

**LAPORAN TUGAS AKHIR
PEMROGRAMAN BERORIENTASI OBJEK**

KELOMPOK 9



Disusun Oleh
Ilham Saputra – 2210631250015

**PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SINGAPERBANGSA KARAWANG
2024**

KATA PENGANTAR

Alhamdulillah, saya bersyukur kepada Tuhan Yang Maha Esa yang telah memberikan kemudahan dan kesehatan kepada saya sehingga saya dapat menyelesaikan tugas laporan akhir pembuatan aplikasi perpustakaan untuk mata kuliah pemrograman berbasis objek.

Laporan ini telah saya susun untuk menyelesaikan tugas akhir mata kuliah pemrograman berbasis objek yang membutuhkan waktu dan tenaga yang cukup besar.

Saya sadar bahwa isi laporan ini masih jauh dari sempurna karena keterbatasan saya. Oleh karena itu, saya berharap adanya umpan balik berupa kritik dan saran yang membangun agar di kemudian hari saya dapat membuat laporan yang lebih baik lagi.

Semoga laporan dan aplikasi yang telah saya buat dapat berguna bagi dunia pendidikan.

Bekasi, 20 Mei 2024

Ilham Saputra

DAFTAR ISI

BAB I	4
PENDAHULUAN	4
A. Latar Belakang	4
B. Rumusan Masalah	4
C. Tujuan Masalah.....	4
BAB II.....	5
PERANCANGAN DIAGRAM UNIFIELD MODELING LANGUAGE	5
A. Class Diagram.....	5
B. Use Case Diagram.....	8
BAB III	12
IMPLEMENTASI KODE DAN PENGUJIAN	12
A. Koneksi Database.....	12
B. Form Login	13
C. Menu Utama.....	16
D. Menu Anggota.....	19
E. Halaman Data Anggota.....	24
F. Menu Buku.....	27
G. Halaman Data Buku.....	33
H. Menu Kategori	36
I. Menu Penerbit.....	42
J. Menu Petugas.....	48
K. Menu Peminjaman	54
L. Halaman Data Peminjaman.....	61
M. Menu Master	65
N. Menu Transaksi	68
O. Menu Laporan	70
P. Halaman Laporan Peminjaman.....	72
Q. Tampilan Halaman Laporan Peminjaman Menggunakan Jasper	77
BAB IV	78
KESIMPULAN	78
1. Tujuan Program:	78
2. Fitur Utama:	78
3. Konsep PBO yang Diterapkan:	78

BAB I

PENDAHULUAN

A. Latar Belakang

Universitas memiliki peran penting dalam menyediakan perpustakaan yang memudahkan mahasiswa untuk mengakses informasi yang dibutuhkan untuk mendukung kegiatan akademik mereka. Kemudahan akses ke perpustakaan sangat penting karena dapat membantu mahasiswa memenuhi kebutuhan akademik dengan lebih efisien. Dalam kehidupan akademik mahasiswa, perpustakaan menjadi ruang lingkup yang tak terbatas dalam menyediakan sumber daya pendidikan.

Dengan kemajuan teknologi saat ini, akses ke perpustakaan dapat ditingkatkan melalui pengembangan aplikasi Perpustakaan (PERPUSIKA). Aplikasi ini akan memungkinkan mahasiswa mengakses koleksi perpustakaan dengan mudah dan cepat. Pembuatan aplikasi perpustakaan merupakan langkah strategis untuk meningkatkan keefektifan sistem perpustakaan, memberikan kemudahan yang lebih besar bagi mahasiswa, serta memaksimalkan pemanfaatan sumber daya yang tersedia.

B. Rumusan Masalah

Sesuai yang dipaparkan Pada Latar Belakang dapat dijadikan Referensi dalam Membangun Aplikasi Perpustakaan. Berikut Poin – poin dapat diambil Menjadi Pedoman Membangun Aplikasi :

1. Siapa Saja yang Terlibat Pada System Aplikasi Perpustakaan ?
2. Bagaimana Pengimplementasian Teknologi dalam Perpustakaan Untuk Universitas ?
3. Bagaimana Sistem Perpustakaan Lebih Efisien untuk Mahasiswa ?

C. Tujuan Masalah

Berdasarkan Poin – poin dari Rumusan Masalah yang dipaparkan, Terdapat Tujuan dari Masalah yang ada Berikut Tujuannya :

1. Yang Terlibat dalam System ini adalah Admin Perpustakaan dan Mahasiswa
2. Pengimplementasian dilakukan dengan Membuat Aplikasi Perpustakaan (PERPUSIKA).
3. Dibuatkan Fitur pada Aplikasi Seperti Penambahan Anggota Perpustakaan, Penambahan Data Buku, Peminjaman Buku, dan Pengembalian Buku.

BAB II

PERANCANGAN DIAGRAM UNIFIELD MODELING LANGUAGE

A. Class Diagram

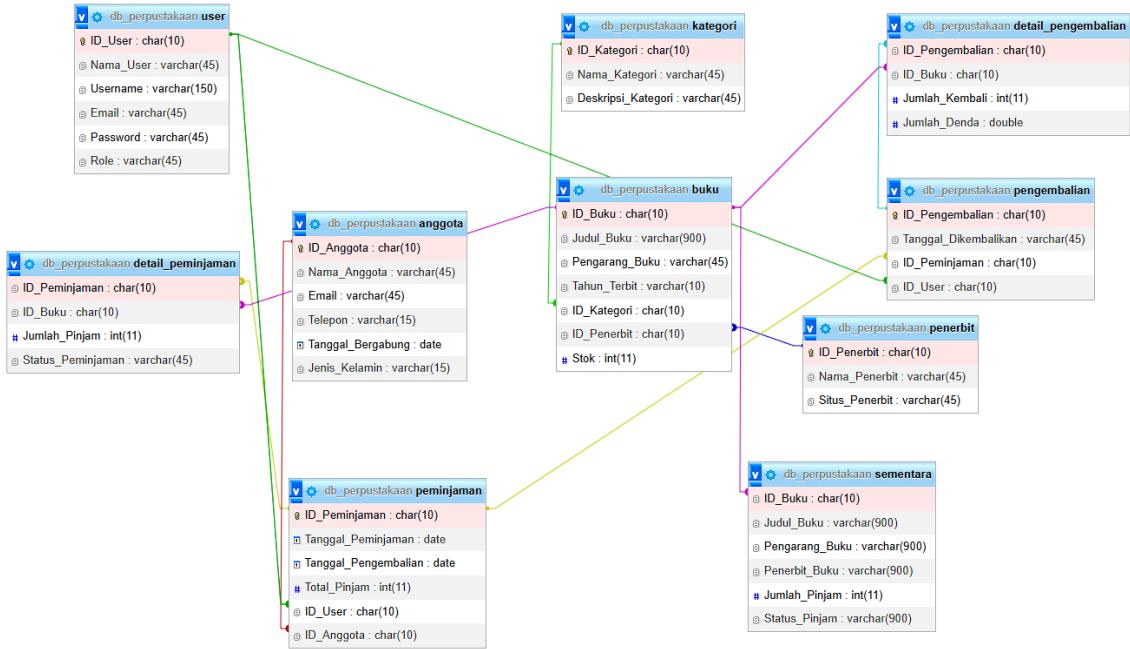


Diagram class ini menggambarkan sistem perpustakaan dengan detail lebih lanjut. Berikut adalah penjelasannya:

1. User: Merupakan tabel yang berisi data pengguna perpustakaan, yaitu data yang dimiliki oleh individu atau entitas yang memiliki akses untuk mengakses sistem perpustakaan. Tabel ini memiliki atribut-atribut sebagai berikut:
 - ID_User: primary key dari tabel User, berupa char(10).
 - Nama_User: atribut yang menyimpan nama lengkap dari pengguna, berupa varchar(45).
 - Username: atribut yang menyimpan username dari pengguna, berupa varchar(150).
 - Email: atribut yang menyimpan email dari pengguna, berupa varchar(45).
 - Password: atribut yang menyimpan password dari pengguna, berupa varchar(45).
 - Role: atribut yang menyimpan peran dari pengguna, berupa varchar(45).
2. Kategori: Merupakan tabel yang berisi data kategori buku, yaitu data yang digunakan untuk mengklasifikasikan buku sesuai dengan tema atau kategori yang relevan. Tabel ini memiliki atribut-atribut sebagai berikut:
 - ID_Kategori: primary key dari tabel Kategori, berupa char(10).
 - Nama_Kategori: atribut yang menyimpan nama kategori, berupa varchar(45).
 - Deskripsi_Kategori: atribut yang menyimpan deskripsi kategori, berupa varchar(45).

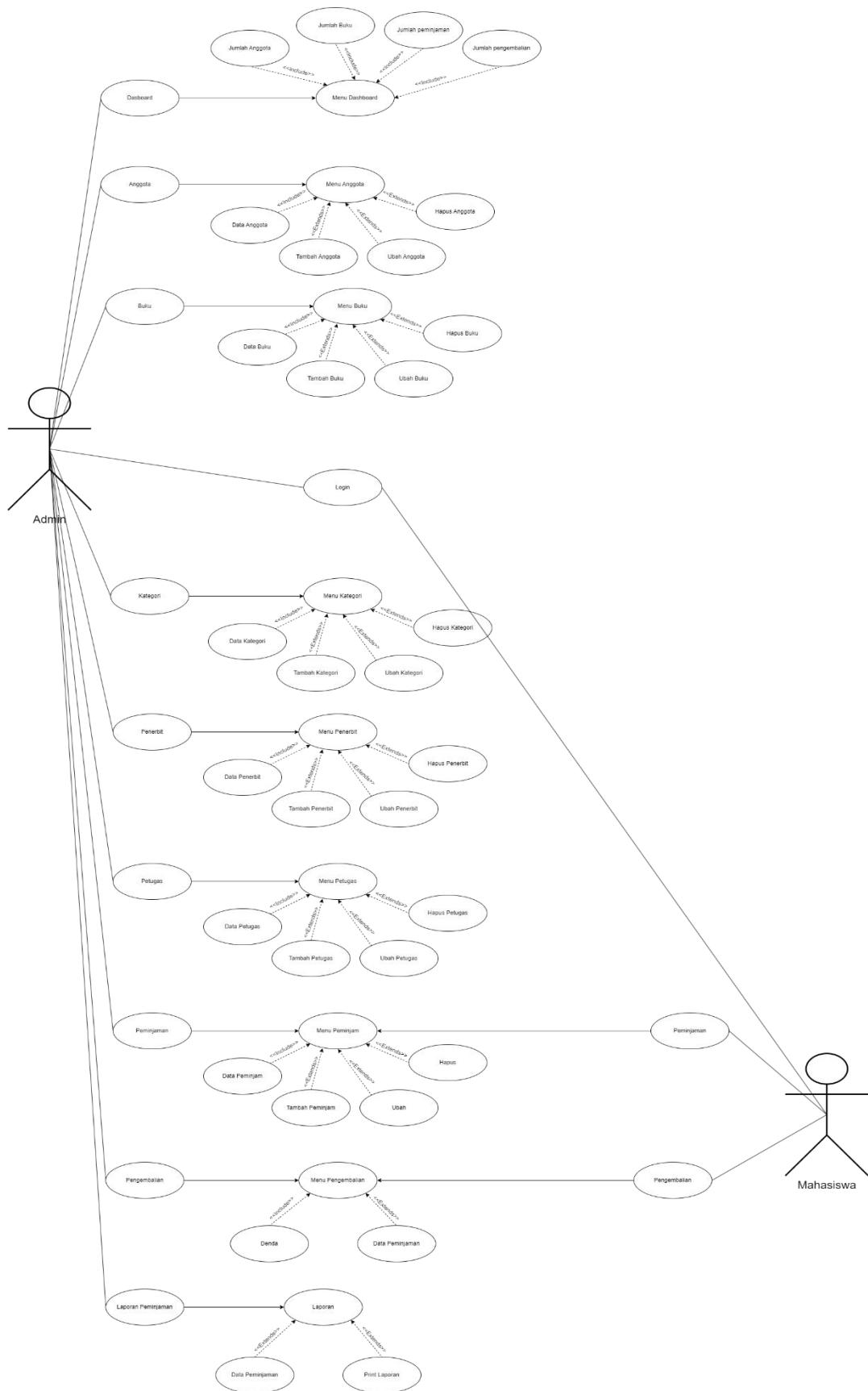
3. Detail Pengembalian: Merupakan tabel yang berisi data detail pengembalian buku, yaitu data yang menyimpan informasi tentang jumlah buku yang dikembalikan dan denda yang dikenakan. Tabel ini memiliki atribut-atribut sebagai berikut:
- ID_Pengembalian: primary key dari tabel Detail Pengembalian, berupa char(10).
 - ID_Buku: foreign key dari tabel Buku, berupa char(10).
 - Jumlah_Kembali: atribut yang menyimpan jumlah buku yang dikembalikan, berupa int(11).
 - Jumlah_Denda: atribut yang menyimpan denda yang dikenakan, berupa double.
4. Detail Peminjaman: Merupakan tabel yang berisi data detail peminjaman buku, yaitu data yang menyimpan informasi tentang jumlah buku yang dipinjam dan status peminjaman. Tabel ini memiliki atribut-atribut sebagai berikut:
- ID_Peminjaman: primary key dari tabel Detail Peminjaman, berupa char(10).
 - ID_Buku: foreign key dari tabel Buku, berupa char(10).
 - Jumlah_Pinjam: atribut yang menyimpan jumlah buku yang dipinjam, berupa int(11).
 - Status_Peminjaman: atribut yang menyimpan status peminjaman, berupa varchar(45).
5. Anggota: Merupakan tabel yang berisi data anggota perpustakaan, yaitu data yang dimiliki oleh individu atau entitas yang telah mendaftar dan menjadi anggota. Tabel ini memiliki atribut-atribut sebagai berikut:
- ID_Anggota: primary key dari tabel Anggota, berupa char(10).
 - Nama_Anggota: atribut yang menyimpan nama lengkap dari anggota, berupa varchar(45).
 - Alamat_Anggota: atribut yang menyimpan alamat dari anggota, berupa varchar(45).
 - No_Telp_Anggota: atribut yang menyimpan nomor telepon dari anggota, berupa varchar(45).
6. Buku: Merupakan tabel yang berisi data buku, yaitu data yang dimiliki oleh buku yang tersedia di perpustakaan. Tabel ini memiliki atribut-atribut sebagai berikut:
- ID_Buku: primary key dari tabel Buku, berupa char(10).
 - Judul_Buku: atribut yang menyimpan judul buku, berupa varchar(45).
 - Pengarang_Buku: atribut yang menyimpan pengarang buku, berupa varchar(45).
 - Penerbit_Buku: atribut yang menyimpan penerbit buku, berupa varchar(45).
 - Tahun_Terbit: atribut yang menyimpan tahun terbit buku, berupa int(11).
7. Pengembalian: Merupakan tabel yang berisi data pengembalian buku, yaitu data yang menyimpan informasi tentang tanggal pengembalian dan status pengembalian. Tabel ini memiliki atribut-atribut sebagai berikut:
- ID_Pengembalian: primary key dari tabel Pengembalian, berupa char(10).
 - ID_User: foreign key dari tabel User, berupa char(10).
 - Tanggal_Pengembalian: atribut yang menyimpan tanggal pengembalian, berupa date.
 - Status_Pengembalian: atribut yang menyimpan status pengembalian, berupa varchar(45).

