Tugas
Praktikum Pemrograman II

UNPAS
Teknik Informatika
Universitas Pasundan

Laporan Tugas Pertemuan ke-11

Identitas

- NPM : 223040024
- Nama : Diaz Alfiari Rachmad
- Kelas : A
- URL Repository Github : https://github.com/Diazalfiari/pp2_223040024_A/tree/main/Latihan/Latihan3_konkurensi_sesi13

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

code diatas merupakan konfigurasi untuk MyBatis,sebuah framework java untuk ORM atau pengelolaan database

User.java

```
package model;

/**
 *
 * @author Diaza
 */
public class User {
    private int id;
    private String name;
    private String email;

    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 getEmail() {
        return email;
    }

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

Penjelasan

User.java

code diatas merupakan definisi sebuah kelas Java bernama User,yang berada didalam paket model. Kelas ini digunakan untuk merepresentasikan sebuah entitas atau objek User dalam aplikasi,dengan atribut dan metode terkait.

UserMapper.java

```
package model;

import java.util.List;
import org.apache.ibatis.annotations.*;

/**
 *
 * @author Diaza
 */
public interface UserMapper {
    @Select("SELECT * FROM users")
    List<User> getAllUsers();

    @Insert("INSERT INTO users (name, email) VALUES (#{name},
#{email})")
    void insertUser(User user);
}
```

Penjelasan

code ini merupakan definisi interface java bernama userMapper, yang berada dalam paket model. Interface ini digunakan dalam kerangka MyBatis untuk memetahkan operasi database (SQL) ke metode Java. Tujuannya adalah untuk mempermudah akses manipulasi data di tabel users melalui kode Java.

MyBatisUtil.java

```
package model;

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

import java.io.IOException;
```

MyBatisUtil.java

```
/**
 *
 * @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 mendefinisikan sebuah kelas Java bernama MyBatisUtil, yang berada didalam paket model. Kelas ini berfungsi sebahagai utilitas untuk mengatur dan menyediakan koneksi ke database menggunakan MyBatis. Bertujuan untuk menyederhanakan pembuatan SqlSession yang di gunakan untuk berinteraksi dengan database

UserView.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 view;
```

UserView.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionListener;

/**
 *
 * @author Diaza
 */
public class UserView extends JFrame {
    private JTextField txtName = new JTextField(20);
    private JTextField txtEmail = new JTextField(20);
    private JButton btnAdd = new JButton("Add User");
    private JButton btnRefresh = new JButton("Refresh");
    private JButton btnExport = new JButton("Export");
    private JList<String> userList = new JList<>();
    private DefaultListModel<String> listModel = new
DefaultListModel<>();

    // Tambahan komponen baru
    private JProgressBar progressBar;
    private JLabel statusLabel;

    public UserView() {
        setTitle("User Management");
        setSize(400, 400); // Ukuran diperbesar
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Panel utama dengan GridBagLayout
        JPanel mainPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5);

        // Input panel
        JPanel inputPanel = new JPanel(new GridLayout(4, 2, 5, 5));
        inputPanel.add(new JLabel("Name:"));
        inputPanel.add(txtName);
        inputPanel.add(new JLabel("Email:"));
        inputPanel.add(txtEmail);
```

UserView.java

```
        // Button panel
        JPanel buttonPanel = new JPanel(new FlowLayout());
        buttonPanel.add(btnAdd);
        buttonPanel.add(btnRefresh);
        buttonPanel.add(btnExport);

        // Progress components
        progressBar = new JProgressBar(0, 100);
        progressBar.setStringPainted(true);
        statusLabel = new JLabel("Ready", JLabel.CENTER);

        // Layout arrangement
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 2;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        mainPanel.add(inputPanel, gbc);

        gbc.gridy = 1;
        mainPanel.add(buttonPanel, gbc);

        gbc.gridy = 2;
        mainPanel.add(statusLabel, gbc);

        gbc.gridy = 3;
        mainPanel.add(progressBar, gbc);

        gbc.gridy = 4;
        gbc.weighty = 1.0;
        gbc.fill = GridBagConstraints.BOTH;
        mainPanel.add(new JScrollPane(userList), gbc);

        userList.setModel(listModel);
        add(mainPanel);
    }

    // Getter methods yang sudah ada
    public String getNameInput() {
        return txtName.getText();
    }
```

