LAPORAN PROYEK AKHIR PERANCANGAN APLIKASI

“APLIKASI MY MENTAL HEALTH”

Mata Kuliah : Pemrograman Berorientasi Objek

Dosen Pengampu : Imam Adi Nata M.Kom

Disusun Oleh:

Eka Makhsusan/ 2320506042

Dhiya Rahma Azifah/ 2340506073

Fatina Sabitul Azmi/ 2340506076

Nasywaa Jihaan Labiibah/ 2340506078

Program Studi S1 Teknologi Informasi

Fakultas Teknik, Universitas Tidar

Ganjil 2024

I. PENDAHULUAN

Pemrograman Berorientasi Objek (PBO) adalah paradigma pemrograman yang berfokus pada penggunaan objek untuk merepresentasikan data dan perilaku dalam aplikasi. Dalam PBO, objek adalah unit dasar yang menggabungkan data (atribut) dan metode (fungsi) yang bekerja pada data tersebut. Pendekatan ini memungkinkan pengembang untuk menciptakan program yang lebih modular, mudah dikelola, dan dapat digunakan kembali.

Java, sebagai salah satu bahasa pemrograman yang paling populer, sepenuhnya mendukung PBO. Dengan fitur-fitur seperti kelas, objek, pewarisan, polimorfisme, dan enkapsulasi, Java memungkinkan pengembang untuk membangun aplikasi yang kompleks dengan cara yang terstruktur dan efisien. Selain itu, Java juga menawarkan kelebihan seperti platform-independensi, keamanan, dan kemudahan penggunaan, menjadikannya pilihan yang ideal untuk pengembangan perangkat lunak.

Melalui praktikum ini, peserta akan mempelajari konsep-konsep dasar PBO dalam Java, termasuk cara mendefinisikan kelas dan objek, menerapkan enkapsulasi, pewarisan, polimorfisme, dan memanfaatkan interface serta kelas abstrak. Selain itu, peserta juga akan belajar tentang komposisi, agregasi, serta teknik file I/O dan serialisasi untuk menyimpan dan mengelola data. Dengan pemahaman yang mendalam tentang prinsip-prinsip ini, diharapkan peserta dapat merancang dan mengembangkan aplikasi perangkat lunak yang efisien dan terstruktur.

a. Tujuan

1) Enkapsulasi: Memahami dan menerapkan perlindungan data menggunakan metode getter dan setter.
2) Kelas dan Objek: Menguasai pembuatan kelas dan objek, serta perbedaan antara keduanya.
3) Bekerja Dengan Java: Mengetahui struktur dasar program Java dan cara menjalankannya.

4) Polimorfisme: Menerapkan method overloading dan overriding untuk meningkatkan fleksibilitas kode.Pewarisan: Mengorganisir kode secara hierarkis dengan menggunakan pewarisan.
5) Abstraksi dan Kelas Abstrak: Mendefinisikan kelas abstrak untuk menyederhanakan kompleksitas sistem.
6) Interface: Menggunakan interface untuk mendukung polimorfisme dan modularitas.
7) Komposisi dan Agregasi: Memahami hubungan antar objek melalui komposisi dan agregasi.
8) File I/O dan Serialisasi: Menguasai teknik membaca/menulis file serta proses serialisasi dan deserialisasi.

b. Dasar Teori

1. Enkapsulasi

   Enkapsulasi adalah konsep dasar dalam PBO yang mengamankan data dalam suatu kelas dengan menyembunyikan detail implementasi dari pengguna luar. Dengan enkapsulasi, atribut di dalam kelas biasanya dideklarasikan sebagai private, dan akses ke atribut tersebut dilakukan melalui metode publik (getter dan setter). Ini tidak hanya melindungi data dari akses langsung tetapi juga memungkinkan pengendalian akses dan validasi data sebelum modifikasi.

2. Kelas dan Objek

   Kelas merupakan blueprint atau template yang mendefinisikan atribut (data) dan metode (fungsi) yang dimiliki oleh objek. Objek adalah instance dari kelas yang memiliki keadaan (state) dan perilaku (behavior) berdasarkan definisi kelas. Setiap objek dapat memiliki nilai atribut yang berbeda, tetapi semua objek dari kelas yang sama akan memiliki metode yang sama.

3. Bekerja Dengan Java

   Java adalah bahasa pemrograman berorientasi objek yang dirancang untuk memiliki sifat platform-independen, artinya program Java dapat dijalankan

di berbagai sistem operasi tanpa perlu diubah. Java menggunakan Java Virtual Machine (JVM) untuk mengeksekusi kode byte yang dihasilkan oleh compiler Java, sehingga menjadikannya mudah disebarkan dan digunakan di berbagai lingkungan.