8. Penerbit: Merupakan tabel yang berisi data penerbit buku, yaitu data yang dimiliki oleh penerbit buku yang tersedia di perpustakaan. Tabel ini memiliki atribut-atribut sebagai berikut:
 - ID_Penerbit: primary key dari tabel Penerbit, berupa char(10).
 - Nama_Penerbit: atribut yang menyimpan nama penerbit, berupa varchar(45).
 - Alamat_Penerbit: atribut yang menyimpan alamat penerbit, berupa varchar(45).
9. Peminjaman: Merupakan tabel yang berisi data peminjaman buku, yaitu data yang menyimpan informasi tentang tanggal peminjaman dan status peminjaman. Tabel ini memiliki atribut-atribut sebagai berikut:
 - ID_Peminjaman: primary key dari tabel Peminjaman, berupa char(10).
 - ID_User: foreign key dari tabel User, berupa char(10).
 - Tanggal_Peminjaman: atribut yang menyimpan tanggal peminjaman, berupa date.
 - Status_Peminjaman: atribut yang menyimpan status peminjaman, berupa varchar(45).
10. Sementara: Merupakan tabel sementara yang menyimpan data peminjaman buku, yaitu data yang digunakan untuk mengelola peminjaman buku. Tabel ini memiliki atribut-atribut sebagai berikut:
 - ID_Sementara: primary key dari tabel Sementara, berupa char(10).
 - ID_Buku: foreign key dari tabel Buku, berupa char(10).
 - ID_User: foreign key dari tabel User, berupa char(10).
 - Tanggal_Peminjaman: atribut yang menyimpan tanggal peminjaman, berupa date.

Hubungan antar Tabel:

- User memiliki hubungan dengan Detail Pengembalian dan Pengembalian melalui ID_User.
- User memiliki hubungan dengan Peminjaman melalui ID_User.
- Kategori memiliki hubungan dengan Buku melalui ID_Kategori.
- Detail Pengembalian memiliki hubungan dengan Buku melalui ID_Buku.
- Detail Peminjaman memiliki hubungan dengan Buku melalui ID_Buku.
- Detail Peminjaman memiliki hubungan dengan Anggota melalui ID_Anggota.
- Detail Peminjaman memiliki hubungan dengan Peminjaman melalui ID_Peminjaman.
- Anggota memiliki hubungan dengan Peminjaman melalui ID_Anggota.
- Buku memiliki hubungan dengan Penerbit melalui Pengembalian memiliki hubungan dengan Detail Pengembalian melalui ID_Pengembalian.
- Pengembalian memiliki hubungan dengan Peminjaman melalui ID_Peminjaman.
- Sementara memiliki hubungan dengan Buku melalui ID_Buku.
- Sementara memiliki hubungan dengan User melalui ID_User.
- Sementara memiliki hubungan dengan Peminjaman melalui ID_Peminjaman.

B. Use Case Diagram



Use Case Diagram ini menggambarkan sistem perpustakaan dengan berbagai fungsi dan entitas yang terlibat di dalamnya. Berikut penjelasan detailnya:

Entitas:

- Anggota: Entitas ini mewakili pengguna perpustakaan, yang dapat meminjam buku.
- Buku: Entitas ini mewakili koleksi buku yang tersedia di perpustakaan.
- Kategori: Entitas ini mengelompokkan buku berdasarkan jenisnya (misalnya, fiksi, non-fiksi, sejarah, dll.)
- Penerbit: Entitas ini mewakili perusahaan yang menerbitkan buku.
- Petugas: Entitas ini mewakili staf perpustakaan yang mengelola sistem.
- Peminjaman: Entitas ini merepresentasikan transaksi peminjaman buku oleh anggota.
- Pengembalian: Entitas ini merepresentasikan transaksi pengembalian buku oleh anggota.

Use Cases:

- Jumlah Buku: Use Case ini menampilkan jumlah total buku yang tersedia di perpustakaan.
- Jumlah Anggota: Use Case ini menampilkan jumlah total anggota perpustakaan.
- Jumlah Peminjaman: Use Case ini menampilkan jumlah total transaksi peminjaman buku.
- Jumlah Pengembalian: Use Case ini menampilkan jumlah total transaksi pengembalian buku.

Fungsi Utama:

- Dasboard: Use Case ini menampilkan informasi dasar tentang sistem perpustakaan, seperti jumlah buku, anggota, peminjaman, dan pengembalian.
- Data Anggota: Use Case ini memungkinkan petugas untuk mengakses dan mengelola data anggota, termasuk menambahkan, mengubah, dan menghapus data anggota.
- Data Buku: Use Case ini memungkinkan petugas untuk mengakses dan mengelola data buku, termasuk menambahkan, mengubah, dan menghapus data buku.
- Data Kategori: Use Case ini memungkinkan petugas untuk mengakses dan mengelola data kategori buku, termasuk menambahkan, mengubah, dan menghapus data kategori.
- Data Penerbit: Use Case ini memungkinkan petugas untuk mengakses dan mengelola data penerbit buku, termasuk menambahkan, mengubah, dan menghapus data penerbit.
- Data Petugas: Use Case ini memungkinkan petugas untuk mengakses dan mengelola data petugas, termasuk menambahkan, mengubah, dan menghapus data petugas.
- Data Peminjam: Use Case ini memungkinkan petugas untuk mengakses dan mengelola data peminjaman buku, termasuk menambahkan, mengubah, dan menghapus data peminjaman.
- Menu Dashboard: Use Case ini menampilkan berbagai menu utama yang tersedia di sistem perpustakaan.
- Menu Anggota: Use Case ini menampilkan menu khusus untuk mengelola data anggota.
- Menu Buku: Use Case ini menampilkan menu khusus untuk mengelola data buku.

- Menu Kategori: Use Case ini menampilkan menu khusus untuk mengelola data kategori buku.
- Menu Penerbit: Use Case ini menampilkan menu khusus untuk mengelola data penerbit buku.
- Menu Petugas: Use Case ini menampilkan menu khusus untuk mengelola data petugas.
- Menu Peminjam: Use Case ini menampilkan menu khusus untuk mengelola data peminjaman buku.
- Menu Pengembalian: Use Case ini menampilkan menu khusus untuk mengelola data pengembalian buku.
- Tambah Anggota: Use Case ini memungkinkan petugas untuk menambahkan data anggota baru ke sistem.
- Ubah Anggota: Use Case ini memungkinkan petugas untuk mengubah data anggota yang sudah ada di sistem.
- Hapus Anggota: Use Case ini memungkinkan petugas untuk menghapus data anggota dari sistem.
- Tambah Buku: Use Case ini memungkinkan petugas untuk menambahkan data buku baru ke sistem.
- Ubah Buku: Use Case ini memungkinkan petugas untuk mengubah data buku yang sudah ada di sistem.
- Hapus Buku: Use Case ini memungkinkan petugas untuk menghapus data buku dari sistem.
- Tambah Kategori: Use Case ini memungkinkan petugas untuk menambahkan kategori buku baru ke sistem.
- Ubah Kategori: Use Case ini memungkinkan petugas untuk mengubah kategori buku yang sudah ada di sistem.
- Hapus Kategori: Use Case ini memungkinkan petugas untuk menghapus kategori buku dari sistem.
- Tambah Penerbit: Use Case ini memungkinkan petugas untuk menambahkan data penerbit baru ke sistem.
- Ubah Penerbit: Use Case ini memungkinkan petugas untuk mengubah data penerbit yang sudah ada di sistem.
- Hapus Penerbit: Use Case ini memungkinkan petugas untuk menghapus data penerbit dari sistem.
- Tambah Petugas: Use Case ini memungkinkan petugas untuk menambahkan data petugas baru ke sistem.
- Ubah Petugas: Use Case ini memungkinkan petugas untuk mengubah data petugas yang sudah ada di sistem.
- Hapus Petugas: Use Case ini memungkinkan petugas untuk menghapus data petugas dari sistem.
- Tambah Peminjam: Use Case ini memungkinkan petugas untuk mencatat transaksi peminjaman buku baru.
- Ubah: Use Case ini memungkinkan petugas untuk mengubah data peminjaman buku yang sudah ada.
- Hapus: Use Case ini memungkinkan petugas untuk menghapus data peminjaman buku dari sistem.
- Denda: Use Case ini memungkinkan petugas untuk mencatat denda bagi anggota yang terlambat mengembalikan buku.

- Laporan: Use Case ini memungkinkan petugas untuk melihat berbagai laporan terkait dengan data perpustakaan.
- Data Peminjaman: Use Case ini memungkinkan petugas untuk mengakses data peminjaman buku.
- Print Laporan: Use Case ini memungkinkan petugas untuk mencetak laporan yang dibutuhkan.

BAB III

IMPLEMENTASI KODE DAN PENGUJIAN

A. Koneksi Database

```
● ● ●
package Koneksi;

import java.sql.Connection;
import java.sql.DriverManager;
import java.util.logging.Logger;
import java.util.logging.Level;

public class Koneksi {
    private static Connection conn;

    public static Connection getConnection(){
        if(conn == null){
            try{
                String url = "jdbc:mysql://localhost:3306/db_perpustakaan";
                String user = "root";
                String pass = "";
                Class.forName("com.mysql.cj.jdbc.Driver");
                conn = DriverManager.getConnection(url, user, pass);

                if (conn != null) {
                    System.out.println("Koneksi ke database berhasil.");
                }
                else {
                    System.out.println("Gagal melakukan koneksi ke database.");
                }
            } catch(Exception e){
                Logger.getLogger(Koneksi.class.getName()).log(Level.SEVERE, null, e);
                System.out.println("Gagal melakukan koneksi ke database: " +
e.getMessage());
            }
        }
        return conn;
    }
}
```

1. Enkapsulasi:

- Variabel conn dideklarasikan sebagai private static, sehingga hanya dapat diakses dalam kelas Koneksi.