UserView.java

```
    public String getEmailInput() {
        return txtEmail.getText();
    }

    public void setUserList(String[] users) {
        listModel.clear();
        for (String user : users) {
            listModel.addElement(user);
        }
    }

    // Menambahkan method untuk progress bar dan status
    public void setProgress(int progress) {
        progressBar.setValue(progress);
    }

    public void setStatus(String status) {
        statusLabel.setText(status);
    }

    public void enableButtons(boolean enable) {
        btnAdd.setEnabled(enable);
        btnRefresh.setEnabled(enable);
        btnExport.setEnabled(enable);
    }

    // Listener methods yang sudah ada
    public void addAddUserListener(ActionListener listener) {
        btnAdd.addActionListener(listener);
    }

    public void addRefreshListener(ActionListener listener) {
        btnRefresh.addActionListener(listener);
    }

    public void addExportListener(ActionListener listener) {
        btnExport.addActionListener(listener);
    }
}
```

UserView.java

Penjelasan

Kode ini merupakan implementasi View dalam arsitektur MVC, yang fokus pada tampilan UI dan interaksi dengan pengguna, dengan tambahan fitur konkurensi melalui progress bar dan status label untuk memberikan feedback visual kepada pengguna saat operasi sedang berlangsung.

UserController.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 controller;

import model.*;
import view.UserView;
import view.UserPdf;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import org.apache.ibatis.session.SqlSession;

/**
 *
 * @author Diaza
 */
public class UserController {
    private UserView view;
    private UserMapper mapper;
    private UserPdf pdf;

    public UserController(UserView view, UserMapper mapper, UserPdf
pdf) {
        this.view = view;
```

UserController.java

```
        this.mapper = mapper;
        this.pdf = pdf;

        this.view.addAddUserListener(new AddUserListener());
        this.view.addRefreshListener(new RefreshListener());
        this.view.addExportListener(new ExportListener());
    }

    class AddUserListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            String name = view.getNameInput();
            String email = view.getEmailInput();
            if (!name.isEmpty() && !email.isEmpty()) {
                view.enableButtons(false);
                view.setStatus("Adding user...");

                SwingWorker<Void, Void> worker = new SwingWorker<>()
{
                    @Override
                    protected Void doInBackground() throws Exception
{
                        User user = new User();
                        user.setName(name);
                        user.setEmail(email);
                        mapper.insertUser(user);
                        return null;
                    }

                    @Override
                    protected void done() {
                        view.enableButtons(true);
                        view.setStatus("User added successfully!");
                        JOptionPane.showMessageDialog(view, "User
added successfully!");
                    }
                };
                worker.execute();
            } else {
                JOptionPane.showMessageDialog(view, "Please fill in
```

UserController.java

```
all fields.");
                }
            }
        }

        class RefreshListener implements ActionListener {
            @Override
            public void actionPerformed(ActionEvent e) {
                view.enableButtons(false);
                view.setStatus("Refreshing user list...");

                SwingWorker<List<User>, Void> worker = new
SwingWorker<>() {
                    @Override
                    protected List<User> doInBackground() throws
Exception {
                        return mapper.getAllUsers();
                    }

                    @Override
                    protected void done() {
                        try {
                            List<User> users = get();
                            String[] userArray = users.stream()
                                .map(u -> u.getName() + " (" +
u.getEmail() + ")")
                                .toArray(String[]::new);
                            view.setUserList(userArray);
                            view.setStatus("User list refreshed");
                            view.enableButtons(true);
                        } catch (Exception ex) {
                            ex.printStackTrace();
                            view.setStatus("Error refreshing users");
                        }
                    }
                };
                worker.execute();
            }
        }
```

UserController.java