4. Polimorfisme

   Polimorfisme adalah kemampuan untuk menggunakan metode yang sama dengan cara yang berbeda, tergantung pada konteks objek yang memanggilnya. Ada dua jenis polimorfisme:

   a) Polimorfisme pada waktu kompilasi, yang terjadi melalui method overloading (metode dengan nama yang sama tetapi dengan parameter yang berbeda), dan
   b) Polimorfisme pada waktu eksekusi, yang terjadi melalui method overriding (metode di subkelas yang menggantikan metode di superkelas). Ini memungkinkan fleksibilitas dalam penggunaan kelas dan objek.

5. Pewarisan

   Pewarisan adalah mekanisme di mana satu kelas (subkelas) mewarisi atribut dan metode dari kelas lain (superkelas). Ini memungkinkan kode untuk digunakan kembali dan memperluas fungsionalitas tanpa perlu menduplikasi kode. Subkelas dapat menambah atau mengubah perilaku yang diwarisi dari superkelas, yang meningkatkan modularitas dan pengorganisasian kode.

6. Abstraksi dan Kelas Abstrak

   Abstraksi adalah proses menyembunyikan detail kompleks dan hanya menampilkan fitur penting kepada pengguna. Kelas abstrak adalah kelas yang tidak dapat diinstansiasi secara langsung dan dapat memiliki metode abstrak (metode tanpa implementasi). Subkelas dari kelas abstrak harus mengimplementasikan metode abstrak tersebut. Ini berguna untuk mendefinisikan kerangka kerja umum untuk kelas-kelas terkait.

7. Interface

Interface adalah kontrak yang mendefinisikan metode tanpa memberikan implementasi. Kelas yang mengimplementasikan interface harus menyediakan implementasi untuk semua metode yang didefinisikan dalam interface tersebut. Interface memungkinkan penerapan polimorfisme dan mendukung desain yang lebih fleksibel dan terpisah antara spesifikasi dan implementasi.

8. Komposisi dan Agregasi
   a) Komposisi adalah hubungan yang menunjukkan bahwa satu objek (bagian) tidak dapat eksis tanpa objek lain (keseluruhan). Dalam komposisi, jika keseluruhan dihapus, bagian juga akan hilang.
   b) Agregasi adalah hubungan yang menunjukkan bahwa satu objek (bagian) dapat eksis secara independen dari objek lainnya (keseluruhan). Dalam agregasi, bagian bisa ada tanpa keseluruhan, tetapi tetap memiliki hubungan dengan objek tersebut.
9. File I/O dan Serialisasi

   File I/O di Java memungkinkan pembacaan dan penulisan data dari dan ke file. Java menyediakan berbagai kelas untuk menangani operasi ini, seperti FileReader, FileWriter, BufferedReader, dan BufferedWriter. Serialisasi adalah proses mengubah objek menjadi aliran byte sehingga dapat disimpan atau dikirim melalui jaringan. Deserialisasi adalah proses mengubah aliran byte kembali menjadi objek. Ini sangat berguna untuk menyimpan status objek atau mengirim objek melalui jaringan.

II. METODE PRAKTIKUM

a. Alat dan Bahan

1) PC/Laptop dengan koneksi internet
2) Apache NetBeans

b. Langkah Kerja

1. Membuat rancangan dan menentukan fitur apa saja yang akan diterapkan pada aplikasi my mental health.
2. Setelah menentukan fitur yang akan dibuat, lanjut untuk membuat kode program dengan bahasa java yang dilakukan pada IDE Java NetBeans.
3. Langkah pertama dalam pembuatan kode ini dengan membuat kelas untuk masing-masing fitur aplikasi. Yang mana fitur aplikasi terdapat forum, jurnal, meditation, dan juga mood. Kemudian kelas ditambah dengan kelas utama atau main, dan juga menambahkan kelas splash untuk tampilan awal menjalankan aplikasi.
4. Pertama membuat kelas forum untuk halaman utama sebelum masuk pada tampilan sebenarnya.

```
package MentalHealth;

import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author MSI THIN
 */
public class Forum extends javax.swing.JFrame {

    /**
     * Creates new form Forum
     */
    public Forum() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code
```

Gambar 2.b.1

```
    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Mainmenu mm2 = new Mainmenu();
        mm2.show();

        dispose();
    }

    private void chatActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Forum1 f1 = new Forum1();
        f1.show();

    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Forum2 f2 = new Forum2();
        f2.show();

    }
```

Gambar 2.b.2

```
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Forum().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton chat;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton7;
    private javax.swing.JButton jButton8;
    private javax.swing.JButton jButton9;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel10;
    private javax.swing.JPanel jPanel12;
    private javax.swing.JPanel jPanel9;
    // End of variables declaration
}
```