2. Exception Handling:

- Penggunaan try-catch untuk menangani kemungkinan kesalahan saat koneksi ke database, seperti kesalahan kelas tidak ditemukan atau koneksi ke database gagal.
- Log dari Logger dicetak jika terjadi pengecualian selama proses koneksi.

3. Fungsi Setiap Method:

- `getConnection()`: Method statis yang digunakan untuk mendapatkan objek koneksi ke database. Jika objek koneksi belum ada, method ini akan membuat koneksi baru menggunakan URL, username, dan password yang telah ditentukan. Method ini juga menangani pengecualian yang mungkin terjadi selama proses koneksi.

B. Form Login

Tampilan :

The screenshot shows a login interface for a library system. On the left, there is a white sidebar with the text "Selamat Datang" at the top, followed by input fields for "USERNAME" and "PASSWORD". Below these is a red "LOGIN" button. On the right, the main content area has a red background. At the top right is a logo consisting of a yellow torch-like emblem above the text "UNSUNG" in blue and green. Below the logo, the text "Perpustakaan" is in blue, and "Universitas Singaperbangsa Karawang" is in white. The bottom half of the screen features a stylized illustration of books arranged on shelves, depicted in various colors like brown, blue, and red.

Kode :

```
package Kode;

import Koneksi.Koneksi;
import Main.MenuUtama;
import Main.MenuUtama;
import com.formdev.flatlaf.FlatLightLaf;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.awt.Color;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.HashMap;
import java.util.Map;
import static java.util.Objects.hash;
import java.util.logging.Logger;
import java.util.logging.Level;
import javax.swing.JOptionPane;
import javax.swing.UIManager;
```

public class FormLogin extends javax.swing.JFrame {

```
    int xx, yy;
    private Connection conn;
```

public FormLogin() {
 initComponents();
 conn = Koneksi.getConnection();
 setBackground(new Color(0, 0, 0, 0));
 setActionButton();
 }
}

```
private void formMousePressed(java.awt.event.MouseEvent evt) {
    xx = evt.getX();
    yy = evt.getY();
}

private void formMouseDragged(java.awt.event.MouseEvent evt) {
    int x = evt.getScreenX();
    int y = evt.getScreenY();
    this.setLocation(x - xx, y - yy);
}

public static void main(String args[]) {
    FlatLightLaf.setup();
    UIManager.put("component.arc", 15);

    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new FormLogin().setVisible(true);
        }
    });
}
```

```
private void setActionButton() {
    lb_eye.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            lb_eye.setVisible(false);
            lb_hideeye.setVisible(true);
            jt_password.setEchoChar((char)0);
        }
    });
    lb_hideeye.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            lb_eye.setVisible(true);
            lb_hideeye.setVisible(false);
            jt_password.setEchoChar('*');
        }
    });
    btn_login.addMouseListener(new MouseAdapter(){
        @Override
        public void mouseClicked(MouseEvent e){
            prosesLogin();
        }
    });
    lb_exit.addMouseListener(new MouseAdapter(){
        @Override
        public void mouseClicked(MouseEvent e){
            dispose();
            System.exit(0);
        }
    });
    jt_password.addKeyListener(new KeyAdapter(){
        @Override
        public void keyPressed(KeyEvent e){
            if(e.getKeyCode() == KeyEvent.VK_ENTER){
                btn_login.doClick();
            }
        }
    });
}
```

```
public String getMd5java(String message){
    String digest = null;
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] hash = md.digest(message.getBytes("UTF-8"));

        StringBuilder sb = new StringBuilder(2 * hash.length);
        for (byte b : hash) {
            sb.append(String.format("%02x", b & 0xff));
        }
        digest = sb.toString();
    } catch (Exception e) {
        Logger.getLogger(FormLogin.class.getName()).log(Level.SEVERE, null, e);
    }
    return digest;
}

private boolean validasiInput(){
    boolean valid = false;
    if(jt_username.getText().trim().isEmpty()){
        JOptionPane.showMessageDialog(this, "Username Tidak Boleh Kosong");
    }
    else if(new String(jt_password.getPassword()).trim().isEmpty()){
        JOptionPane.showMessageDialog(this, "Password Tidak Boleh Kosong");
    }
    else{
        valid = true;
    }
    return valid;
}

private Map<String, String> checkLogin(String Username, String Password) {
    Map<String, String> result = new HashMap<>();
    try {
        if (conn != null) {
            String sql = "SELECT ID_User, Nama_User, Role FROM user WHERE Username = ? AND Password = ?";
            try (PreparedStatement st = conn.prepareStatement(sql)) {
                st.setString(1, Username);
                st.setString(2, Password);

                try (ResultSet rs = st.executeQuery()) {
                    if (rs.next()) {
                        result.put("ID_User", rs.getString("ID_User"));
                        result.put("Nama_User", rs.getString("Nama_User"));
                        result.put("Role", rs.getString("Role"));
                    }
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return null;
    }
```

```
private void prosesLogin(){
    if (validasiInput()) {
        String username = jt_username.getText();
        char[] passwordChars = jt_password.getPassword();
        String password = new String(passwordChars);
        String hashedPassword = getMd5java(password);

        Map<String, String> loginResult = checkLogin(username, hashedPassword);

        if (loginResult != null) {
            String userID = loginResult.get("ID_User");
            String namaUser = loginResult.get("Nama_User");
            String roleUser = loginResult.get("Role");

            MenuUtama mn = new MenuUtama(userID, namaUser, roleUser);
            mn.setVisible(true);
            mn.revalidate();
            dispose();
        } else {
            JOptionPane.showMessageDialog(this, "Username atau Password Salah", "Pesan", JOptionPane.INFORMATION_MESSAGE);
        }
    }
}
```

Penjelasan :

1. Enkapsulasi:

- Penggunaan class dengan modifier private untuk menyembunyikan detail implementasi dari luar class.
- Penggunaan getter dan setter dalam class property.JtextfieldCustom dan property.JpasswordfieldCustom untuk mengakses dan mengubah nilai dari variabel private.

2. Event-Driven:

- Mendengarkan event mouse click pada tombol login (btn_login), tombol untuk menyembunyikan dan menampilkan password (lb_eye dan lb_hideeye), serta tombol exit (lb_exit) dengan menggunakan MouseListener dan ActionListener.
- Mendengarkan event key press pada field password (jt_password) dengan menggunakan KeyListener.

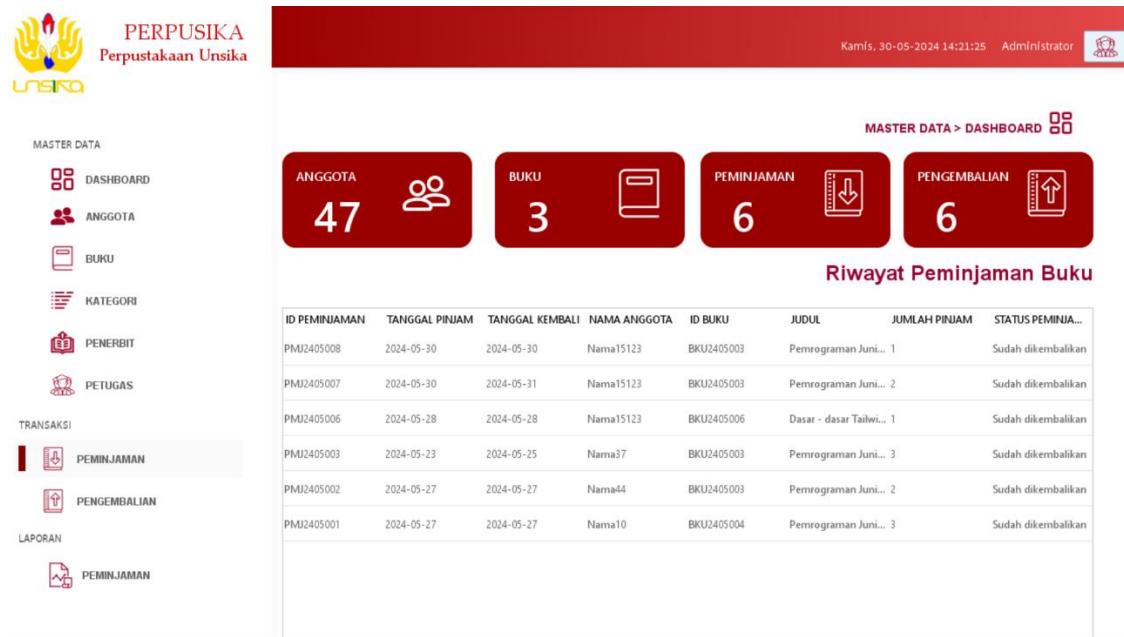
3. Fungsi Setiap Method:

- initComponents(): Menginisialisasi komponen GUI seperti tombol, label, dan field.
- setActionButton(): Mengatur aksi yang terjadi saat tombol dan label ditekan atau diklik.
- getMd5java(String message): Menghasilkan hash MD5 dari string yang diberikan.
- validasiInput(): Memeriksa apakah input username dan password sudah diisi atau tidak.
- checkLogin(String Username, String Password): Memeriksa kredensial login di database. Jika benar, mengembalikan informasi pengguna.
- prosesLogin(): Memproses login dengan memeriksa validasi input, melakukan hashing password, memeriksa kredensial login, dan membuka menu utama jika login berhasil.

C. Menu Utama

Tampilan :

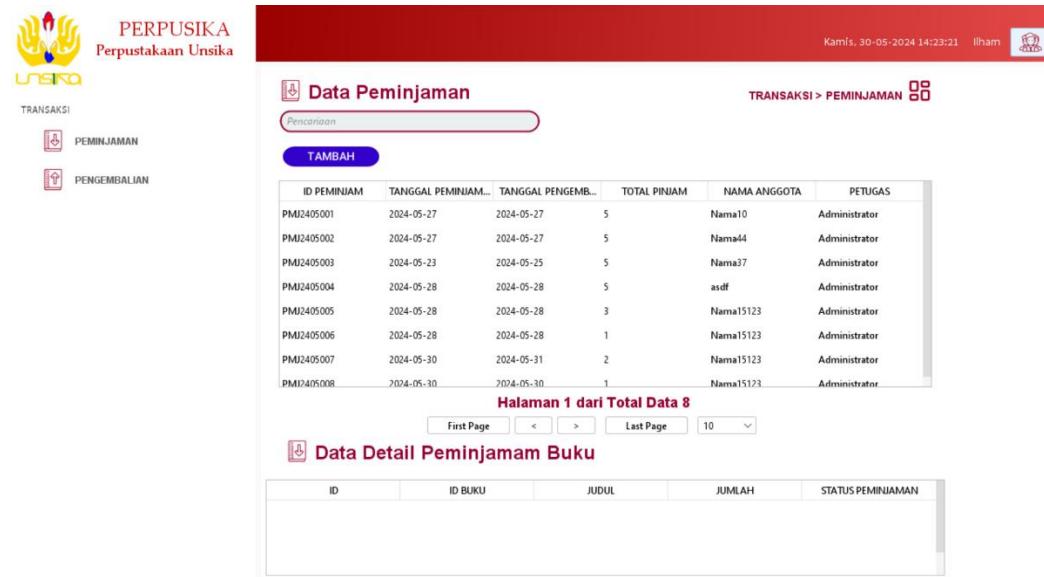
1) Tampilan Menu Utama Admin



The screenshot shows the PERPUSIKA Admin Dashboard. On the left, there's a sidebar with categories like MASTER DATA, TRANSAKSI, and LAPORAN. The TRANSAKSI section has links for PEMINJAMAN and PENGEMBALIAN, both of which are highlighted in red. The main area displays four summary boxes: ANGGOTA (47), BUKU (3), PEMINJAMAN (6), and PENGEMBALIAN (6). Below these is a section titled "Riwayat Peminjaman Buku" (History of Book Borrowing) with a table of borrowing records:

ID PEMINJAMAN	TANGGAL PINJAM	TANGGAL KEMBALI	NAMA ANGGOTA	ID BUKU	JUDUL	JUMLAH PINJAM	STATUS PEMINJA...
PMJ2405008	2024-05-30	2024-05-30	Nama15123	BKU2405003	Pemrograman Juni...	1	Sudah dikembalikan
PMJ2405007	2024-05-30	2024-05-31	Nama15123	BKU2405003	Pemrograman Juni...	2	Sudah dikembalikan
PMJ2405006	2024-05-28	2024-05-28	Nama15123	BKU2405006	Dasar - dasar Tailwi...	1	Sudah dikembalikan
PMJ2405003	2024-05-23	2024-05-25	Nama37	BKU2405003	Pemrograman Juni...	3	Sudah dikembalikan
PMJ2405002	2024-05-27	2024-05-27	Nama44	BKU2405003	Pemrograman Juni...	2	Sudah dikembalikan
PMJ2405001	2024-05-27	2024-05-27	Nama10	BKU2405004	Pemrograman Juni...	3	Sudah dikembalikan

2) Tampilan Menu Utama User



The screenshot shows the PERPUSIKA User Dashboard. The sidebar includes links for PEMINJAMAN and PENGEMBALIAN, both of which are highlighted in red. The main area features a "Data Peminjaman" section with a table of borrowing records:

ID PEMINJAM	TANGGAL PEMINJAM...	TANGGAL PENGEMB...	TOTAL PINJAM	NAMA ANGGOTA	PETUGAS
PMJ2405001	2024-05-27	2024-05-27	5	Nama10	Administrator
PMJ2405002	2024-05-27	2024-05-27	5	Nama44	Administrator
PMJ2405003	2024-05-23	2024-05-25	5	Nama37	Administrator
PMJ2405004	2024-05-28	2024-05-28	5	asdf	Administrator
PMJ2405005	2024-05-28	2024-05-28	3	Nama15123	Administrator
PMJ2405006	2024-05-28	2024-05-28	1	Nama15123	Administrator
PMJ2405007	2024-05-30	2024-05-31	2	Nama15123	Administrator
PMJ2405008	2024-05-30	2024-05-30	1	Nama15123	Administrator

Below this is a "Data Detail Peminjamam Buku" section with a table:

ID	ID BUKU	JUDUL	JUMLAH	STATUS PEMINJAMAN

Kode :

```
● ● ●
package Main;

import Kode.MenuAnggota;
import Kode.MenuBuku;
import java.awt.Color;
import Kode.MenuDashboard;
import Kode.MenuKategori;
import Kode.MenuPeminjaman;
import Kode.MenuPenerbit;
import Kode.MenuPengembalian;
import Kode.MenuPetugas;
import Kode.MenuProfile;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.Locale;
import javax.swing.JPanel;
import javax.swing.Timer;

public class MenuUtama extends javax.swing.JFrame {

    int xx, xy;

    private String userID;
    private String namaUser;
    private String roleUser;
    private Timer timer;

    public MenuUtama(String userID, String namaUser, String roleUser) {
        initComponents();
        this.userID = userID;
        this.roleUser = roleUser;
        lb_profil.setText(namaUser);
        addMenu();
        setDate();
    }

    public MenuUtama() {
        initComponents();
    }

    public String getUserID() {
        return userID;
    }

    public String getRoleUser() {
        return roleUser;
    }

    public JPanel getPanelUtama() {
        return jpn_utama;
    }

    private void addMenu(){
        if(roleUser.equals("Admin")){
            sideMenu.add(new MenuMaster(this));
            sideMenu.add(new MenuTransaksi(this));
            sideMenu.add(new MenuLaporan(this));
        }
        else{
            sideMenu.add(new MenuTransaksi(this));
        }
    }

    private void setDate() {
        timer = new Timer(1000, new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Calendar calendar = Calendar.getInstance();
                Date now = calendar.getTime();
                SimpleDateFormat formatHari = new SimpleDateFormat("EEEE", new Locale("in", "ID"));
                SimpleDateFormat formatTanggal = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");

                String hari = formatHari.format(now);
                String tanggal = formatTanggal.format(now);

                lb_date.setText(hari + ", " + tanggal);
            }
        });
        timer.start();
    }
}
```

```
● ● ●
private void formMousePressed(java.awt.event.MouseEvent evt) {
    xx = evt.getX();
    xy = evt.getY();
}

private void formMouseDragged(java.awt.event.MouseEvent evt) {
    int x = evt.getXOnScreen();
    int y = evt.getYOnScreen();
    this.setLocation(x - xx, y - xy);
}

private void jbtn_profileActionPerformed(java.awt.event.ActionEvent evt) {

    MenuProfile menu = new MenuProfile(this, true, this);
    Point p = jbtn_profile.getLocationOnScreen();
    int x = p.x + jbtn_profile.getWidth() - menu.getWidth();
    int y = p.y + jbtn_profile.getHeight();
    menu.setLocation(x, y);
    menu.setVisible(true);
}

private void formWindowOpened(java.awt.event.WindowEvent evt) {
    jpn_utama.removeAll();
    jpn_utama.add(new MenuDashboard());
    jpn_utama.repaint();
    jpn_utama.revalidate();
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            String userID = "ID";
            String namaUser = "Nama";
            String roleUser = "Role";

            new MenuUtama(userID, namaUser, roleUser).setVisible(true);
        }
    });
}
```

Penjelasan :

1. Enkapsulasi:

- Penggunaan variabel userID, namaUser, dan roleUser sebagai variabel instance private, yang hanya dapat diakses melalui metode getter (getUserID(), getRoleUser()).

2. Event-Driven:

- Mendengarkan event mouse pressed dan mouse dragged pada frame (formMousePressed() dan formMouseDragged()) untuk mengizinkan pengguna untuk menggeser frame dengan menahan dan menarik mouse.
- Mendengarkan event button action pada tombol profil (jbtn_profileActionPerformed()), yang memunculkan menu profil saat tombol profil ditekan.

3. Fungsi Setiap Method:

- addMenu(): Menambahkan menu berdasarkan peran pengguna (admin atau bukan admin).
- setDate(): Mengatur timer untuk menampilkan tanggal dan waktu saat ini secara real-time pada label lb_date.
- formMousePressed(): Menangani event ketika mouse ditekan pada frame untuk memperbarui posisi xx dan xy.
- formMouseDragged(): Menangani event ketika mouse di-drag pada frame untuk menggeser frame sesuai dengan perubahan posisi mouse.
- jbtn_profileActionPerformed(): Menangani event ketika tombol profil ditekan untuk menampilkan menu profil pada posisi yang sesuai.
- formWindowOpened(): Menangani event ketika jendela dibuka untuk menampilkan menu dashboard sebagai menu utama secara default.

D. Menu Anggota

Tampilan :

1) Tampilan Data Anggota

Data Anggota Perpustakaan

Pencarian

TAMBAH HAPUS BATAL

MASTER DATA > ANGGOTA 00

Halaman 1 of Total Halaman

First Page < > Last Page 10

2) Tampilan Tambah Data Anggota

Tambah Data Anggota Perpustakaan

SIMPAN BATAL

MASTER DATA > ANGGOTA 00

ID
Masukkan Id Anggota

NAMA
Masukkan Id Anggota

EMAIL
Masukkan Nama Anggota

TELEPHONE
Masukkan Nomor Telephone

TANGGAL BERGABUNG
Masukkan Tanggal bergabung

JENIS KELAMIN

Laki - Laki Perempuan

Kode :

```
● ● ●

package Kode;

import Koneksi.Koneksi;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
import javax.swing.table.DefaultTableModel;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class MenuAnggota extends javax.swing.JPanel {

    private static final String ID_Anggota = "id_anggota";
    private static final String Nama_Anggota = "nama_anggota";
    private static final String Email = "email";
    private static final String Telepon = "telepon";
    private static final String Tanggal_Bergabung =
    "tanggal_bergabung";
    private static final String Jenis_Kelamin = "jenis_kelamin";

    private int halamanSaatIni = 1;
    private int dataPerHalaman = 10;
    private int totalPages;

    private final Connection conn;
    private String setIDAnggota;

    public MenuAnggota() {
        JTextField jtxt_id = new JTextField();
        initComponents();

        conn = Koneksi.getConnection();
        setTableModel();
        loadData();
        paginationAnggota();
    }

}
```

```
● ● ●

private void btn_tambahActionPerformed(java.awt.event.ActionEvent evt) {
    jpn_main.removeAll();
    jpn_main.add(jpn_add);
    jpn_main.repaint();
    jpn_main.revalidate();
    jtxt_id.setText(setIDAnggota());
    if(btn_tambah.getText().equals("UBAH")){
        dataTabel();
        btn_simpan.setText("PERBARUI");
    }
}

private void btn_hapusActionPerformed(java.awt.event.ActionEvent evt) {
    deleteData();
}

private void btn_batalActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_btActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_simpanActionPerformed(java.awt.event.ActionEvent evt) {
    if(btn_simpan.getText().equals("TAMBAH") || btn_simpan.getText().equals("SIMPAN")){
        insertData();
    }
    else if (btn_simpan.getText().equals("PERBARUI")){
        updateData();
    }
}

private void tbl_dataMouseClicked(java.awt.event.MouseEvent evt) {
    if(btn_tambah.getText().equals("TAMBAH")){
        btn_tambah.setText("UBAH");
        btn_hapus.setVisible(true);
        btn_batal.setVisible(true);
    }
}

private void txt_cariKeyPressed(java.awt.event.KeyEvent evt) {
    searchData();
}
```

```

private void loadData(){
    calculateTotalPages();
    int totalData = getTotalData();
    jLabel_halaman.setText(String.valueOf("Halaman " + halamanSaatIni + " dari Total Data " +
    totalData));
}

int startIndex = (halamanSaatIni - 1) * dataPerHalaman;
getData(startIndex, dataPerHalaman, (DefaultTableModel) tbl_data.getModel());
btn_hapus.setVisible(false);
btn_batal.setVisible(false);
}

private void showPanel(){
jpn_main.removeAll();
jpn_main.add(new MenuAnggota());
jpn_main.repaint();
jpn_main.revalidate();
}

private void setTabelModel(){
DefaultTableModel model = (DefaultTableModel) tbl_data.getModel();
model.addColumn("ID ANGGOTA");
model.addColumn("NAMA");
model.addColumn("EMAIL");
model.addColumn("TELEPON");
model.addColumn("TANGGAL BERGABUNG");
model.addColumn("JENIS KELAMIN");
}

private void getData(int startIndex, int entriesPage, DefaultTableModel model) {
model.setRowCount(0);

try{
String sql = "SELECT * FROM anggota LIMIT ?,?";
try(PreparedStatement st = conn.prepareStatement(sql)){
st.setInt(1, startIndex);
st.setInt(2, entriesPage);
ResultSet rs = st.executeQuery();

while(rs.next()){
String idAnggota = rs.getString(ID_Anggota);
String namaAnggota = rs.getString(Nama_Anggota);
String emailAnggota = rs.getString(Email);
String teleponAnggota = rs.getString(Telepon);
String tanggalBergabung = rs.getString(Tanggal_Bergabung);
String jenisKelamin = rs.getString(Jenis_Kelamin);

Object[] rowData = {idAnggota, namaAnggota, emailAnggota, teleponAnggota,
tanggalBergabung, jenisKelamin};
model.addRow(rowData);
}
}

catch(Exception e){
Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
}
}

private String setIdAnggota() {
String urutan = null;
Date now = new Date();
SimpleDateFormat noFormat = new SimpleDateFormat("yyMM");
String no = noFormat.format(now);

String sql = "SELECT RIGHT(ID_Anggota, 3) AS Nomor " +
"FROM anggota " +
"WHERE ID_Anggota LIKE 'USK" + no + "%' " +
"ORDER BY Nomor DESC " +
"LIMIT 1";

try (PreparedStatement st = conn.prepareStatement(sql)) {
ResultSet rs = st.executeQuery();

if (rs.next()) {
int nomor = Integer.parseInt(rs.getString("Nomor"));
nomor++;
urutan = "USK" + no + String.format("%03d", nomor);
} else {
urutan = "USK" + no + "001";
}

} catch (SQLException e) {
Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
}
return urutan;
}
}

