[Pertemuan 8]

Identitas

- NPM : 223040033
- Nama : Muhammad Fauzan Dwi Putera
- Kelas : A
- URL Repository Github :

https://github.com/Fauzan2617/pp2-2024_223040033_A/tree/main/Membership

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_membership" />
                <property name="username" value="root" />
                <property name="password" value="" />
            </dataSource>
        </environment>
    </environments>
    <mappers>
        <package name="mapper" />
    </mappers>
</configuration>

]
```

Penjelasan

berkas ini mengonfigurasi koneksi dan pengaturan MyBatis agar dapat berinteraksi dengan database yang sudah ditentukan.

[JenisMember]

```
[package model; public
class JenisMember{
private String id;
private String nama;

    public void setId(String id){
this.id = id;
    }
    public String getId(){
return id;
    }
    public void setNama(String nama){
this.nama = nama;
    }
    public String getNama(){
return nama;
    }
}
]
```

Penjelasan

Tujuan utama berkas ini adalah untuk menyediakan struktur data yang dapat digunakan untuk menyimpan dan mengakses informasi tentang jenis-jenis member.

[Member]

```
[package model;

public class Member{     private
String id;     private String nama;
private String jenisMemberId;
private JenisMember jenisMember;
     public void setId(String id)
{          this.id = id;
    }
    public String getId() {
return id;
    }      public void setNama(String
nama) {          this.nama = nama;
    }      public String
getNama() {          return
nama;
```

```
    }
    public void setJenisMemberId(String jenisMemberId) {
this.jenisMemberId = jenisMemberId;
    }
    public String getJenisMemberId() {
return jenisMemberId;
    }
    public JenisMember getJenisMember() {
return jenisMember;
    }
    public void setJenisMember(JenisMember jenisMember) {
this.jenisMember = jenisMember;
    }


}
]
```

Penjelasan

Tujuan utama berkas ini adalah untuk menyimpan informasi tentang member, termasuk jenis member yang terkait, dan memberikan metode untuk mengakses serta memodifikasi data tersebut.

[JenisMemberMapper]

```
[package mapper;
 import java.util.List;
import model.JenisMember;
import org.apache.ibatis.annotations.Delete; import
org.apache.ibatis.annotations.Insert; import
org.apache.ibatis.annotations.Result; import
org.apache.ibatis.annotations.Results; import
org.apache.ibatis.annotations.Select; import
org.apache.ibatis.annotations.Update;

public interface JenisMemberMapper {
    @Insert("INSERT INTO jenis_member (id, nama) VALUES(#{id}, #{nama})")
public Integer insert(JenisMember jenisMember);

    @Update("UPDATE jenis_member SET nama = #{nama} WHERE id = #{id}")
public Integer update(JenisMember jenisMember);

    @Delete("DELETE FROM jenis_member WHERE id = #{id}")
public Integer delete(JenisMember jenisMember);

    @Select("SELECT * FROM jenis_member")
    @Results(value = {
        @Result(property = "id", column = "id"),
        @Result(property = "nama", column = "nama")
})
    List<JenisMember> findAll();
```

```
}


]
```

Penjelasan

Tujuan berkas ini adalah untuk menyediakan operasi dasar untuk mengelola data jenis_member dalam aplikasi menggunakan MyBatis.

[MemberMapper]

```
[package mapper;
 import java.util.List;
import model.JenisMember;
import model.Member;
import org.apache.ibatis.annotations.Delete;
import org.apache.ibatis.annotations.Insert;
import org.apache.ibatis.annotations.One; import
org.apache.ibatis.annotations.Result; import
org.apache.ibatis.annotations.Results; import
org.apache.ibatis.annotations.Select; import
org.apache.ibatis.annotations.Update;

public interface MemberMapper {

    @Insert("INSERT INTO member (id, nama, jenis_member_id) VALUES(#{id},
#{nama}, #{jenisMemberId})")
    public Integer insert(Member member);

    @Update("UPDATE member SET nama = #{nama}, jenis_member_id =
#{jenisMemberId} WHERE id = #{id}")
public Integer update(Member member);

    @Delete("DELETE FROM member WHERE id = #{id}")
public Integer delete(Member member);
    @Select("SELECT * FROM member")
    @Results(value = {
        @Result(property = "id", column = "id"),
        @Result(property = "nama", column = "nama"),
        @Result(property = "jenisMember", column = "jenis_member_id",
javaType = JenisMember.class,                  one = @One(select =
"mapper.MemberMapper.getJenisMember"))
    })
    List<Member> findAll();

    @Select("SELECT * FROM jenis_member WHERE id = #{id}")
    @Results(value = {
        @Result(property = "id", column = "id"),
        @Result(property = "nama", column = "nama")
})
    JenisMember getJenisMember(String id); }
```

```
]
```

Penjelasan

Tujuan berkas ini adalah untuk menyediakan operasi dasar pada tabel member dan mengelola relasi antara member dengan jenis_member, sehingga memungkinkan akses data yang lebih kompleks dan terstruktur.

[JenisMemberDao]