Gambar 2.b.3

5. Kemudian membuat kelas baru dengan nama forum1 yang digunakan untuk buka server pada port, kemudian menunggu klien terhubung melalui ServerSocket, baca pesan dari klien menggunakan DataInputStream, serta tampilkan pesan di jTextArea.

```java
package MentalHealth;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;

/**
 *
 * @author MSI THIN
 */
public class Forum1 extends javax.swing.JFrame {
static Socket socket;
static DataInputStream dis;
static DataOutputStream dout;
    /**
     * Creates new form Forum1
     */
    public Forum1() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code
```

Gambar 2.b.4

```java
private void jButtonConnectActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new Thread(new Runnable(){
    @Override
    public void run(){
        try{
            String ip = jTextFieldIp.getText();
            int port = Integer.valueOf(jTextFieldPort.getText());
            String messageIn ="";
            socket = new Socket(ip,port);
            if(socket.isConnected()){
                jTextArea.append("Connected Successfully.");
            }
            dis = new DataInputStream(socket.getInputStream());
            dout = new DataOutputStream(socket.getOutputStream());

            while(!messageIn.equals("Out")){
                    messageIn = dis.readUTF();
                    jTextArea.append(jTextFieldMessage.getText()+ "\nServer :"+ messageIn);
            }
        }catch (Exception e){
            System.out.println(e.getMessage());

        }
    }
    )).start();
}
```

Gambar 2.b.5

```
private void jButtonSendActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        String message = "";
        message = jTextFieldMessage.getText();
        jTextArea.append("\nClient : " + message);
        dout.writeUTF(message);
        jTextFieldMessage.setText("");

    }catch (Exception e){
        System.out.println(e.getMessage());
    }
}

private void jTextFieldPortActionPerformed(java.awt.event.ActionEvent evt)
    // TODO add your handling code here:
}

private void jTextFieldIpActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
```

Gambar 2.b.6

```
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
            new Forum1().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButtonConnect;
    private javax.swing.JButton jButtonSend;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea;
    private javax.swing.JTextField jTextFieldIp;
    private javax.swing.JTextField jTextFieldMessage;
    private javax.swing.JTextField jTextFieldPort;
    // End of variables declaration
}
```

Gambar 2.b.7

6. Selanjutnya membuat kelas forum2 yang bertujuan untuk mengirim pesan yang berupa teks diketik di dalam jTextFieldMessage ke klien menggunakan DataOutputStream.

```
package MentalHealth;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

/**
 *
 * @author MSI THIN
 */
public class Forum2 extends javax.swing.JFrame {
static ServerSocket serverSocket;
static Socket socket;
static DataInputStream dis;
static DataOutputStream dout;
    /**
     * Creates new form Forum2
     */
    public Forum2() {
        initComponents();
    }
```

Gambar 2.b.8

```
@SuppressWarnings("unchecked")
Generated Code

public void startServer(int port) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            ServerSocket serverSocket = null;
            try {
                String messageIn = "";
                serverSocket = new ServerSocket(port);
                socket = serverSocket.accept();

                if (socket.isConnected()) {
                    jTextArea.append("Successfully Connected\n");
                }
                dis = new DataInputStream(socket.getInputStream());
                dout = new DataOutputStream(socket.getOutputStream());
                while (!messageIn.equals("Out") && socket.isConnected()) {
                    try {
                        messageIn = dis.readUTF();
                        jTextArea.append("\nClient: " + messageIn);
                    } catch (IOException e) {
                        jTextArea.append("\nConnection Lost");
                        break;
                    }
                }
            } catch (Exception e) {
```

Gambar 2.b.9

```
                System.out.println("Error: " + e.getMessage());
                jTextArea.append("\nError: " + e.getMessage());
            } finally {
                try {
                    if (serverSocket != null) serverSocket.close();
                    if (dis != null) dis.close();
                    if (dout != null) dout.close();
                } catch (IOException e) {
                e.printStackTrace();
                }
            }
        }
    }).start();
}
private void StartServerActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int port = Integer.valueOf(jTextFieldPort.getText());
    startServer(port);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String message = jTextFieldMessage.getText();
    send(message);
}
public void send(String message){
    new Thread(new Runnable(){
        @Override
```

Gambar 2.b.10

```
            public void run(){
                try{
                    dout.writeUTF(message);
                    jTextFieldMessage.setText("");
                }catch (Exception e){
                    jTextArea.append(e.getMessage());
                }
            }
    }).start();
}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Forum2().setVisible(true);
        }
    });
}
```

Gambar 2.b.11

```
    // Variables declaration - do not modify
    private javax.swing.JButton StartServer;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea;
    private javax.swing.JTextField jTextFieldMessage;
    private javax.swing.JTextField jTextFieldPort;
    // End of variables declaration
}
```