```

```

private void insertData() {
String idAnggota = jTextField1.getText();
String namaAnggota = jTextField2.getText();
String emailAnggota = jTextField3.getText();
String teleponAnggota = jTextField4.getText();
String tanggalBergabungStr = jTextField5.getText();
String jenisKelamin;

SimpleDateFormat inputFormat = new SimpleDateFormat("dd-MM-yyyy");
SimpleDateFormat outputFormat = new SimpleDateFormat("yyyy-mm-dd");
Date tanggalBergabung = null;

try {
tanggalBergabung = inputFormat.parse(tanggalBergabungStr);
tanggalBergabungStr = outputFormat.format(tanggalBergabung);
}

catch (ParseException e) {
e.printStackTrace();
}

if(jrb_laki.isSelected()){
jenisKelamin = jrb_laki.getText();
}

else if(jrb_perempuan.isSelected()){
jenisKelamin = jrb_perempuan.getText();
}

else {
jenisKelamin = "";
}

if(namaAnggota.isEmpty() || emailAnggota.isEmpty() || teleponAnggota.isEmpty() ||
tanggalBergabungStr.isEmpty() || jenisKelamin.isEmpty()){
 JOptionPane.showMessageDialog(this, "Semua Kolom Wajib Diisi !!!", "Validasi",
JOptionPane.ERROR_MESSAGE);
return;
}

String sqlCheck = "SELECT ID_Anggota FROM anggota WHERE ID_Anggota = ?";
try (PreparedStatement stCheck = conn.prepareStatement(sqlCheck)) {
stCheck.setString(1, idAnggota);
ResultSet rsCheck = stCheck.executeQuery();
if (rsCheck.next()) {
JOptionPane.showMessageDialog(this, "ID Anggota sudah ada dalam database. Silakan
coba lagi.", "Valisan", JOptionPane.ERROR_MESSAGE);
return;
}
}

String sql = "INSERT INTO anggota (ID_Anggota, Nama_Anggota, Email, Telepon, Tanggal_Bergabung,
Jenis_Kelamin) VALUES (?, ?, ?, ?, ?, ?)";
try(PreparedStatement st = conn.prepareStatement(sql)){
st.setString(1, idAnggota);
st.setString(2, namaAnggota);
st.setString(3, emailAnggota);
st.setString(4, teleponAnggota);
st.setString(5, tanggalBergabungStr);
st.setString(6, jenisKelamin);

int rowInserted = st.executeUpdate();

if(rowInserted > 0){
JOptionPane.showMessageDialog(this, "Data Berhasil Ditambahkan");
resetForm();
loadData();
showPanel();
}
}

catch(SQLException e){
Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
}
}

private void updateData() {
String idAnggota = jTextField1.getText();
String namaAnggota = jTextField2.getText();
String emailAnggota = jTextField3.getText();
String teleponAnggota = jTextField4.getText();
String tanggalBergabungStr = jTextField5.getText();
String jenisKelamin = "";

SimpleDateFormat inputFormat = new SimpleDateFormat("dd-MM-yyyy");
SimpleDateFormat outputFormat = new SimpleDateFormat("yyyy-mm-dd");
Date tanggalBergabung = null;

try {
tanggalBergabung = inputFormat.parse(tanggalBergabungStr);
tanggalBergabungStr = outputFormat.format(tanggalBergabung);
}
catch (ParseException e) {
JOptionPane.showMessageDialog(this, "Format tanggal tidak valid.", "Error",
JOptionPane.ERROR_MESSAGE);
return;
}

if (jrb_laki.isSelected()){
jenisKelamin = jrb_laki.getText();
}

else if (jrb_perempuan.isSelected()){
jenisKelamin = jrb_perempuan.getText();
}

else {
JOptionPane.showMessageDialog(this, "Jenis kelamin belum dipilih.", "Error",
JOptionPane.ERROR_MESSAGE);
return;
}

if (namaAnggota.isEmpty() || emailAnggota.isEmpty() || teleponAnggota.isEmpty() ||
tanggalBergabungStr.isEmpty()){
JOptionPane.showMessageDialog(this, "Semua kolom wajib diisi.", "Validasi",
JOptionPane.ERROR_MESSAGE);
return;
}

try {
// Periksa apakah ID Anggota sudah ada dalam database
String sqlCheck = "SELECT ID_Anggota FROM anggota WHERE ID_Anggota = ?";
try (PreparedStatement stCheck = conn.prepareStatement(sqlCheck)) {
stCheck.setString(1, idAnggota);
ResultSet rsCheck = stCheck.executeQuery();
if (!rsCheck.next()) {
JOptionPane.showMessageDialog(this, "ID Anggota tidak ditemukan dalam database.
Silakan tambahkan data baru.", "Validasi", JOptionPane.ERROR_MESSAGE);
return;
}
}

// Lakukan perbarui data
String sqlUpdate = "UPDATE anggota SET Nama_Anggota=?, Email=?, Telepon=?, Tanggal_bergabung=?,
Jenis_Kelamin=? WHERE ID_Anggota=?";
try (PreparedStatement st = conn.prepareStatement(sqlUpdate)) {
st.setString(1, namaAnggota);
st.setString(2, emailAnggota);
st.setString(3, teleponAnggota);
st.setString(4, tanggalBergabungStr);
st.setString(5, jenisKelamin);
st.setString(6, idAnggota);

int rowUpdated = st.executeUpdate();

if (rowUpdated > 0) {
JOptionPane.showMessageDialog(this, "Data berhasil diperbarui.");
resetForm();
loadData();
showPanel();
}
else {
JOptionPane.showMessageDialog(this, "Gagal memperbarui data.", "Error",
JOptionPane.ERROR_MESSAGE);
}
}

catch (SQLException e) {
JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat memperbarui data: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
}
}
}

```

```

private void resetForm() {
    jtxt_id.setText("");
    jtxt_nama.setText("");
    jtxt_email.setText("");
    jtxt_tlp.setText("");
    jtxt_tgl.setText("");
    rbg_jenisKelamin.clearSelection();
}

private void dataTable() {
    jpn_view.setVisible(false);
    jpn_add.setVisible(true);

    int row = tbl_data.getSelectedRow();
    jLabel7.setText("Perbarui Data Anggota Perpustakaan");

    jtxt_id.setEnabled(false);
    jtxt_tgl.setEnabled(false);

    jtxt_id.setText(tbl_data.getValueAt(row, 0).toString());
    jtxt_nama.setText(tbl_data.getValueAt(row, 1).toString());
    jtxt_email.setText(tbl_data.getValueAt(row, 2).toString());
    jtxt_tlp.setText(tbl_data.getValueAt(row, 3).toString());
    jtxt_tgl.setText(tbl_data.getValueAt(row, 4).toString());

    String jenisKelamin = tbl_data.getValueAt(row, 5).toString();
    if(jenisKelamin.equals("Laki - Laki")){
        jrb_laki.setSelected(true);
    }
    else if(jenisKelamin.equals("Perempuan")){
        jrb_perempuan.setSelected(true);
    }
}

private void deleteData() {
    int selectedRow = tbl_data.getSelectedRow();
    int confirm = JOptionPane.showConfirmDialog(
        this, "Apakah Anda yakin ingin Menghapus Data ?",
        "Konfirmasi Hapus Data", JOptionPane.YES_NO_OPTION);

    if(confirm == JOptionPane.YES_OPTION){
        String id = tbl_data.getValueAt(selectedRow, 0).toString();
        try{
            String sql = "DELETE from anggota WHERE ID_Anggota = ?";
            try(PreparedStatement st = conn.prepareStatement(sql)){
                st.setString(1, id);

                int rowDeleted = st.executeUpdate();
                if(rowDeleted > 0){
                    JOptionPane.showMessageDialog(this, "Data Berhasil Dihapus");
                }
                else{
                    JOptionPane.showMessageDialog(this, "Data Gagal Ditambahkan");
                }
            }
        }
        catch(Exception e){
            Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
        }
    }
    resetForm();
    loadData();
    showPanel();
}

private void searchData() {
    String keyword = txt_cari.getText();

    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.setRowCount(0);

    try{
        String sql = "SELECT * FROM anggota WHERE ID_Anggota LIKE ? OR Nama_Anggota LIKE ? OR Email
LIKE ? ";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.setString1("%" + keyword + "%");
            st.setString2("%" + keyword + "%");
            st.setString3("%" + keyword + "%");

            ResultSet rs = st.executeQuery();
            while(rs.next()){
                String idAnggota = rs.getString(ID_Anggota);
                String namaAnggota = rs.getString(Nama_Anggota);
                String emailAnggota = rs.getString(Email);
                String teleponAnggota = rs.getString(Telepon);
                String tanggalBergabung = rs.getString(Tanggal_Bergabung);
                String jenisKelamin = rs.getString(Jenis_Kelamin);

                Object[] rowData = {idAnggota, namaAnggota, emailAnggota, teleponAnggota,
                tanggalBergabung, jenisKelamin};
                model.addRow(rowData);
            }
        }
        catch(Exception e){
            Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
        }
    }
}

```

```

private void paginationAnggota() {
    btn_firts.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = 1;
            loadData();
        }
    });

    btn_previous.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(halamanSaatIni > 1){
                }
                halamanSaatIni--;
                loadData();
            }
        });
    });

    btn_next.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (halamanSaatIni < totalPages){
                halamanSaatIni++;
                loadData();
            }
        }
    });

    btn_last.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = totalPages;
            loadData();
        }
    });

    jcbx_data.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            dataPerHalaman = Integer.parseInt(jcbx_data.getSelectedItem().toString());
            halamanSaatIni = 1;
            loadData();
        }
    });

    private void calculateTotalPages(){
        int totalData = getTotalData();
        totalPages = (int) Math.ceil((double) totalData / dataPerHalaman);
    }

    private int getTotalData() {
        int totalData = 0;

        try{
            String sql = "SELECT COUNT(*) AS total FROM anggota";
            try(PreparedStatement st = conn.prepareStatement(sql)){
                ResultSet rs = st.executeQuery();
                if(rs.next()){
                    totalData = rs.getInt("total");
                }
            }
        }
        catch(Exception e){
            Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
        }
        return totalData;
    }
}

```

Penjelasan :

1. Enkapsulasi:

- konsep enkapsulasi terimplementasi melalui penggunaan variabel userID, namaUser, dan roleUser sebagai variabel instance private. Ini berarti bahwa variabel-variabel tersebut hanya dapat diakses dan dimodifikasi melalui metode getter (seperti getUserId(), getNamaUser(), dll.), yang memungkinkan kontrol yang lebih baik terhadap akses data.

2. Event-Driven:

- Pertama, melalui event mouse pressed dan mouse dragged pada frame (dalam metode formMousePressed() dan formMouseDragged()), pengguna dapat menggeser frame dengan menahan dan menarik mouse.
- Kedua, ketika tombol profil ditekan (dalam metode jbtn_profileActionPerformed()), aplikasi menanggapi event tersebut dengan menampilkan menu profil sesuai dengan kebutuhan.

3. Fungsi Setiap Method:

- addMenu(): Menambahkan menu berdasarkan peran pengguna (admin atau non-admin), memungkinkan untuk menampilkan opsi menu yang sesuai dengan hak akses pengguna.
- setDate(): Mengatur timer untuk menampilkan tanggal dan waktu saat ini secara real-time pada label lb_date, memberikan informasi yang berguna kepada pengguna.
- formMousePressed(): Menangani event ketika mouse ditekan pada frame, memperbarui posisi x dan y dari frame untuk mempersiapkan pergeseran.
- formMouseDragged(): Menangani event ketika mouse di-drag pada frame, menggeser frame sesuai dengan perubahan posisi mouse.
- jbtn_profileActionPerformed(): Menangani event ketika tombol profil ditekan, menampilkan menu profil pada posisi yang sesuai dalam aplikasi.
- formWindowOpened(): Menangani event ketika jendela dibuka, menampilkan menu dashboard sebagai menu utama secara default, memberikan pengalaman pengguna yang mulus.