```
[package dao;
 import java.util.List;
import model.JenisMember;
import org.apache.ibatis.session.SqlSession; import
org.apache.ibatis.session.SqlSessionFactory;

public class JenisMemberDao {

    private final SqlSessionFactory sqlSessionFactory;

    public JenisMemberDao(SqlSessionFactory sqlSessionFactory) {
this.sqlSessionFactory = sqlSessionFactory;
    }
    public int insert(JenisMember jenisMember){
int result;
        try(SqlSession session = sqlSessionFactory.openSession()){
result = session.insert("mapper.JenisMemberMapper.insert", jenisMember);
            session.commit();
        }

        return result;
    }
    public int update(JenisMember jenisMember){
int result;
        try(SqlSession session = sqlSessionFactory.openSession()){
result = session.update("mapper.JenisMemberMapper.update", jenisMember);
            session.commit();
        }
        return result;
    }      public int delete(String
jenisMemberId){         int result;
        try(SqlSession session = sqlSessionFactory.openSession()){
result = session.delete("mapper.JenisMemberMapper.delete",
jenisMemberId);
            session.commit();
        }
        return result;
    }
    public List<JenisMember> findAll(){
```

```
        List<JenisMember> jenisMembers;
        try(SqlSession session = sqlSessionFactory.openSession()){
jenisMembers =
session.selectList("mapper.JenisMemberMapper.findAll");
}

        return jenisMembers;
    }

}


]
```

Penjelasan

Tujuan dari berkas ini adalah untuk menyediakan lapisan akses data yang memudahkan manipulasi dan pengambilan data JenisMember dari database, sekaligus memisahkan logika akses data dari bagian lain aplikasi.

[MemberDao]

```
[package dao;
 import java.util.List;
import model.Member;
import org.apache.ibatis.session.SqlSession; import
org.apache.ibatis.session.SqlSessionFactory;

public class MemberDao {
    private final SqlSessionFactory sqlSessionFactory;

    public MemberDao(SqlSessionFactory sqlSessionFactory) {
this.sqlSessionFactory = sqlSessionFactory;     }
    public int insert(Member member){
int result;
        try(SqlSession   session   =   sqlSessionFactory.openSession()){
result = session.insert("mapper.MemberMapper.insert", member);
            session.commit();
        }
        return result;
    }
          public int update(Member member){
int result;
        try(SqlSession   session   =   sqlSessionFactory.openSession()){
result = session.update("mapper.MemberMapper.update", member);
            session.commit();
        }
        return result;
    }
```

```
    public int delete(String memberId){
int result;
        try(SqlSession session = sqlSessionFactory.openSession()){
result = session.delete("mapper.MemberMapper.delete", memberId);
            session.commit();
        }
        return result;
    }      public List<Member>
findAll(){
        List<Member> result;
        try(SqlSession session = sqlSessionFactory.openSession()){
result = session.selectList("mapper.MemberMapper.findAll");
        }
        return result;
    }
}



]
```

Penjelasan

Tujuan dari berkas ini adalah untuk menyediakan lapisan akses data yang menangani semua interaksi dengan tabel member di database, menjaga logika aplikasi tetap terpisah dan memudahkan pemeliharaan kode.

Penjelasan Kode

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

[Package(view.jenismember)]

```
[package view.jenismember;

import dao.JenisMemberDao; import
java.awt.event.ActionEvent; import
java.awt.event.ActionListener;
import java.util.UUID; import
model.JenisMember;

public class JenisMemberButtonSimpanActionListener implements
ActionListener {

    private JenisMemberFrame jenisMemberFrame;
private JenisMemberDao jenisMemberDao;

    public
JenisMemberButtonSimpanActionListener(JenisMemberFrame
jenisMemberFrame, JenisMemberDao jenisMemberDao) {
this.jenisMemberFrame = jenisMemberFrame;
this.jenisMemberDao = jenisMemberDao;     }

    @Override
    public void actionPerformed(ActionEvent e) {
String nama = this.jenisMemberFrame.getName();
```

```
        JenisMember jenisMember = new JenisMember();
jenisMember.setId(UUID.randomUUID().toString());
jenisMember.setNama(nama);

        this.jenisMemberFrame.addJenisMember(jenisMember);
this.jenisMemberDao.insert(jenisMember);
    }

}
]



[package view.jenismember;

import dao.JenisMemberDao; import
java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.List; import
javax.swing.JButton; import
javax.swing.JFrame; import
javax.swing.JLabel; import
javax.swing.JScrollPane; import
javax.swing.JTable; import
javax.swing.JTextField; import
model.JenisMember;
import view.jenismemberdetail.JenisMemberDetailFrame;

public class JenisMemberFrame extends JFrame {

    private List<JenisMember>
jenisMemberList;     private JTextField
textFieldNama;     private
JenisMemberTableModel tableModel;     private
JenisMemberDao jenisMemberDao;

    public JenisMemberFrame(JenisMemberDao jenisMemberDao){
this.jenisMemberDao = jenisMemberDao;
        this.jenisMemberList = jenisMemberDao.findAll();
this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        JLabel labelInput = new JLabel("Nama: ");
labelInput.setBounds(15,40,350,10);

        textFieldNama = new JTextField();
textFieldNama.setBounds(15,60,350,30);