Gambar 2.b.12

7. Kode yang selanjutnya digunakan untuk membuat kelas journal. Yang mana pada kelas ini akan dilakukan beberapa proses diantaranya untuk catatan bagi pengguna dengan tanggal, serta memberikan fitur untuk menambah, memperbarui, juga menghapus catatan pada tabel.

```
package MentalHealth;

import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author MSI THIN
 */
public class Journal extends javax.swing.JFrame {

    /**
     * Creates new form Journal
     */
    public Journal() {
        initComponents();
    }
```

Gambar 2.b.13

```
private void txtNoteActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Mainmenu mm2 = new Mainmenu();
    mm2.show();

    dispose();
}

private void AddbuttonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(txtNote.getText().equals("")||txtDate.getText().equals("")){
        JOptionPane.showMessageDialog(this, "Please Enter Text!");
    }else{
        String data[] ={txtNote.getText(),txtDate.getText()};

        DefaultTableModel tblModel = (DefaultTableModel)jTable1.getModel();
        tblModel.addRow(data);

        JOptionPane.showMessageDialog(this, "Data Successfully Added");

        txtNote.setText("");
        txtDate.setText("");
    }
}
```

Gambar 2.b.14

```
private void DeleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel tblModel = (DefaultTableModel)jTable1.getModel();
    if(jTable1.getSelectedRowCount()==1){
        tblModel.removeRow(jTable1.getSelectedRow());
    }else{
        if(jTable1.getRowCount()==0){
            //if table empty
            JOptionPane.showMessageDialog(this, "Table is Empty");
        }else{
            JOptionPane.showMessageDialog(this, "Please select Row for delete");
        }
    }
}

private void UpdateButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel tblModel = (DefaultTableModel)jTable1.getModel();
    if(jTable1.getSelectedRowCount()==1){
        String Note = txtNote.getText();
        String Date = txtDate.getText();
        tblModel.setValueAt(Note, jTable1.getSelectedRow(),0);
        tblModel.setValueAt(Date, jTable1.getSelectedRow(),1);

        JOptionPane.showMessageDialog(this, "Update Successfully");
```

Gambar 2.b.15

```
    }else{
        if(jTable1.getRowCount()==0){
            JOptionPane.showMessageDialog(this, "Table is Empty");
        }else{
            JOptionPane.showMessageDialog(this, "Please select Row");
        }
    }

}

private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel tblModel = (DefaultTableModel)jTable1.getModel();

    String tblNote = tblModel.getValueAt(jTable1.getSelectedRow(), 0).toString();
    String tblDate = tblModel.getValueAt(jTable1.getSelectedRow(), 1).toString();

    txtNote.setText(tblNote);
    txtDate.setText(tblDate);

}
```

Gambar 2.b.16

```
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
     Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Journal().setVisible(true);
        }
    });
}
```

Gambar 2.b.17

```
    // Variables declaration - do not modify
    private javax.swing.JButton Addbutton;
    private javax.swing.JButton DeleteButton;
    private javax.swing.JButton UpdateButton;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton7;
    private javax.swing.JButton jButton8;
    private javax.swing.JButton jButton9;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel10;
    private javax.swing.JPanel jPanel12;
    private javax.swing.JPanel jPanel9;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jTable1;
    private javax.swing.JTextField txtDate;
    private javax.swing.JTextField txtNote;
    // End of variables declaration
}
```

Gambar 2.b.18

8. Kemudian kode selanjutnya merupakan kode untuk membuat kelas berupa meditasi. Kelas ini akan mewujudkan fitur meditasi yang dapat memungkinkan pengguna memulai sesi meditasi atau kembali pada menu utama. Fitur ini dapat membantu pengguna dalam mengontrol emosi dan perasaan pengguna.

```
package MentalHealth;

/**
 *
 * @author MSI THIN
 */
public class Meditation extends javax.swing.JFrame {

    /**
     * Creates new form Meditation
     */
    public Meditation() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code
```

Gambar 2.b.19

```
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Mainmenu mm2 = new Mainmenu();
    mm2.show();

    dispose();
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Meditation2 meditation2 = new Meditation2();
    meditation2.show();

    dispose();
}
```

Gambar 2.b.20

```
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Meditation().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    private javax.swing.JButton jButton7;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel4;
    // End of variables declaration
}
```

Gambar 2.b.21

9. Pada kelas selanjutnya yang akan menjelaskan tampilan meditation yang lebih mendalam. Kelas ini adalah kelas meditation2. Yang mana akan menampilkan sesi meditasi yang mengajak pengguna untuk merilekskan tubuh dan pikiran. Dalam fitur ini juga terdapat tombol untuk kembali pada menu utama.

```
package MentalHealth;

/**
 *
 * @author MSI THIN
 */
public class Meditation2 extends javax.swing.JFrame {

    /**
     * Creates new form Meditation2
     */
    public Meditation2() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Mainmenu mm2 = new Mainmenu();
        mm2.show();
```

Gambar 2.b.22

Gambar 2.b.23

10. Kode yang selanjutnya menjelaskan tentang pembuatan kelas mood. Yang mana pada kelas ini, pengguna dapat menambahkan rasa apa yang sedang di alami pengguna pada hari ini. Dalam gambar kode di bawah dapat dilihat bahwa pencatatan mood oleh pengguna dapat dilakukan dengan fitur yang kompleks.