E. Halaman Data Anggota

Tampilan :

Data Anggota Perpustakaan

Pencarian				
ID ANGGOTA	NAMA	EMAIL	TELEPON	
AGT2310001	Nama15123	email48@example.com	081234567843	
AGT2310002	Nama41	email4@example.com	081234567848	
AGT2310003	Nama15	email27@example.com	081234567843	
AGT2310004	Nama11	email13@example.com	081234567832	
AGT2310005	Nama40	email22@example.com	081234567839	
AGT2310006	Nama43	email30@example.com	081234567818	
AGT2310007	Nama37	email45@example.com	081234567817	
AGT2310008	Nama21	email13@example.com	08123456783	
AGT2310009	Nama44	email21@example.com	081234567824	
AGT2310010	Nama10	email47@example.com	08123456782	
AGT2310011	Nama15	email32@example.com	081234567812	
AGT2310012	Nama48	email19@example.com	081234567849	
AGT2310013	Nama25	email19@example.com	081234567821	
AGT2310014	Nama15	email8@example.com	081234567844	
AGT2310015	Nama12	email19@example.com	08123456786	
AGT2310016	Nama25	email4@example.com	081234567844	

Halaman 1 dari Total Data 47

First Page < > Last Page 10

Kode :

```
package Kode;

import Koneksi.Koneksi;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.table.DefaultTableModel;

public class DataAnggota extends java.awt.Dialog {

    private int halamanSaatIni = 1;
    private int dataPerHalaman = 10;
    private int totalPages;

    private final Connection conn;

    private String idAnggota;
    private String namaAnggota;
    private String emailAnggota;
    private String teleponAnggota;

    public String getIdAnggota() {
        return idAnggota;
    }

    public String getNamaAnggota() {
        return namaAnggota;
    }

    public String getEmailAnggota() {
        return emailAnggota;
    }

    public String getTeleponAnggota() {
        return teleponAnggota;
    }

    public DataAnggota(java.awt.Frame parent, boolean modal)
    {
        super(parent, modal);
        initComponents();

        conn = Koneksi.getConnection();
        setTabelModel();
        loadData();
        paginationAnggota();
    }
}
```

```
private void closedialog(java.awt.event.WindowEvent evt) {
    setVisible(false);
    dispose();
}

private void tbl_dataMouseClicked(java.awt.event.MouseEvent evt) {
    pilihData();
}

private void txt_cariKeyReleased(java.awt.event.KeyEvent evt) {
    searchData();
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            DataAnggota dialog = new DataAnggota(new java.awt.Frame(), true);
            dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                public void windowClosing(java.awt.event.WindowEvent e) {
                    System.exit(0);
                }
            });
            dialog.setVisible(true);
        }
    });
}
```

```

private void paginationAnggota() {
    btn_firts.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = 1;
            loadData();
        }
    });
    btn_previous.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(halamanSaatIni > 1){
                halamanSaatIni--;
                loadData();
            }
        }
    });
    btn_next.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (halamanSaatIni < totalPages){
                halamanSaatIni++;
                loadData();
            }
        }
    });
    btn_last.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = totalPages;
            loadData();
        }
    });
    jcbx_data.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            dataPerHalaman =
            Integer.parseInt(jcbox_data.getSelectedItem().toString());
            loadData();
        }
    });
}

private void calculateTotalPages(){
    int totalData = getTotalData();
    totalPages = (int) Math.ceil((double) totalData / dataPerHalaman);
}

```

```

private int getTotalData() {
    int totalData = 0;
    try{
        String sql = "SELECT COUNT(*) AS total FROM anggota";
        PreparedStatement st = conn.prepareStatement(sql);
        ResultSet rs = st.executeQuery();
        if(rs.next()){
            totalData = rs.getInt("total");
        }
    } catch(Exception e){
        Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
    }
    return totalData;
}

private void loadData(){
    calculateTotalPages();
    int totalData = getTotalData();
    jLabel_halaman.setText(String.valueOf("Halaman" + halamanSaatIni + " dari Total Data" +
    totalData));
    int startIndex = (halamanSaatIni - 1) * dataPerHalaman;
    getData(startIndex, dataPerHalaman, (DefaultTableModel) tbl_data.getModel());
}

private void getData(int startIndex, int entriesPage, DefaultTableModel model) {
    model.setRowCount(0);
    try {
        String sql = "SELECT * FROM anggota LIMIT ?, ?";
        try (PreparedStatement st = conn.prepareStatement(sql)) {
            st.setInt(1, startIndex);
            st.setInt(2, entriesPage);
            ResultSet rs = st.executeQuery();
            while (rs.next()) {
                String idAnggota = rs.getString("ID_Anggota");
                String namaAnggota = rs.getString("Nama_Anggota");
                String emailAnggota = rs.getString("Email");
                String teleponAnggota = rs.getString("Telepon");
                Object[] rowData = {idAnggota, namaAnggota, emailAnggota, teleponAnggota};
                model.addRow(rowData);
            }
        }
    } catch (SQLException e) {
        Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void setTableModel(){
    DefaultTableModel model = (DefaultTableModel) tbl_data.getModel();
    model.addColumn("ID ANGGOTA");
    model.addColumn("NAMA");
    model.addColumn("EMAIL");
    model.addColumn("TELEPON");
}

```

```

private void searchData() {
    String keyword = txt_cari.getText();

    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.setRowCount(0);

    try{
        String sql = "SELECT * FROM anggota WHERE ID_Anggota LIKE ? OR Nama_Anggota LIKE ? OR Email
        LIKE ? ";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.setString1("%" + keyword + "%");
            st.setString2("%" + keyword + "%");
            st.setString3("%" + keyword + "%");

            ResultSet rs = st.executeQuery();
            while(rs.next()){
                String idAnggota = rs.getString("ID_Anggota");
                String namaAnggota = rs.getString("Nama_Anggota");
                String emailAnggota = rs.getString("Email");
                String teleponAnggota = rs.getString("Telepon");

                Object[] rowData = {idAnggota, namaAnggota, emailAnggota, teleponAnggota};
                model.addRow(rowData);
            }
        }
    } catch(Exception e){
        Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void pilhData(){
    int row = tbl_data.getSelectedRow();

    idAnggota = tbl_data.getValueAt(row, 0).toString();
    namaAnggota = tbl_data.getValueAt(row, 1).toString();
    emailAnggota = tbl_data.getValueAt(row, 2).toString();
    teleponAnggota = tbl_data.getValueAt(row, 3).toString();

    tbl_data.setVisible(false);
    dispose();
}

```

Penjelasan :

1. Enkapsulasi:

- Dalam kode ini, terdapat konsep enkapsulasi yang diimplementasikan melalui penggunaan variabel instance private seperti idAnggota, namaAnggota, emailAnggota, dan teleponAnggota, yang hanya dapat diakses melalui metode getter (seperti getIdAnggota(), getNamaAnggota(), dll.). Ini membantu dalam menjaga keamanan data dan memastikan bahwa akses terhadap variabel tersebut terkontrol.

2. Event-Driven:

- Event mouse clicked pada tabel data (dalam metode tbl_dataMouseClicked()) untuk memilih data ketika baris tabel diklik.
- Event key released pada kotak pencarian (dalam metode txt_cariKeyReleased()) untuk memicu pencarian data setiap kali pengguna mengetikkan atau menghapus karakter dalam kotak pencarian.
- Event action performed pada tombol navigasi halaman (dalam metode paginationAnggota()) untuk memungkinkan navigasi antar halaman data.

3. Fungsi Setiap Method:

- closeDialog(): Menutup dialog saat jendela ditutup oleh pengguna.
- tbl_dataMouseClicked(): Menangani event ketika baris tabel data diklik, memilih data yang sesuai.
- txt_cariKeyReleased(): Menangani event ketika pengguna mengetikkan atau menghapus karakter dalam kotak pencarian, memicu pencarian data.
- main(): Metode main untuk menjalankan dialog DataAnggota.
- paginationAnggota(): Menangani event pada tombol navigasi halaman untuk memungkinkan navigasi antar halaman data.
- calculateTotalPages(): Menghitung jumlah total halaman berdasarkan jumlah data dan entri per halaman.
- getTotalData(): Menghitung total jumlah data dalam database.
- loadData(): Memuat data anggota dari database berdasarkan halaman saat ini dan jumlah entri per halaman.
- getData(): Mengambil data anggota dari database berdasarkan indeks awal dan jumlah entri per halaman.
- setTableModel(): Mengatur model tabel dengan menambahkan kolom-kolom yang sesuai.
- searchData(): Mencari data anggota berdasarkan kata kunci pencarian.
- pilihData(): Memilih data anggota yang dipilih oleh pengguna dan menutup dialog.

F. Menu Buku

Tampilan :

