfPrice = new JTextField();
        inputPanel.add(tfPrice);
        inputPanel.add(new JLabel("Stock:"));
        tfStock = new JTextField();
        inputPanel.add(tfStock);
        add(inputPanel, BorderLayout.NORTH);

        // Button Panel
        JPanel buttonPanel = new JPanel();
        btnAdd = new JButton("Add");
        btnUpdate = new JButton("Update");
        btnDelete = new JButton("Delete");
        btnRefresh = new JButton("Refresh");
        buttonPanel.add(btnAdd);
        buttonPanel.add(btnUpdate);
        buttonPanel.add(btnDelete);
        buttonPanel.add(btnRefresh);
        add(buttonPanel, BorderLayout.SOUTH);
    }

    public JTable getTable() { return table; }
    public JTextField getTfName() { return tfName; }
    public JTextField getTfBrand() { return tfBrand; }
    public JTextField getTfPrice() { return tfPrice; }
    public JTextField getTfStock() { return tfStock; }
    public JButton getBtnAdd() { return btnAdd; }
    public JButton getBtnUpdate() { return btnUpdate; }
    public JButton getBtnDelete() { return btnDelete; }
    public JButton getBtnRefresh() { return btnRefresh; }
}
```

Penjelasan

Berkas ini berisi kelas ProductsView, yang bertanggung jawab untuk menampilkan antarmuka grafis pengguna (Graphical User Interface/GUI) dalam aplikasi pengelolaan

data produk. Kelas ini menggunakan pustaka Swing untuk membangun antarmuka yang memungkinkan pengguna berinteraksi dengan data produk.

Penjelasan Kode

Jika **berkas memiliki output**, gunakan tabel ini.

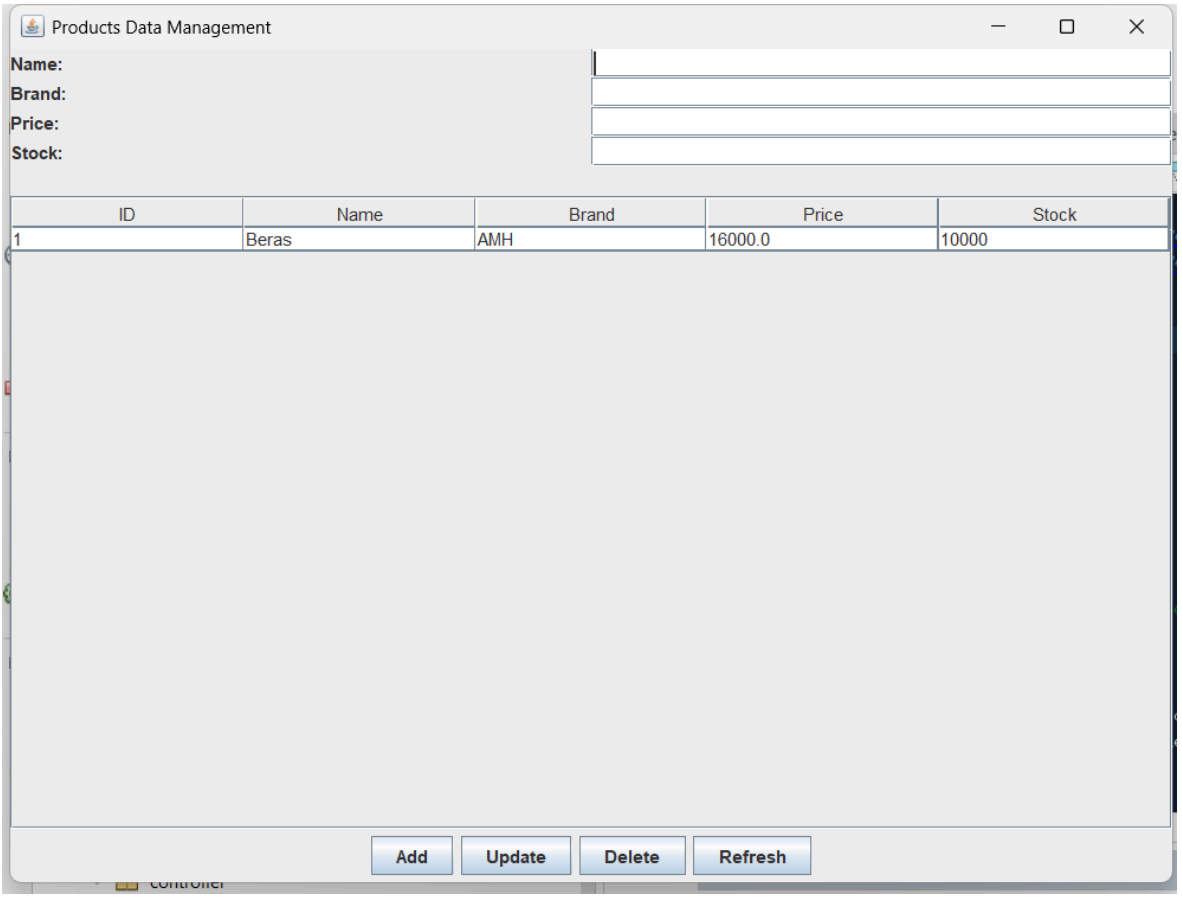
Main.java

```
package main;

import model.MyBatisUtil;
import org.apache.ibatis.session.SqlSession;
import view.ProductsView;
import controller.ProductsController;
import model.ProductMapper;
/**
 *
 * @author Diaza
 */
public class Main {
    public static void main(String[] args) {
        SqlSession session = MyBatisUtil.getSqlSession();
        ProductMapper mapper =
session.getMapper(ProductMapper.class);

        ProductsView view = new ProductsView();
        new ProductsController(view, mapper);

        view.setVisible(true);
    }
}
```

Penjelasan

Berkas ini berisi kelas Main, yang berfungsi sebagai titik awal (entry point) dari aplikasi pengelolaan data produk. Kelas ini bertanggung jawab untuk menginisialisasi komponen-komponen utama aplikasi dan menghubungkannya sehingga dapat berjalan sesuai dengan pola Model-View-Controller (MVC).

Output


Products Data Management
Name:
Brand:
Price:
Stock:
ID
Name
Brand
Price
Stock
1
Beras
AMH
16000.0
10000
Add
Update
Delete
Refresh