```
    class ExportListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            view.enableButtons(false);
            view.setStatus("Exporting to PDF...");
            view.setProgress(0);

            SwingWorker<Void, Integer> worker = new SwingWorker<>() {
                @Override
                protected Void doInBackground() throws Exception {
                    List<User> users = mapper.getAllUsers();
                    // Simulate progress
                    for (int i = 0; i <= 100; i += 10) {
                        Thread.sleep(100); // Simulate work
                        publish(i);
                    }
                    pdf.exportPdf(users);
                    return null;
                }

                @Override
                protected void process(List<Integer> chunks) {
                    int latestProgress = chunks.get(chunks.size() -
1);
                    view.setProgress(latestProgress);
                }

                @Override
                protected void done() {
                    view.setProgress(100);
                    view.setStatus("PDF exported successfully");
                    view.enableButtons(true);
                    JOptionPane.showMessageDialog(view, "PDF exported
successfully!");
                }
            };
            worker.execute();
        }
    }
}
```

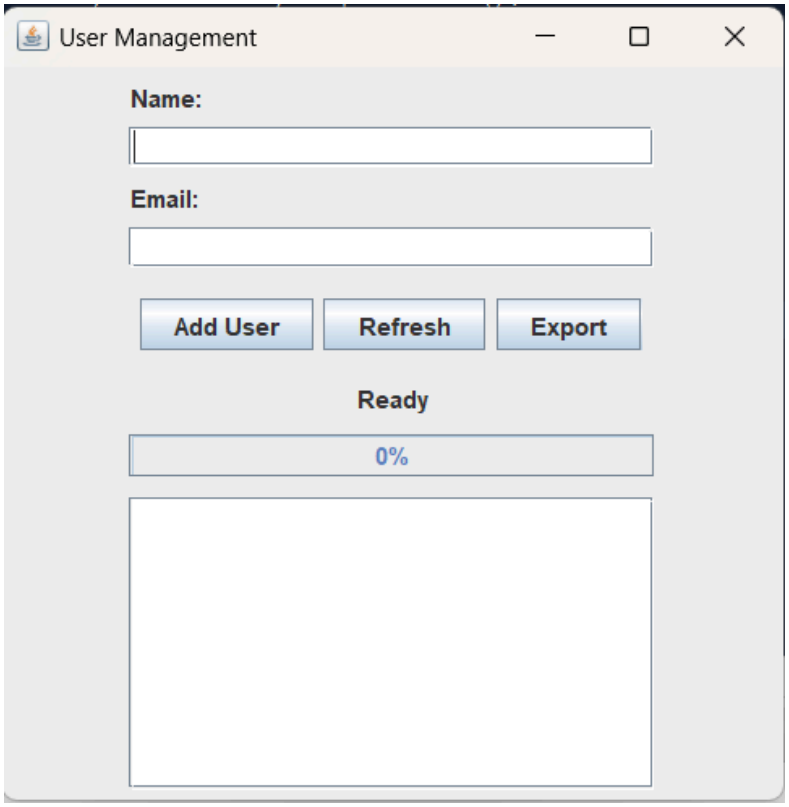
UserController.java
Penjelasan
Kode ini mendemonstrasikan implementasi konkurensi dalam aplikasi Swing untuk menjaga UI tetap responsif selama melakukan operasi yang memakan waktu seperti database dan file I/O.

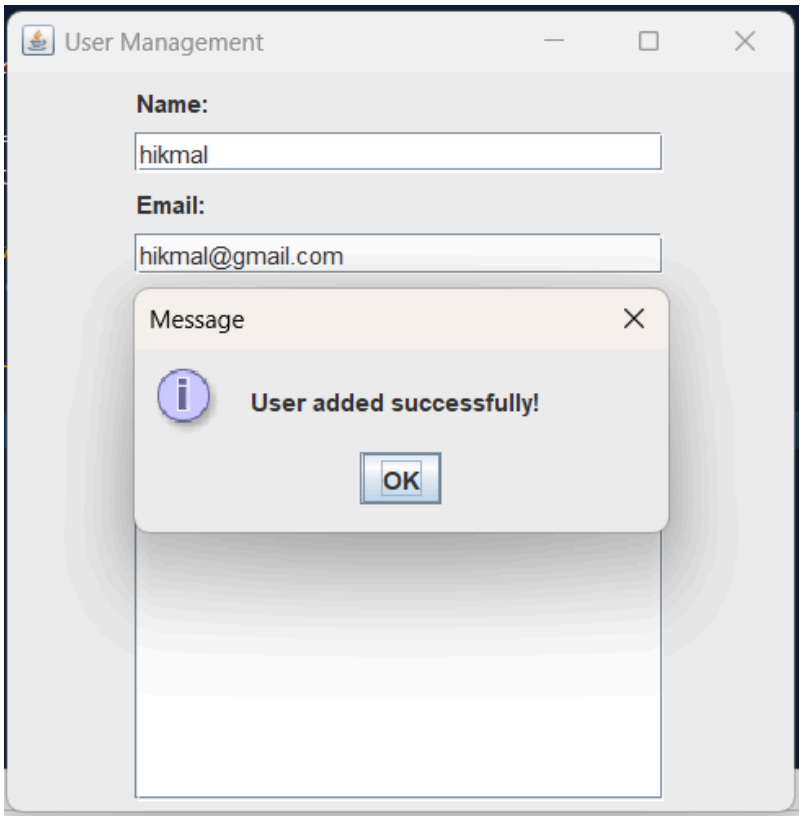
Jika **berkas memiliki output**, gunakan tabel ini.

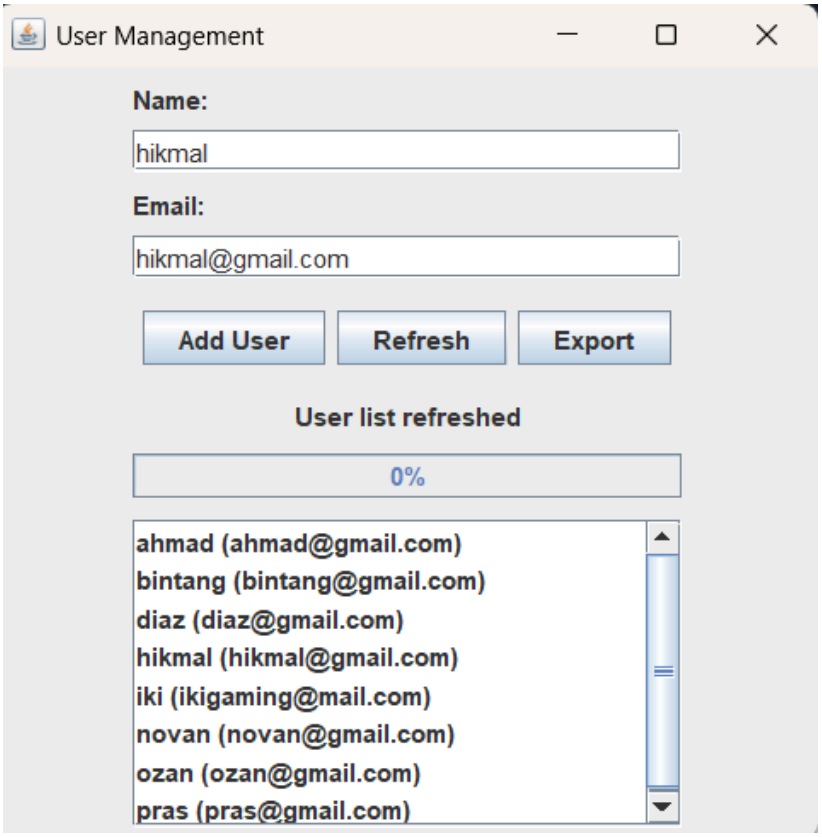
Main.java

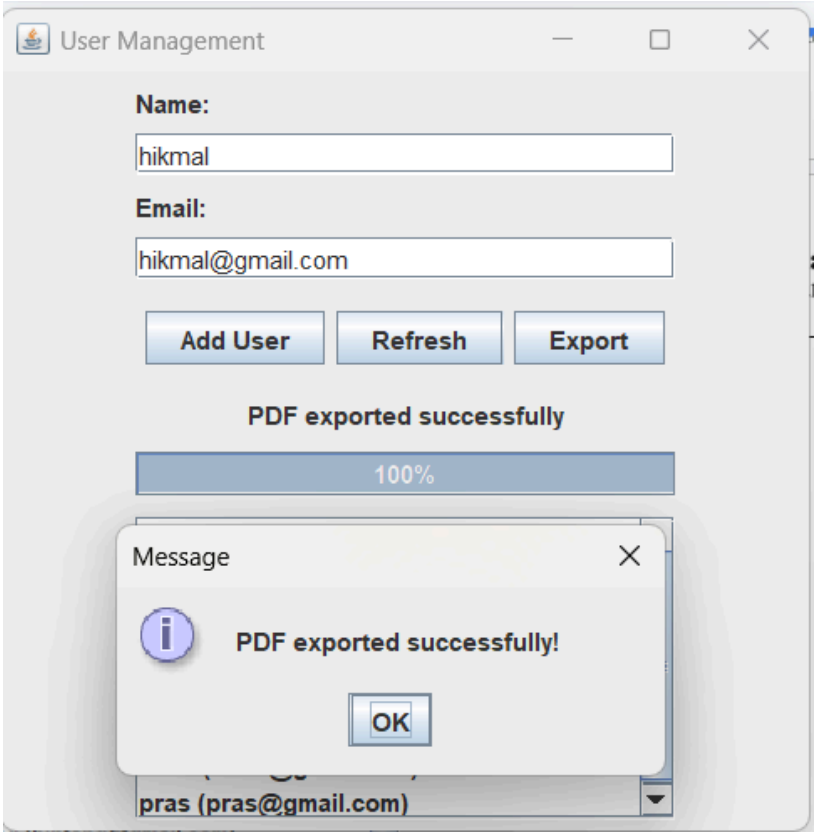
```
package main;