Gambar 2.b.24

```
private void txtMoodActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Mainmenu mm2 = new Mainmenu();
    mm2.show();

    dispose();
}

private void bttnAddActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(txtMood.getText().equals("")||txtDate.getText().equals("")){
        JOptionPane.showMessageDialog(this, "Please Enter Text!");
    }else{
        String data[] ={txtMood.getText(),txtDate.getText()};

        DefaultTableModel tblModel = (DefaultTableModel)jTable1.getModel();
        tblModel.addRow(data);

        JOptionPane.showMessageDialog(this, "Data Successfully Added");

        txtMood.setText("");
        txtDate.setText("");
    }
}
```

Gambar 2.b.25

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel tblModel = (DefaultTableModel)jTable1.getModel();

    if(jTable1.getSelectedRowCount()==1){
        tblModel.removeRow(jTable1.getSelectedRow());
    }else{
        if(jTable1.getRowCount()==0){
            //if table empty
            JOptionPane.showMessageDialog(this, "Table is Empty");
        }else{
            JOptionPane.showMessageDialog(this, "Please select Row for delete");
        }
    }
}

private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel tblModel = (DefaultTableModel)jTable1.getModel();

    String tblMood = tblModel.getValueAt(jTable1.getSelectedRow(), 0).toString();
    String tblDate = tblModel.getValueAt(jTable1.getSelectedRow(), 1).toString();

    txtMood.setText(tblMood);
    txtDate.setText(tblDate);

}
```

Gambar 2.b.26

```
private void bttnEditActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel tblModel = (DefaultTableModel)jTable1.getModel();
    if(jTable1.getSelectedRowCount()==1){
        String Mood = txtMood.getText();
        String Date = txtDate.getText();
        tblModel.setValueAt(Mood, jTable1.getSelectedRow(),0);
        tblModel.setValueAt(Date, jTable1.getSelectedRow(),1);

        JOptionPane.showMessageDialog(this, "Update Successfully");


    }else{
        if(jTable1.getRowCount()==0){
            JOptionPane.showMessageDialog(this, "Table is Empty");
        }else{
            JOptionPane.showMessageDialog(this, "Please select Row");
        }
    }
}
```

Gambar 2.b.27

```
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
     Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Mood().setVisible(true);
        }
    });
}
```

Gambar 2.b.28

```
    // Variables declaration - do not modify
    private javax.swing.JButton bttnAdd;
    private javax.swing.JButton bttnEdit;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton7;
    private javax.swing.JButton jButton8;
    private javax.swing.JButton jButton9;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel10;
    private javax.swing.JPanel jPanel12;
    private javax.swing.JPanel jPanel9;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTable jTable1;
    private javax.swing.JTable jTable2;
    private javax.swing.JTextField txtDate;
    private javax.swing.JTextField txtMood;
    // End of variables declaration
}
```

Gambar 2.b.29

11. Kode selanjutnya untuk membuat kelas splash yang berfungsi untuk menampilkan tampilan awal untuk aplikasi yang dibuat.

Gambar 2.b.30

Gambar 2.b.31

12. Kemudian kelas selanjutnya adalah kelas main menu. Yang kelas ini digunakan sebagai blue print dari kode aplikasi ini. Semua kode akan berawal pada kelas ini. Jadi kelas ini yang nantinya akan dijadikan sebagai titik umpu dari program. Pada kelas ini pula semua fitu-fitur dapat dipanggil.

```
package MentalHealth;

/**
 *
 * @author MSI THIN
 */
public class Mainmenu extends javax.swing.JFrame {

    /**
     * Creates new form Mainmenu
     */
    public Mainmenu() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code
```

Gambar 2.b.32

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Mood mood = new Mood();
    mood.show();

    dispose();
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Journal journal = new Journal();
    journal.show();

    dispose();
}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Meditation meditation = new Meditation();
    meditation.show();

    dispose();
}
```

Gambar 2.b.33

```
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Forum forum = new Forum();
    forum.show();

    dispose();
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Mainmenu().setVisible(true);
        }
    });
}
```