        JButton button = new JButton("Simpan");
button.setBounds(15,100,100,40);

        javax.swing.JTable table = new JTable();
        JScrollPane scrollableTable = new JScrollPane(table);
scrollableTable.setBounds(15,150,350,200);

        tableModel = new
JenisMemberTableModel(jenisMemberList);
table.setModel(tableModel);
        JenisMemberButtonSimpanActionListener actionListener = new
JenisMemberButtonSimpanActionListener(this,  jenisMemberDao);
```

```
        button.addActionListener(actionListener);
        table.addMouseListener(new MouseAdapter()
{          public void mouseClicked(MouseEvent e)
{
                    if (e.getClickCount() == 2) { // Double click untuk
membuka detail
                        int row = table.getSelectedRow();
JenisMember selectedJenisMember = jenisMemberList.get(row);
                        JenisMemberDetailFrame detailFrame = new
JenisMemberDetailFrame(selectedJenisMember, jenisMemberDao);

                        detailFrame.setVisible(true);
                    }
                }
        });

        this.add(button);
this.add(textFieldNama);
this.add(labelInput);
this.add(scrollableTable);

this.setSize(400,500);
this.setLayout(null);

    }
    public String getName(){
        return textFieldNama.getText();
    }
    public void addJenisMember(JenisMember jenisMember){
tableModel.add(jenisMember);
textFieldNama.setText("");
    }

}
]

[package view.jenismember;

import java.util.List;
import javax.swing.table.AbstractTableModel;
import model.JenisMember;
 public class JenisMemberTableModel extends
AbstractTableModel{     private String[] columnNames =
{"Nama"};     private List<JenisMember> data;

    public JenisMemberTableModel(List<JenisMember> data) {
this.data = data;
    }      public int
getColumnCount() {          return
columnNames.length;
    }
    public int getRowCount() {
```

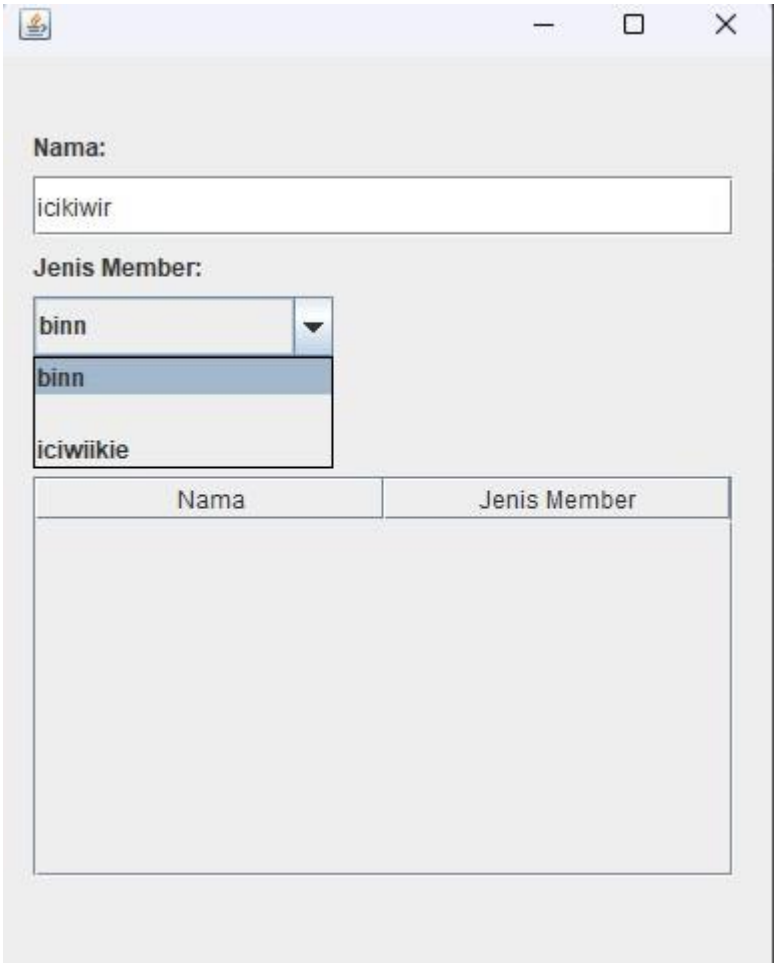
```
        return data.size();
    }
    public String getColumnName(int column) {
return columnNames[column];
    }      public Object getValueAt(int row,
int col) {
        JenisMember rowItem = data.get(row);
        String value="";

        switch(col){
case 0:
                value = rowItem.getNama();
break;
        }
        return value;
    }
    public boolean isCellEditable(int row, int col) {
return false;
    }
    public void add(JenisMember value){
data.add(value);
        fireTableRowsInserted(data.size() - 1, data.size() -
1);     }
    public void setData(List<JenisMember> data)
{         this.data = data;
fireTableDataChanged();
    }
}