1) Tampilan Data Buku

Data Buku

Pencarian

TAMBAH HAPUS BATAL

MASTER DATA > BUKU

Halaman of Total Halaman

First Page < > Last Page 10

2) Tampilan Tambah Buku

Tambah Data Buku

SIMPAN BATAL

MASTER DATA > BUKU

ID
Masukkan Id Buku

NAMA
Masukkan Nama Buku

PENGARANG
Masukkan Nama Pengarang

TAHUN TERBIT
Masukkan Tahun Terbit

KATEGORI

PENERBIT

STOK BUKU
Masukkan Stok Buku

Kode :

```
● ● ●

package Kode;

import Koneksi.Koneksi;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.JComboBox;
import javax.swing.table.DefaultTableModel;

public class MenuBuku extends javax.swing.JPanel {

    private String ID_Buku = "id_Buku";
    private String Nama_Buku = "nama_Buku";
    private String Pengarang_Buku =
"pengarang_Buku";
    private String Tahun_Terbit = "tahun_Terbit";
    private String idKategori;
    private String idPenerbit;

    private int halamanSaatIni = 1;
    private int dataPerHalaman = 10;
    private int totalPages;

    private final Connection conn;
    private String setIDBuku;

    public MenuBuku() {
        JTextField jtxt_id = new JTextField();
        initComponents();

        conn = Koneksi.getConnection();
        setTabelModel();
        loadData();
        paginationBuku();
    }
}
```

```
● ● ●

private void btn_tambahActionPerformed(java.awt.event.ActionEvent evt) {
    jpn_main.removeAll();
    jpn_main.add(jpn_add);
    jpn_main.repaint();
    jpn_main.revalidate();

    jtxt_id.setText(setIDBuku());

    if(btn_tambah.getText().equals("UBAH")){
        dataTabel();
        btn_simpan.setText("PERBARUI");
    }
}

private void btn_hapusActionPerformed(java.awt.event.ActionEvent evt) {
    deleteData();
}

private void btn_batalActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_btActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_simpanActionPerformed(java.awt.event.ActionEvent evt) {
    if(btn_simpan.getText().equals("TAMBAH") || btn_simpan.getText().equals("SIMPAN")){
        insertData();
    }
    else if (btn_simpan.getText().equals("PERBARUI")){
        updateData();
    }
}

private void tbl_dataMouseClicked(java.awt.event.MouseEvent evt) {
    if(btn_tambah.getText().equals("TAMBAH")){
        btn_tambah.setText("UBAH");
        btn_hapus.setVisible(true);
        btn_batal.setVisible(true);
    }
}

private void txt_cariKeyPressed(java.awt.event.KeyEvent evt) {
    searchData();
}
```

```

private void loadData(){
    getKategori();
    getPenerbit();
    calculateTotalPages();
    int totalData = getTotalData();
    jlbl_halaman.setText(String.valueOf("Halaman" + halamanSaatIni + " dari Total Data" +
    totalData));
}

int startIndex = (halamanSaatIni - 1) * dataPerHalaman;
getData(startIndex, dataPerHalaman, (DefaultTableModel) tbl_data.getModel());
btn_hapus.setVisible(false);
btn_batal.setVisible(false);
}

private void showPanel(){
    jpn_main.removeAll();
    jpn_main.add(new MenuBuku());
    jpn_main.repaint();
    jpn_main.revalidate();
}

private void setTabelModel(){
    DefaultTableModel model = (DefaultTableModel) tbl_data.getModel();
    model.addColumn("ID Buku");
    model.addColumn("NAMA");
    model.addColumn("PENGARANG BUKU");
    model.addColumn("TAHUN TERBIT");
    model.addColumn("ID KATEGORI");
    model.addColumn("ID PENERBIT");
    model.addColumn("NAMA KATEGORI");
    model.addColumn("NAMA PENERBIT");
    model.addColumn("STOK");
}

private void getData(int startIndex, int entriesPage, DefaultTableModel model) {
    model.setRowCount(0);

    try {
        String sql = "SELECT bk.ID_Buku, bk.Judul_Buku, bk.Pengarang_Buku, bk.Tahun_Terbit,
bk.ID_Kategori, ktg>Nama_Kategori, bk.ID_Penerbit, pnb>Nama_Penerbit, bk.Stok\n" +
        "FROM buku bk\n" +
        "INNER JOIN kategori ktg ON ktg.ID_Kategori = bk.ID_Kategori\n" +
        "INNER JOIN penerbit pnb ON pnb.ID_Penerbit = bk.ID_Penerbit\n" +
        "ORDER BY bk.ID_Buku ASC LIMIT ?, ?";

        try (PreparedStatement st = conn.prepareStatement(sql)) {
            st.setInt(1, startIndex);
            st.setInt(2, entriesPage);
            ResultSet rs = st.executeQuery();

            while (rs.next()) {
                String idBuku = rs.getString("ID_Buku");
                String namaBuku = rs.getString("Judul_Buku");
                String pengarangBuku = rs.getString("Pengarang_Buku");
                String tahunTerbit = rs.getString("Tahun_Terbit");
                String idKategori = rs.getString("ID_Kategori");
                String namaKategori = rs.getString("Nama_Kategori");
                String idPenerbit = rs.getString("ID_Penerbit");
                String namaPenerbit = rs.getString("Nama_Penerbit");
                int stokBuku = rs.getInt("Stok");

                Object[] rowData = {idBuku, namaBuku, pengarangBuku, tahunTerbit, idKategori,
idPenerbit, namaKategori, namaPenerbit, stokBuku};
                model.addRow(rowData);
            }
        }
    } catch (SQLException e) {
        Logger.getLogger(MenuBuku.class.getName()).log(Level.SEVERE, null, e);
    }
}

private String setIDBuku() {
    String urutan = null;
    Date now = new Date();
    SimpleDateFormat noformat = new SimpleDateFormat("yyMM");
    String no = noFormat.format(now);

    String sql = "SELECT RIGHT(ID_Buku, 3) AS Nomor " +
    "FROM buku " +
    "WHERE ID_Buku LIKE 'BKU' + no + '%' " +
    "ORDER BY Nomor DESC " +
    "LIMIT 1";

    try (PreparedStatement st = conn.prepareStatement(sql)) {
        ResultSet rs = st.executeQuery();

        if (rs.next()) {
            int nomor = Integer.parseInt(rs.getString("Nomor"));
            nomor++;
            urutan = "BKU" + no + String.format("%03d", nomor);
        } else {
            urutan = "BKU" + no + "001";
        }
    } catch (SQLException e) {
        Logger.getLogger(MenuBuku.class.getName()).log(Level.SEVERE, null, e);
    }
    return urutan;
}

private void getKategori(){
    try{
        DefaultComboBoxModel<String> model = new DefaultComboBoxModel<>();
        model.addElement("Pilih Kategori");

        String sql = "SELECT ID_Kategori, Nama_Kategori FROM kategori";
        PreparedStatement st = conn.prepareStatement(sql,
                                                ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
        ResultSet rs = st.executeQuery();

        while (rs.next()) {
            String idKategori = rs.getString("ID_Kategori");
            String namaKategori = rs.getString("Nama_Kategori");
            model.addElement(namaKategori);
        }

        jcb_kategori.setModel(model);
        jcb_kategori.addActionListener(e ->{
            int selectedIndex = jcb_kategori.getSelectedIndex();

            if(selectedIndex > 0){
                try{
                    rs.absolute(selectedIndex);
                    idKategori = rs.getString("ID_Kategori");
                } catch(SQLException x){
                    x.printStackTrace();
                }
            }
        });
        jcb_kategori.setModel(model);
    }
    catch(Exception e){
        e.printStackTrace();
    }
}

private void getPenerbit(){
    try{
        DefaultComboBoxModel<String> model = new DefaultComboBoxModel<>();
        model.addElement("Pilih Penerbit");

        String sql = "SELECT ID_Penerbit, Nama_Penerbit FROM penerbit";
        PreparedStatement st = conn.prepareStatement(sql,
                                                ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
        ResultSet rs = st.executeQuery();

        while (rs.next()) {
            String idPenerbit = rs.getString("ID_Penerbit");
            String namaPenerbit = rs.getString("Nama_Penerbit");
            model.addElement(namaPenerbit);
        }

        jcb_penerbit.setModel(model);
        jcb_penerbit.addActionListener(e ->{
            int selectedIndex = jcb_penerbit.getSelectedIndex();

            if(selectedIndex > 0){
                try{
                    rs.absolute(selectedIndex);
                    idPenerbit = rs.getString("ID_Penerbit");
                } catch(SQLException x){
                    x.printStackTrace();
                }
            }
        });
        jcb_penerbit.setModel(model);
    }
    catch(Exception e){
        e.printStackTrace();
    }
}

```

```

private void insertData() {
    String idBuku = jtxt_id.getText();
    String namaBuku = jtxt_nama.getText();
    String pengarangBuku = jtxt_pengarang.getText();
    String tahunTerbit = jtxt_tahunterbit.getText();
    String kategori = jcb_kategori.getSelectedItem().toString();
    String penerbit = jcb_penerbit.getSelectedItem().toString();
    String stokBuku = jtxt_stok.getText().toString();

    if (namaBuku.isEmpty() || pengarangBuku.isEmpty() || tahunTerbit.isEmpty() ||
        "Pilih Kategori".equals(kategori) || "Pilih Penerbit".equals(penerbit) ||
        stokBuku.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Semua Kolom Wajib Diisi !!!", "Validasi",
        JOptionPane.ERROR_MESSAGE);
        return;
    }

    try {
        String sqlCheck = "SELECT ID_Buku FROM buku WHERE ID_Buku = ?";
        try (PreparedStatement stCheck = conn.prepareStatement(sqlCheck)) {
            stCheck.setString(1, idBuku);
            ResultSet rsCheck = stCheck.executeQuery();
            if (rsCheck.next()) {
                JOptionPane.showMessageDialog(this, "ID Buku sudah ada dalam database. Silakan coba lagi.", "Validasi", JOptionPane.ERROR_MESSAGE);
                return;
            }
            rsCheck.close();
        }

        String sql = "INSERT INTO buku (ID_Buku, Judul_Buku, Pengarang_Buku, Tahun_Terbit,
        ID_Kategori, ID_Penerbit, Stok) VALUES (?, ?, ?, ?, ?, ?, ?)";
        try (PreparedStatement st = conn.prepareStatement(sql)) {
            st.setString(1, idBuku);
            st.setString(2, namaBuku);
            st.setString(3, pengarangBuku);
            st.setString(4, tahunTerbit);
            st.setString(5, idKategori);
            st.setString(6, idPenerbit);
            st.setString(7, stokBuku);

            int rowInserted = st.executeUpdate();
            if (rowInserted > 0) {
                JOptionPane.showMessageDialog(this, "Data Berhasil Ditambahkan");
                resetForm();
                loadData();
                showPanel();
            }
        } catch (SQLException e) {
            Logger.getLogger(MenuBuku.class.getName()).log(Level.SEVERE, null, e);
        }
    }

    private void updateData() {
        String idBuku = jtxt_id.getText();
        String namaBuku = jtxt_nama.getText();
        String pengarangBuku = jtxt_pengarang.getText();
        String tahunTerbit = jtxt_tahunterbit.getText();
        String kategori = jcb_kategori.getSelectedItem().toString();
        String penerbit = jcb_penerbit.getSelectedItem().toString();
        int stokBuku = Integer.parseInt(jtxt_stok.getText());

        if (namaBuku.isEmpty() || pengarangBuku.isEmpty() || tahunTerbit.isEmpty() ||
            "Pilih Kategori".equals(kategori) || "Pilih Penerbit".equals(penerbit)) {
            JOptionPane.showMessageDialog(this, "Semua Kolom Wajib Dipilih !!!", "Validasi",
            JOptionPane.ERROR_MESSAGE);
            return;
        }

        try {
            String idKategori = getIdKategori(kategori);
            String idPenerbit = getIdPenerbit(penerbit);

            if (idKategori == null || idPenerbit == null) {
                JOptionPane.showMessageDialog(this, "ID Kategori atau ID Penerbit tidak ditemukan.",
                "Error", JOptionPane.ERROR_MESSAGE);
                return;
            }

            String sql = "UPDATE buku SET Judul_Buku=?, Pengarang_Buku=?, Tahun_Terbit=?,
            ID_Kategori=?, ID_Penerbit=?, Stok=? WHERE ID_Buku=?";
            try (PreparedStatement st = conn.prepareStatement(sql)) {
                st.setString(1, namaBuku);
                st.setString(2, pengarangBuku);
                st.setString(3, tahunTerbit);
                st.setString(4, idKategori);
                st.setInt(5, idPenerbit);
                st.setString(6, stokBuku);
                st.setString(7, idBuku);

                int rowUpdated = st.executeUpdate();
                if (rowUpdated > 0) {
                    JOptionPane.showMessageDialog(this, "Data berhasil diperbarui.");
                    resetForm();
                    loadData();
                    showPanel();
                } else {
                    JOptionPane.showMessageDialog(this, "Gagal memperbarui data.", "Error",
                    JOptionPane.ERROR_MESSAGE);
                }
            } catch (SQLException e) {
                JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat memperbarui data: " +
                e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}

```

```

private void resetForm() {
    jtxt_id.setText("");
    jtxt_nama.setText("");
    jtxt_pengarang.setText("");
    jtxt_tahunterbit.setText("");
}

private void dataTabel() {
    jpn_view.setVisible(false);
    jpn_addr.setVisible(true);

    int row = tbl_data.getSelectedRow();
    jLabel1.setText("Perbarui Data Buku Perpustakaan");

    jtxt_id.setEnabled(false);
    String id = tbl_data.getModel().getValueAt(row, 0).toString();
    jtxt_id.setText(tbl_data.getValueAt(row, 0).toString());
    jtxt_nama.setText(tbl_data.getValueAt(row, 1).toString());
    jtxt_pengarang.setText(tbl_data.getValueAt(row, 2).toString());
    jtxt_tahunterbit.setText(tbl_data.getValueAt(row, 3).toString());
    idKategori = tbl_data.getModel().getValueAt(row, 4).toString();
    idPenerbit = tbl_data.getModel().getValueAt(row, 5).toString();
    jtxt_stok.setText(tbl_data.getValueAt(row, 6).toString());
}

getKategoriID(idKategori);
getPenerbitID(idPenerbit);

private void deleteData() {
    int selectedRow = tbl_data.getSelectedRow();
    int confirm = JOptionPane.showConfirmDialog(
        this, "Apakah Anda yakin ingin Menghapus Data ?",
        "Konfirmasi Hapus Data", JOptionPane.YES_NO_OPTION);

    if(confirm == JOptionPane.YES_OPTION){
        String id = tbl_data.getValueAt(selectedRow, 0).toString();
        try{
            String sql = "DELETE from buku WHERE ID_Buku = ?";
            try(PreparedStatement st = conn.prepareStatement(sql)){
                st.setString(1, id);

                int rowDeleted = st.executeUpdate();
                if(rowDeleted > 0){
                    JOptionPane.showMessageDialog(this, "Data Berhasil Dihapus");
                } else{
                    JOptionPane.showMessageDialog(this, "Data Gagal Ditambahkan");
                }
            }
        } catch(Exception e){
            Logger.getLogger(MenuBuku.class.getName()).log(Level.SEVERE, null, e);
        }
    }
    resetForm();
    loadData();
    showPanel();
}

private void searchData() {
    String keyword = txt_cari.getText();

    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.setRowCount(0);

    try{
        String sql = "SELECT bk.ID_Buku, bk.Judul_Buku, bk.Pengarang_Buku, bk.Tahun_Terbit,
        + "bk.ID_Kategori, ktg>Nama_Kategori, bk.ID_Penerbit, pnb>Nama_Penerbit,
        bk.Stok "
        + "FROM buku bk "
        + "INNER JOIN kategori ktg ON ktg.ID_Kategori = bk.ID_Kategori "
        + "INNER JOIN penerbit pnb ON pnb.ID_Penerbit = bk.ID_Penerbit "
        + "WHERE bk.Judul_Buku LIKE ? OR bk.Pengarang_Buku LIKE ? ";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.setString(1, "%" + keyword + "%");
            st.setString(2, "%" + keyword + "%");

            ResultSet rs = st.executeQuery();

            while (rs.next()) {
                String idBuku = rs.getString("ID_Buku");
                String namaBuku = rs.getString("Judul_Buku");
                String pengarangBuku = rs.getString("Pengarang_Buku");
                String tahunTerbit = rs.getString("Tahun_Terbit");
                String idKategori = rs.getString("ID_Kategori");
                String namaKategori = rs.getString("Nama_Kategori");
                String idPenerbit = rs.getString("ID_Penerbit");
                String namaPenerbit = rs.getString("Nama_Penerbit");
                String stokBuku = rs.getString("Stok");

                Object[] rowData = {idBuku, namaBuku, pengarangBuku, tahunTerbit, idKategori,
                idPenerbit, namaKategori, namaPenerbit, stokBuku};
                model.addRow(rowData);
            }
        }
    } catch(Exception e){
        Logger.getLogger(MenuBuku.class.getName()).log(Level.SEVERE, null, e);
    }
}