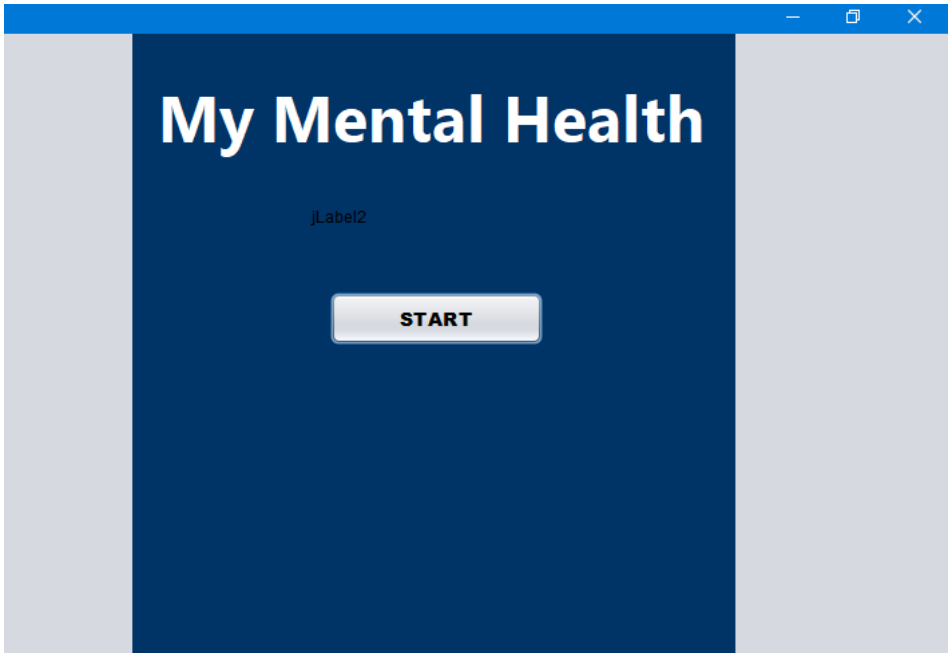
Gambar 2.b.34

```
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    private javax.swing.JButton jButton7;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    // End of variables declaration
}
```

Gambar 2.b.35

III. HASIL DAN PEMBAHASAN

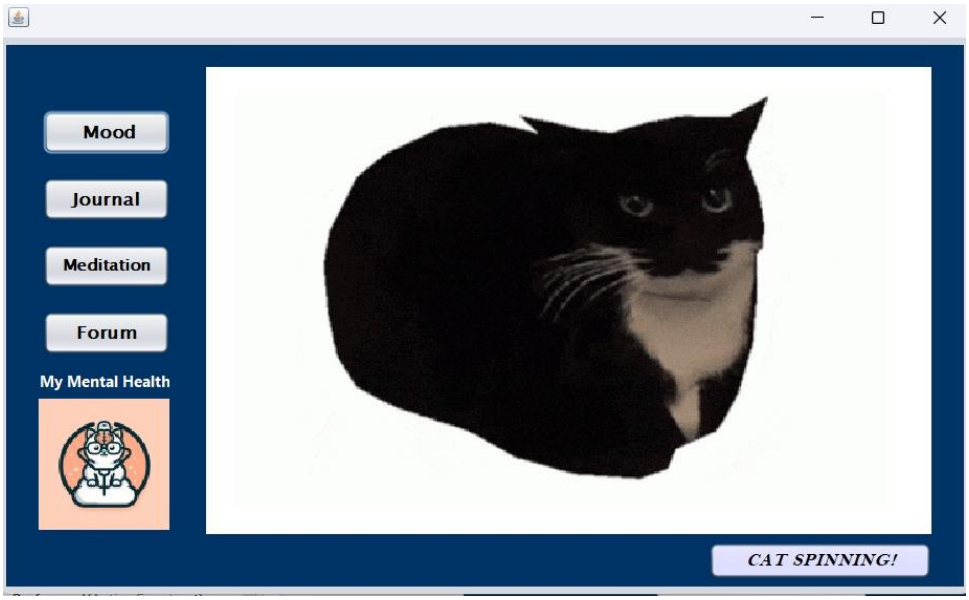
1. Tampilan Awal



Gambar 3.1

Gambar diatas ini menunjukkan tampilan awal dari aplikasi My Mental Health. Aplikasi ini dirancang untuk mendukung kesehatan mental pengguna dengan menyediakan berbagai fitur