]
```

Penjelasan

Tujuan keseluruhan dari berkas-berkas ini adalah untuk menyediakan antarmuka yang memungkinkan pengguna untuk:

- Menambah, mengedit, dan menampilkan data JenisMember secara interaktif.
- Menggunakan frame yang mencakup form input dan tabel, dengan komunikasi langsung ke database untuk menyimpan dan mengambil data JenisMember melalui DAO (JenisMemberDao).

Output



[Package(view.jenismemberdetail)]

```
[package view.jenismemberdetail;
 import
dao.JenisMemberDao;
import dao.MemberDao;
import java.awt.event.ActionEvent;
import
java.awt.event.ActionListener;
import java.util.List; import
javax.swing.JButton; import
javax.swing.JComboBox; import
javax.swing.JFrame; import
javax.swing.JLabel; import
javax.swing.JTextField; import
model.JenisMember;
 public class JenisMemberDetailFrame extends JFrame
{
     private JTextField textFieldId;
private JTextField textFieldNama;
private JenisMember jenisMember;
private JenisMemberDao jenisMemberDao;
private JComboBox comboJenis;
    private List<JenisMember> jenisMemberList;

    public JenisMemberDetailFrame(JenisMember jenisMember,JenisMemberDao
```

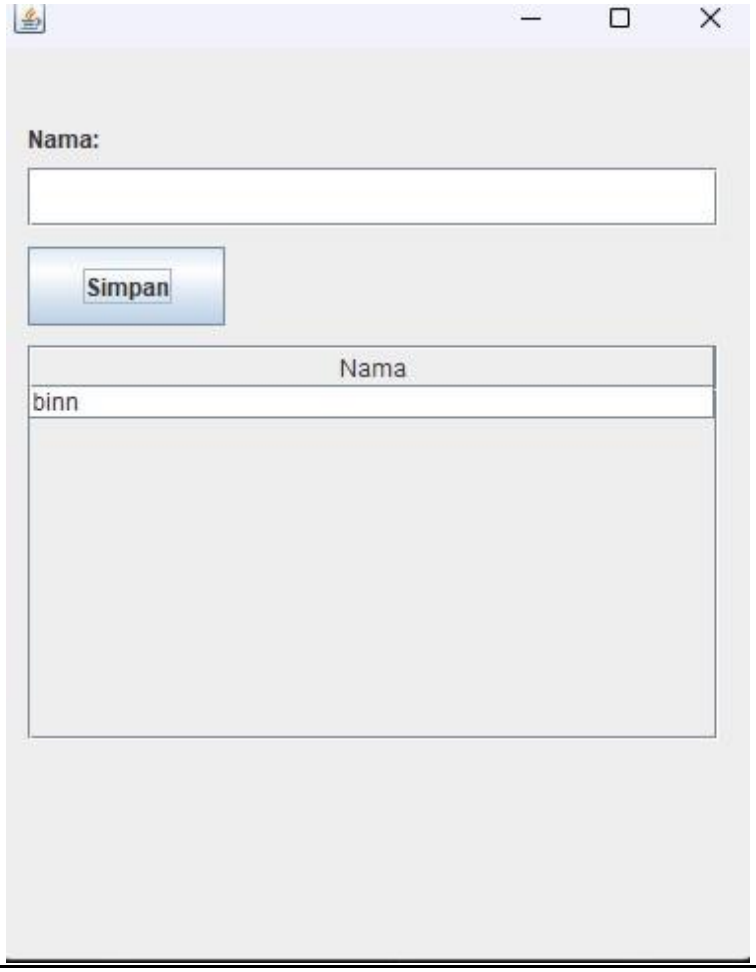
```
jenisMemberDao) {
        this.jenisMember = jenisMember;
this.jenisMemberDao = jenisMemberDao;
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        JLabel labelId = new JLabel("ID:");
labelId.setBounds(15, 40, 350, 10);
        textFieldId = new JTextField(jenisMember.getId());
textFieldId.setBounds(15, 60, 350, 30);
        textFieldId.setEditable(false);  // ID tidak bisa diedit
        JLabel labelNama = new JLabel("Nama:");
labelNama.setBounds(15, 100, 350, 10);
        textFieldNama = new JTextField(jenisMember.getNama());
textFieldNama.setBounds(15, 120, 350, 30);

        JButton buttonUpdate = new JButton("Update");
buttonUpdate.setBounds(15, 240, 100, 40);
        buttonUpdate.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
updateMember();
            }
        });