import model.MyBatisUtil;
import model.UserMapper;
import org.apache.ibatis.session.SqlSession;
import view.UserView;
import controller.UserController;

/**
 *
 * @author Diaza
 */
public class Main {
    public static void main(String[] args) {
        SqlSession session = MyBatisUtil.getSqlSession();
        UserMapper mapper = session.getMapper(UserMapper.class);

        UserView view = new UserView();
        new UserController(view, mapper);

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

Penjelasan
Code ini mendefinisikan Kelas Main, yang merupakan entry point dari aplikasi . Fungsinya untuk menginisialisasi dan menghubungkan seluruh komponen dalam arsitektur MVC sehingga aplikasi dalam berjalan. Berkas ini memanfaatkan MyBatis sebagai ORM untuk akses database , serta Swing untuk antarmuka pengguna.
Output


User Management
Name:
Email:
Add User
Refresh
Export
Ready
0%


User Management
Name:
hikmal
Email:
hikmal@gmail.com
Message
User added successfully!
OK


User Management
Name:
hikmal
Email:
hikmal@gmail.com
Add User
Refresh
Export
User list refreshed
0%
ahmad (ahmad@gmail.com)
bintang (bintang@gmail.com)
diaz (diaz@gmail.com)
hikmal (hikmal@gmail.com)
iki (ikigaming@mail.com)
novan (novan@gmail.com)
ozan (ozan@gmail.com)
pras (pras@gmail.com)


User Management
Name:
hikmal
Email:
hikmal@gmail.com
Add User
Refresh
Export
PDF exported successfully
100%
Message
PDF exported successfully!
OK
pras (pras@gmail.com)

No	Name	Email
1	ahmad	ahmad@gmail.com
2	bintang	bintang@gmail.com
3	diaz	diaz@gmail.com
4	hikmal	hikmal@gmail.com
5	iki	ikigaming@mail.com
6	novan	novan@gmail.com
7	ozan	ozan@gmail.com
8	pras	pras@gmail.com