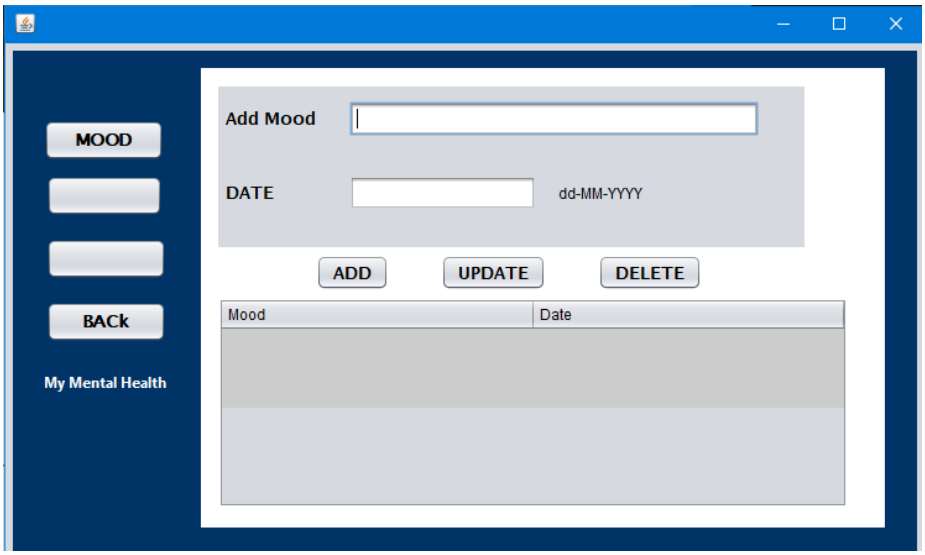
2. Tampilan Menu



Gambar 3.2

Dengan adanya gambar kucing, aplikasi ini mungkin juga berusaha menciptakan suasana yang lebih hangat dan menyenangkan, membantu pengguna merasa lebih baik.

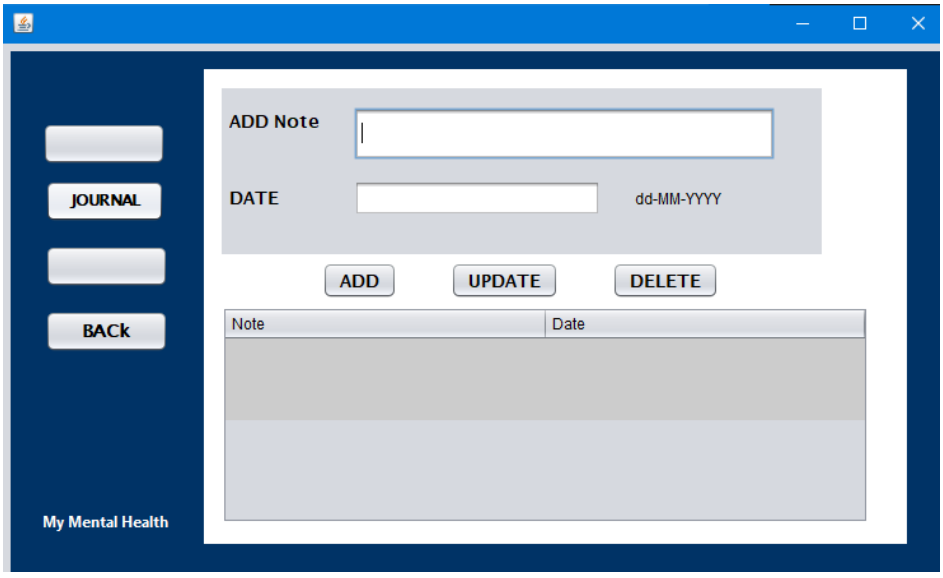
3. Tampilan Mood



Gambar 3.3

Pada gambar diatas ini menampilkan fitur mood dalam aplikasi My Mental Health. Fitur ini memungkinkan pengguna untuk melacak mood mereka dan melihat trennya dari waktu ke waktu. Ini dapat membantu pengguna untuk mengidentifikasi pola dalam mood mereka dan untuk mengetahui pemicu dari perubahan mood mereka.

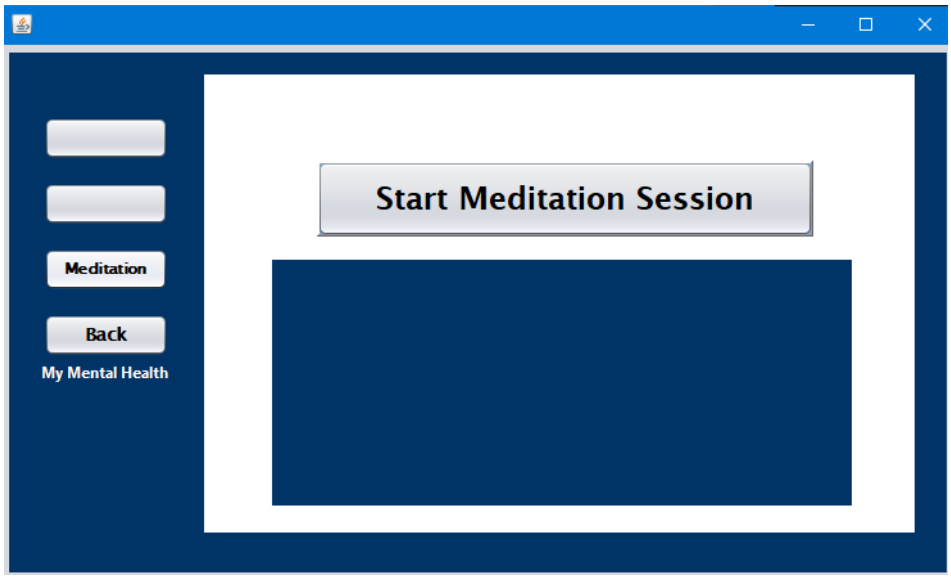
4. Tampilan Journal



Gambar 3.4

Gambar ini menampilkan fitur journal dari aplikasi my mental health. Fitur ini memungkinkan pengguna untuk menulis catatan dan pikiran mereka, yang dapat membantu pengguna untuk memproses emosi mereka dan untuk mengidentifikasi pola dalam pikiran mereka.