        JButton buttonDelete = new JButton("Delete");
buttonDelete.setBounds(125, 240, 100, 40);
        buttonDelete.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
deleteMember();
            }
        });
         this.add(labelId);
this.add(textFieldId);
this.add(labelNama);
this.add(textFieldNama);
this.add(buttonUpdate);
this.add(buttonDelete);

        this.setSize(400, 500);
this.setLayout(null);
    }

    private void updateMember() {
        jenisMember.setNama(textFieldNama.getText());
jenisMemberDao.update(jenisMember);
        this.dispose(); // Tutup window setelah update
// Refresh tabel di MemberFrame (lihat langkah 3)     }
    private void deleteMember() {
        jenisMemberDao.delete(jenisMember.getId());   //Asumsi   ID   adalah
integer
        this.dispose(); // Tutup window setelah delete
        // Refresh tabel di MemberFrame (lihat langkah 3)
    }
```

```
}
]
```

Penjelasan

Berkas ini menyediakan antarmuka untuk pengguna untuk memperbarui atau menghapus entitas JenisMember dalam aplikasi. Ini memungkinkan pengguna untuk mengelola data jenis member dengan cara yang lebih interaktif dan mudah. Setelah operasi dilakukan, frame ini akan ditutup untuk memberi ruang bagi pengguna untuk melanjutkan interaksi dengan aplikasi.

Output



[Package(view.main)]

```
[package view.main;
 import java.awt.event.ActionEvent;
import
java.awt.event.ActionListener;

public class MainButtonActionListener implements ActionListener {

    private MainFrame mainFrame;
```

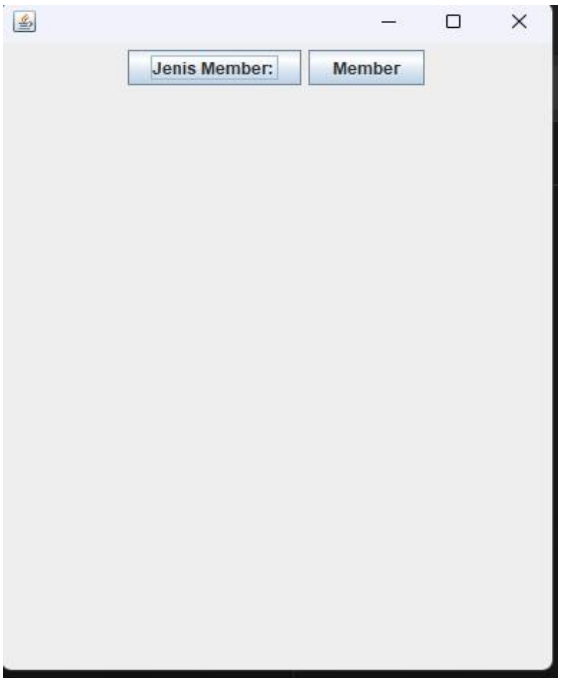
```
    public MainButtonActionListener(MainFrame mainFrame) {
this.mainFrame = mainFrame;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == mainFrame.getButtonJenisMember()){
mainFrame.showJenisMemberFrame();
        } else {
            mainFrame.showMemberFrame();
        }
    }
}

]

[package view.main;
 import
dao.JenisMemberDao; import
dao.MemberDao; import
java.awt.FlowLayout;
import
javax.swing.JButton;
import javax.swing.JFrame;
import view.jenismember.JenisMemberFrame;
import view.member.MemberFrame;

public class MainFrame extends JFrame {
private JenisMemberFrame jenisMemberFrame ;
private MemberFrame memberFrame;     private
JButton buttonJenisMember;     private JButton
buttonMember;     private JenisMemberDao
jenisMemberDao;     private MemberDao
memberDao;

    public MainFrame(JenisMemberDao jenisMemberDao, MemberDao memberDao){
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        this.setSize(400,500);

        this.jenisMemberDao = jenisMemberDao;
this.memberDao = memberDao;
         this.jenisMemberFrame = new
JenisMemberFrame(jenisMemberDao);         this.memberFrame = new
MemberFrame(memberDao, jenisMemberDao);

        this.setLayout(new FlowLayout());
        MainButtonActionListener actionListener = new
MainButtonActionListener(this);
         this.buttonJenisMember = new JButton("Jenis Member:
");          this.buttonMember = new JButton("Member");

        this.buttonJenisMember.addActionListener(actionListener);
this.buttonMember.addActionListener(actionListener);

this.add(buttonJenisMember);
this.add(buttonMember);
    }
```

```
    public JButton getButtonJenisMember(){
return buttonJenisMember;
    }
    public JButton getButtonMember(){
return buttonMember;
    }      public void
showJenisMemberFrame(){
if(jenisMemberFrame == null){
            jenisMemberFrame = new JenisMemberFrame(jenisMemberDao);
        }
        jenisMemberFrame.setVisible(true);
    }
    public void showMemberFrame(){

        if(memberFrame == null){
            memberFrame = new MemberFrame(memberDao, jenisMemberDao);
        }
        memberFrame.populateComboJenis();
memberFrame.setVisible(true);
    }
}