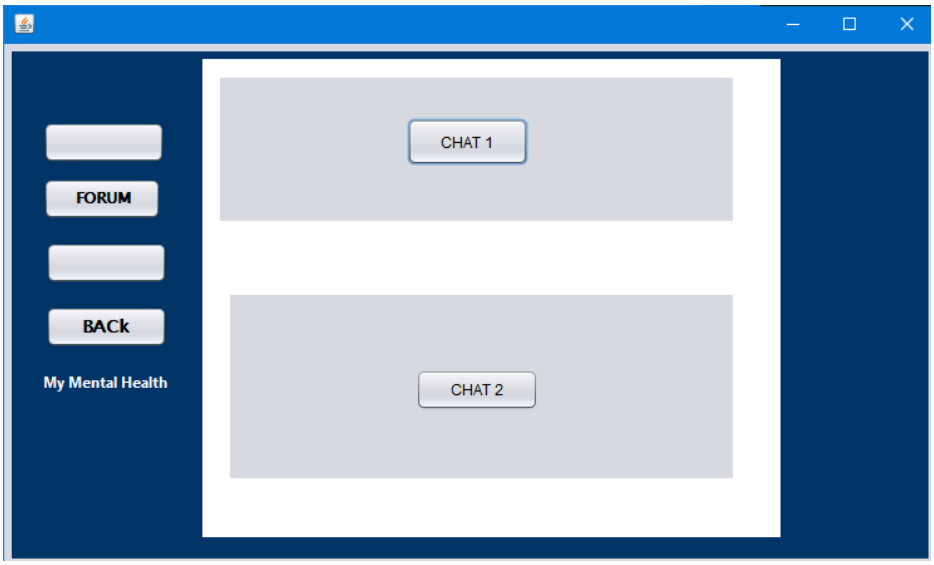
5. Tampilan Meditation



Gambar 3.5

Gambar diatas ini menampilkan meditation session dalam aplikasi My Mental Health. Fitur ini menawarkan sesi meditasi yang dipandu, yang dapat membantu pengguna untuk rileks dan untuk mengurangi tingkat stres.

6. Tampilan Forum



Gambar 3.6

Gambar diatas ini menampilkan fitur forum yang ada dalam aplikasi My Mental Health. Fitur ini memungkinkan pengguna untuk terhubung dengan orang lain yang memiliki masalah kesehatan mental yang sama. Ini dapat memberikan rasa dukungan dan komunitas bagi pengguna.

IV. KESIMPULAN

Pembuatan aplikasi My Mental Health menggunakan Apache NetBeans menerapkan prinsip Pemrograman Berorientasi Objek (PBO) secara efektif. Enkapsulasi digunakan untuk melindungi data sensitif pengguna, hanya mengizinkan akses melalui metode tertentu. Aplikasi ini dibangun dengan kelas-kelas yang memudahkan pengelolaan objek. Polimorfisme memungkinkan metode yang sama berfungsi berbeda berdasarkan objek yang memanggilnya, sementara pewarisan mengurangi duplikasi kode dengan membagikan atribut dan metode antar kelas. Kelas abstrak mendefinisikan kerangka dasar untuk fitur tertentu, dan interface memastikan interaksi yang konsisten antar komponen. Konsep komposisi dan agregasi membangun hubungan yang jelas antar objek, sedangkan file I/O dan serialisasi memungkinkan penyimpanan dan pemulihan data pengguna. Secara

keseluruhan, aplikasi ini memenuhi kebutuhan kesehatan mental pengguna dan juga menerapkan prinsip-prinsip pemrograman berorientasi objek dengan baik dalam pengembangan perangkat lunak.

Eka Makhsusan

- Membuat rancangan awal aplikasi
- Membuat diagram kelas
- Menyusun laporan

Dhiya Rahma A

- Membuat rancangan awal aplikasi
- Membuat materi PPT
- Menulis kode program aplikasi

Fatina Sabitul A

- Membuat rancangan awal aplikasi
- Menulis kode program aplikasi
- Membuat UI aplikasi

Nasywaa Jihaan L

- Membuat rancangan awal aplikasi
- Membuat materi PPT
- Menyusun laporan