]
```

Penjelasan

Berkas ini bertindak sebagai jembatan antar dua fitur utama dalam aplikasi. Pengguna dapat memilih apakah mereka ingin mengelola **Jenis Member** atau **Member**, dan aplikasi akan menampilkan antarmuka yang sesuai dengan fungsionalitas yang diinginkan. Setelah memilih tombol, aplikasi akan menampilkan window baru untuk masing-masing kategori dengan cara yang intuitif.

Output



[Package(view.member)]

```
[package view.member;

import dao.MemberDao;
import java.awt.event.ActionEvent;
import
java.awt.event.ActionListener;
import java.util.UUID; import
model.JenisMember; import
model.Member;

public class MemberButtonSimpanActionListener implements ActionListener {
     private MemberFrame
memberFrame;     private MemberDao
memberDao;
     public             MemberButtonSimpanActionListener(MemberFrame
memberFrame, MemberDao memberDao) {
        this.memberFrame = memberFrame;
this.memberDao = memberDao;
    }      public void
actionPerformed(ActionEvent e){         String
nama = this.memberFrame.getName();
if(nama.isEmpty()){
            this.memberFrame.showAlert("Nama tidak boleh kosong");
        } else {
            JenisMember jenisMember = this.memberFrame.getJenisMember();
Member member = new Member();
            member.setId(UUID.randomUUID().toString());
member.setNama(nama);
            member.setJenisMember(jenisMember);
member.setJenisMemberId(jenisMember.getId());
this.memberFrame.addMember(member);
this.memberDao.insert(member);
```

```
        }
    }

}
]

[package view.member;
 import dao.JenisMemberDao; import
dao.MemberDao; import
java.awt.event.MouseAdapter; import
java.awt.event.MouseEvent; import
java.util.List; import javax.swing.JButton;
import javax.swing.JComboBox; import
javax.swing.JFrame; import javax.swing.JLabel;
import javax.swing.JOptionPane; import
javax.swing.JScrollPane; import
javax.swing.JTable; import
javax.swing.JTextField; import
model.JenisMember; // Import class baru import
model.Member;
import view.memberdetail.MemberDetailFrame;

public class MemberFrame extends JFrame {

    private List<JenisMember> jenisMemberList;     private
List<Member> memberList;     private JTextField
textFieldNama;     private MemberTableModel tableModel;
private JComboBox comboJenis;     private MemberDao
memberDao;     private JenisMemberDao jenisMemberDao;
public MemberFrame(MemberDao memberDao, JenisMemberDao
jenisMemberDao){
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
this.memberDao = memberDao;         this.jenisMemberDao =
jenisMemberDao;

        this.memberList = this.memberDao.findAll();
this.jenisMemberList = this.jenisMemberDao.findAll();
        JLabel labelInput = new JLabel("Nama: ");
labelInput.setBounds(15,40,350,10);
         textFieldNama = new JTextField();
textFieldNama.setBounds(15,60,350,30);

       JLabel labelJenis = new JLabel("Jenis Member: ");
labelJenis.setBounds(15,100,350,10);

       comboJenis = new JComboBox();
comboJenis.setBounds(15,120,150,30);
```

```
JButton button = new JButton("Simpan");
```

```
button.setBounds(15,160,100,40);    javax.swing.JTable table = new
JTable();
       JScrollPane scrollableTable = new JScrollPane(table);
scrollableTable.setBounds(15,210,350,200);

        tableModel = new MemberTableModel(memberList);
table.setModel(tableModel);

        MemberButtonSimpanActionListener actionListener = new
MemberButtonSimpanActionListener(this, memberDao);

        button.addActionListener(actionListener);
         table.addMouseListener(new MouseAdapter()
{          public void mouseClicked(MouseEvent e) {
                    if (e.getClickCount() == 2) { // Double click untuk
membuka detail
                        int row = table.getSelectedRow();
                        Member selectedMember = memberList.get(row);
                        MemberDetailFrame detailFrame = new
MemberDetailFrame(selectedMember, memberDao, jenisMemberDao);
detailFrame.populateComboJenis();
detailFrame.setVisible(true);
                    }
                }
         });
        this.add(button);
this.add(textFieldNama);
this.add(labelInput);
this.add(scrollableTable);
this.add(labelJenis);
this.add(comboJenis);

        this.setSize(400,500);
this.setLayout(null);
    }
    public void populateComboJenis(){
        this.jenisMemberList = this.jenisMemberDao.findAll();
comboJenis.removeAllItems();
        for(JenisMember jenisMember: this.jenisMemberList){
comboJenis.addItem(jenisMember.getNama());
        }
    }
    public String getName(){
        return textFieldNama.getText();
    }      public JenisMember
getJenisMember(){
       return jenisMemberList.get(comboJenis.getSelectedIndex());
   }
```

```
   public void addMember(Member member){
tableModel.add(member);         textFieldNama.setText("");
   }

   public void showAlert(String message){
       JOptionPane.showMessageDialog(this, message);
```

```
}
```

```
}

]

[    package view.member;

    import java.util.List;
    import javax.swing.table.AbstractTableModel;
import model.Member;

    public class MemberTableModel extends AbstractTableModel {

        private String[] columnNames = { "Nama", "Jenis Member"};
private List<Member> data;

        public MemberTableModel(List<Member> data) {
this.data = data;
        }          public int
getColumnCount() {             return
columnNames.length;
        }          public int
getRowCount() {             return
data.size();
        }
        public String getColumnName(int col){
return columnNames[col];
        }
        public Object getValueAt(int row, int col) {
            Member rowItem = data.get(row);
            String value = "";

            switch(col) {                 case
0:                      value =
rowItem.getNama();                      break;
case 1:
                    // Add null check for getJenisMember()
if (rowItem.getJenisMember() != null) {
value = rowItem.getJenisMember().getNama();
                    } else {
                        value = "No Jenis Member";  // or any default
value
                    }
break;
            }
           return value;
       }
```

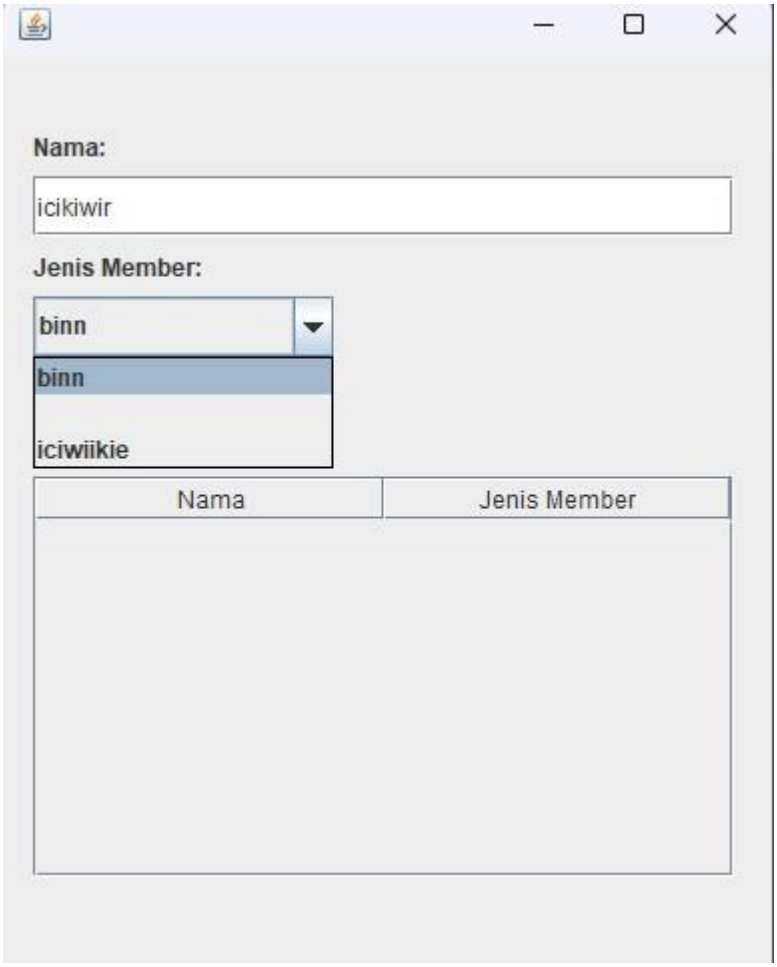
```
       public boolean isCellEditable(int row, int col){
return false;
       }         public void add(Member

value){
```

```
            data.add(value);
            fireTableRowsInserted(data.size() - 1, data.size() - 1);
}

        public void removeMember(Member member) {
data.remove(member);
            fireTableRowsDeleted(data.size() - 1, data.size() - 1);
}

        public void setData(List<Member> data) {
this.data = data;
        }
    }

]
```

Penjelasan

- **MemberFrame** bertindak sebagai antarmuka untuk manajemen member, memungkinkan pengguna untuk menambah dan melihat member yang ada, serta mengelola jenis member yang terhubung.
- **MemberButtonSimpanActionListener** memastikan bahwa data yang dimasukkan valid dan kemudian menyimpannya ke database.
- **MemberTableModel** mengelola tampilan data member dalam tabel, memungkinkan operasi dasar seperti penambahan dan penghapusan.

Output



[Package(view.memberdetail)]

```
[package view.memberdetail;
 import
dao.JenisMemberDao;
import dao.MemberDao;
import java.awt.event.ActionEvent;
import
java.awt.event.ActionListener;
import java.util.List; import
javax.swing.JButton; import
javax.swing.JComboBox; import
javax.swing.JFrame; import
javax.swing.JLabel; import
javax.swing.JTextField; import
model.JenisMember; import
model.Member;
 public class MemberDetailFrame extends JFrame
{

    private JTextField textFieldId;
private JTextField textFieldNama;
private Member member;     private
MemberDao memberDao;     private
JComboBox comboJenis;
```

```
    private List<JenisMember> jenisMemberList;
private JenisMemberDao jenisMemberDao;
```

```
    public MemberDetailFrame(Member member, MemberDao memberDao,
JenisMemberDao jenisMemberDao) {
this.member = member;          this.memberDao
= memberDao;          this.jenisMemberDao =
jenisMemberDao;
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        JLabel labelId = new JLabel("ID:");
labelId.setBounds(15, 40, 350, 10);
textFieldId = new JTextField(member.getId());
textFieldId.setBounds(15, 60, 350, 30);
        textFieldId.setEditable(false);  // ID tidak bisa diedit
        JLabel labelNama = new JLabel("Nama:");
labelNama.setBounds(15, 100, 350, 10);
        textFieldNama = new JTextField(member.getNama());
textFieldNama.setBounds(15, 120, 350, 30);

        JLabel labelJenis = new JLabel("Jenis Member: ");
labelJenis.setBounds(15,160,350,10);

        comboJenis = new JComboBox();
comboJenis.setBounds(15,180,350,30);

        JButton buttonUpdate = new JButton("Update");
buttonUpdate.setBounds(15, 240, 100, 40);
        buttonUpdate.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
updateMember();
            }
        });

        JButton buttonDelete = new JButton("Delete");
buttonDelete.setBounds(125, 240, 100, 40);
        buttonDelete.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
deleteMember();
            }
        });
         this.add(labelId);
this.add(textFieldId);
this.add(labelNama);
this.add(textFieldNama);
this.add(buttonUpdate);
this.add(buttonDelete);
this.add(labelJenis);
this.add(comboJenis);
         this.setSize(400,
500);
this.setLayout(null);
    }
    public void populateComboJenis(){
        this.jenisMemberList = this.jenisMemberDao.findAll();
comboJenis.removeAllItems();
        for(JenisMember jenisMember: this.jenisMemberList){
```

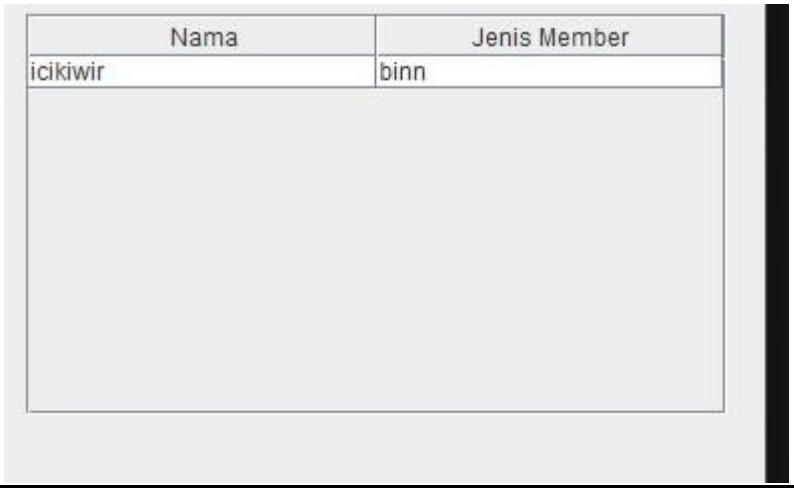
```
            comboJenis.addItem(jenisMember.getNama());
        }
    }
    private void updateMember() {
        JenisMember jenisMember = this.getJenisMember();
member.setNama(textFieldNama.getText());
member.setJenisMember(jenisMember);
member.setJenisMemberId(jenisMember.getId());
memberDao.update(member);
        this.dispose(); // Tutup window setelah update
// Refresh tabel di MemberFrame (lihat langkah 3)     }
    private void deleteMember() {
        memberDao.delete(member.getId()); //Asumsi ID adalah integer
this.dispose(); // Tutup window setelah delete
        // Refresh tabel di MemberFrame (lihat langkah 3)
}
    public JenisMember getJenisMember(){
        return    jenisMemberList.get(comboJenis.getSelectedIndex());
}


}
]
```

Penjelasan

- **MemberDetailFrame** bertujuan untuk memberikan detail lebih lanjut tentang member yang dipilih dari daftar di **MemberFrame** dan memungkinkan pengguna untuk memperbarui atau menghapus data member tersebut.
- Mengelola hubungan antara member dan jenis member, dengan memberikan opsi untuk memilih jenis member yang sesuai melalui **JComboBox**.

Secara keseluruhan, berkas ini merupakan bagian dari aplikasi manajemen member yang memungkinkan manipulasi data member secara mendetail.

Output



Tugas
Praktikum Pemrograman II

Catatan:

- Silakan modifikasi dan tambahkan informasi sesuai dengan proyek yang Anda kerjakan baik pada tanda [kalimat].
- Sesuaikan nama file laporan dengan format **Tugas[Pertemuan ke-X]_PP1_NPM** Contoh: Tugas[P10]_PP1_203040104
- Sesuaikan judul laporan dengan format **Laporan Tugas Pertemuan ke-X** Contoh: Laporan Tugas Pertemuan ke-10