```

```

private void paginationBuku() {
    btn_firts.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = 1;
            loadData();
        }
    });
    btn_previous.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(halamanSaatIni > 1){
                halamanSaatIni--;
                loadData();
            }
        }
    });
    btn_next.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (halamanSaatIni < totalPages){
                halamanSaatIni++;
                loadData();
            }
        }
    });
    btn_last.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = totalPages;
            loadData();
        }
    });
    jcbx_data.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            dataPerHalaman = Integer.parseInt(jcbx_data.getSelectedItem().toString());
            halamanSaatIni = 1;
            loadData();
        }
    });
}

private void calculateTotalPages(){
    int totalData = getTotalData();
    totalPages = (int) Math.ceil((double) totalData / dataPerHalaman);
}

private int getTotalData() {
    int totalData = 0;

    try{
        String sql = "SELECT COUNT(*) AS total FROM buku";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            ResultSet rs = st.executeQuery();
            if(rs.next()){
                totalData = rs.getInt("total");
            }
        }
    } catch(Exception e){
        Logger.getLogger(MenuBuku.class.getName()).log(Level.SEVERE, null, e);
    }
    return totalData;
}

private String getIdKategori(String namaKategori) {
    String idKategori = null;
    String sql = "SELECT ID_Kategori FROM kategori WHERE Nama_Kategori = ?";
    try (PreparedStatement st = conn.prepareStatement(sql)) {
        st.setString(1, namaKategori);
        ResultSet rs = st.executeQuery();
        if (rs.next()) {
            idKategori = rs.getString("ID_Kategori");
        }
    } catch (SQLException e) {
        Logger.getLogger(MenuBuku.class.getName()).log(Level.SEVERE, null, e);
    }
    return idKategori;
}

private String getIDPenerbit(String namaPenerbit) {
    String idPenerbit = null;
    String sql = "SELECT ID_Penerbit FROM penerbit WHERE Nama_Penerbit = ?";
    try (PreparedStatement st = conn.prepareStatement(sql)) {
        st.setString(1, namaPenerbit);
        ResultSet rs = st.executeQuery();
        if (rs.next()) {
            idPenerbit = rs.getString("ID_Penerbit");
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "SQL Error: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
    return idPenerbit;
}

private void getKategoriID(String id) {
    try {
        String sql = "SELECT ID_Kategori, Nama_Kategori FROM kategori";
        PreparedStatement st = conn.prepareStatement(sql);
        ResultSet rs = st.executeQuery();

        jcb_kategori.removeAllItems();

        while (rs.next()) {
            String idKategori = rs.getString("ID_Kategori");
            String namaKategori = rs.getString("Nama_Kategori");
            jcb_kategori.addItem(namaKategori);

            if (id.equals(idKategori)) {
                jcb_kategori.setSelectedItem(namaKategori);
            }
        }

        jcb_kategori.addItemListener(new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent e) {
                if (e.getStateChange() == ItemEvent.SELECTED) {
                    String selectedKategori = jcb_kategori.getSelectedItem().toString();
                    updateKategoriID(selectedKategori);
                }
            }
        });
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "SQL Error: " + e.getMessage());
    }
}

private void updateKategoriID(String selectedKategori) {
    try {
        String sql = "SELECT ID_Kategori, Nama_Kategori FROM kategori WHERE Nama_Kategori = ?";
        PreparedStatement st = conn.prepareStatement(sql);
        st.setString(1, selectedKategori);
        ResultSet rs = st.executeQuery();

        if (rs.next()) {
            String idKategori = rs.getString("ID_Kategori");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private void getPenerbitID(String id) {
    try {
        String sql = "SELECT ID_Penerbit, Nama_Penerbit FROM penerbit";
        PreparedStatement st = conn.prepareStatement(sql);
        ResultSet rs = st.executeQuery();

        jcb_penerbit.removeAllItems();

        while (rs.next()) {
            String idPenerbit = rs.getString("ID_Penerbit");
            String namaPenerbit = rs.getString("Nama_Penerbit");
            jcb_penerbit.addItem(namaPenerbit);

            if (id.equals(idPenerbit)) {
                jcb_penerbit.setSelectedItem(namaPenerbit);
            }
        }

        jcb_penerbit.addItemListener(new ItemListener() {
            @Override
            public void itemStateChanged(ItemEvent e) {
                if (e.getStateChange() == ItemEvent.SELECTED) {
                    String selectedPenerbit = jcb_penerbit.getSelectedItem().toString();
                    updatePenerbitID(selectedPenerbit);
                }
            }
        });
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "SQL Error: " + e.getMessage());
    }
}

private void updatePenerbitID(String selectedPenerbit){
    try {
        String sql = "SELECT ID_Penerbit, Nama_Penerbit FROM penerbit WHERE Nama_Penerbit = ?";
        PreparedStatement st = conn.prepareStatement(sql);
        st.setString(1, selectedPenerbit);
        ResultSet rs = st.executeQuery();

        if (rs.next()) {
            String idPenerbit = rs.getString("ID_Penerbit");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

Penjelasan :

1. Enkapsulasi:
 - Konsep enkapsulasi terlihat dalam penggunaan variabel instance (private fields) yang diakses melalui metode getter dan setter. Contohnya adalah variabel conn, halamanSaatIni, dataPerHalaman, totalPages, ID_Buku, Nama_Buku, dan sebagainya. Variabel-variabel tersebut diakses melalui metode-metode public yang disediakan oleh kelas.
2. Polimorfisme:
 - Polimorfisme terjadi dalam beberapa metode, terutama dalam metode insertData() dan updateData(). Kedua metode tersebut dapat menerima argumen dengan tipe yang berbeda, yaitu data yang ingin dimasukkan atau diperbarui dalam database.
3. Event-driven:
 - Kode menggunakan pendekatan event-driven, dimana tindakan pengguna seperti klik tombol atau pemilihan item dari combo box diproses menggunakan event listener. Contohnya adalah pada metode btn_tambahActionPerformed(), btn_hapusActionPerformed(), jcb_kategori.addActionListener(), dan sebagainya.
4. Exception Handling:
 - Penanganan exception dilakukan untuk menangani kemungkinan kesalahan dalam koneksi database, eksekusi SQL, dan operasi lainnya. Misalnya, di dalam metode insertData() dan updateData(), terdapat blok try-catch untuk menangani kemungkinan kesalahan dalam eksekusi query SQL.
5. Fungsi dari setiap methodnya:
 - btn_tambahActionPerformed(), btn_hapusActionPerformed(), btn_batalActionPerformed(), btn_btlActionPerformed(), btn_simpanActionPerformed(): Mengatur tindakan yang terjadi saat tombol ditekan, seperti menambah, menghapus, membatalkan, atau menyimpan data.
 - jtxt_pengarangActionPerformed(), jtxt_tahunterbitActionPerformed(), txt_cariActionPerformed(), jtxt_idActionPerformed(): Mengatur tindakan yang terjadi saat teks diubah atau ditekan tombol enter.
 - tbl_dataMouseClicked(), txt_cariKeyTyped(): Mengatur tindakan yang terjadi saat tabel atau teks pencarian diubah.
 - loadData(), showPanel(), setTabelModel(), getData(), setIDBuku(), getKategori(), getPenerbit(), insertData(), updateData(), resetForm(), dataTabel(), deleteData(), searchData(), paginationBuku(), calculateTotalPages(), getTotalData(), getIDKategori(), getIDPenerbit(), getKategoriID(), updateKategoriID(), getPenerbitID(), updatePenerbitID(): Metode-metode ini bertanggung jawab atas berbagai tindakan terkait dengan pengelolaan data buku, seperti mengambil data dari database, memperbarui tampilan, menambahkan atau menghapus data, dan mengatur navigasi halaman.

G. Halaman Data Buku

Tampilan :



Kode :

```
package Kode;

import Koneksi.Koneksi;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.table.DefaultTableModel;

public class DataBuku extends java.awt.Dialog {

    private int halamanSaatIni = 1;
    private int dataPerHalaman = 10;
    private int totalPages;
    private final Connection conn;
    private String idBuku;
    private String judulBuku;
    private String pengarangBuku;
    private String penerbitBuku;

    public String getIdBuku() {
        return idBuku;
    }

    public String getJudulBuku() {
        return judulBuku;
    }

    public String getPengarangBuku() {
        return pengarangBuku;
    }

    public String getPenerbitBuku() {
        return penerbitBuku;
    }

    public DataBuku(java.awt.Frame parent, boolean modal)
    {
        super(parent, modal);
        initComponents();

        conn = Koneksi.getConnection();
        setTabelModel();
        loadData();
        paginationBuku();
    }
}
```

```
private void closeDialog(java.awt.event.WindowEvent evt) {
    setVisible(false);
    dispose();
}

private void tbl_dataMouseClicked(java.awt.event.MouseEvent evt) {
    pilihData();
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            DataBuku dialog = new DataBuku(new java.awt.Frame(), true);
            dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                public void windowClosing(java.awt.event.WindowEvent e) {
                    System.exit(0);
                }
            });
            dialog.setVisible(true);
        }
    });
}
```

```

private void paginationBuku() {
    btn_firts.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = 1;
            loadData();
        }
    });

    btn_previous.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(halamanSaatIni > 1){
                halamanSaatIni--;
                loadData();
            }
        }
    });

    btn_next.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (halamanSaatIni < totalPages){
                halamanSaatIni++;
                loadData();
            }
        }
    });

    btn_last.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = totalPages;
            loadData();
        }
    });

    jcbx_data.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            dataPerHalaman =
            Integer.parseInt(jcbx_data.getSelectedItem().toString());
            loadData();
        }
    });
}

private void calculateTotalPages(){
    int totalData = getTotalData();
    totalPages = (int) Math.ceil((double) totalData / dataPerHalaman);
}
}

private void searchdata() {
    String keyword = txt_cari.getText();

    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.setRowCount(0);

    try{
        String sql = "SELECT bk.ID_Buku, bk.Judul_Buku, bk.Pengarang_Buku, bk.Tahun_Terbit, "
        + "bk.ID_Kategori, ktg.Nama_Kategori, bk.ID_Penerbit, pnb.Nama_Penerbit,
        bk.Stok "
        + "FROM buku bk "
        + "INNER JOIN kategori ktg ON ktg.ID_Kategori = bk.ID_Kategori "
        + "INNER JOIN penerbit pnb ON pnb.ID_Penerbit = bk.ID_Penerbit "
        + "WHERE bk.Judul_Buku LIKE ? OR bk.Pengarang_Buku LIKE ? ";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.setString(1, "%" + keyword + "%");
            st.setString(2, "%" + keyword + "%");
        }

        ResultSet rs = st.executeQuery();

        while (rs.next()) {
            String idBuku = rs.getString("ID_Buku");
            String namaBuku = rs.getString("Judul_Buku");
            String pengarangBuku = rs.getString("Pengarang_Buku");
            String tahunTerbit = rs.getString("Tahun_Terbit");
            String idKategori = rs.getString("ID_Kategori");
            String namaKategori = rs.getString("Nama_Kategori");
            String idPenerbit = rs.getString("ID_Penerbit");
            String namaPenerbit = rs.getString("Nama_Penerbit");
            String stokBuku = rs.getString("Stok");

            Object[] rowData = {idBuku, namaBuku, pengarangBuku, tahunTerbit, idKategori,
            idPenerbit, namaKategori, namaPenerbit, stokBuku};
            model.addRow(rowData);
        }
    }
    catch(Exception e){
        Logger.getLogger(MenuBuku.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void pilihData(){
    int row = tbl_data.getSelectedRow();

    idBuku = tbl_data.getValueAt(row, 0).toString();
    judulBuku = tbl_data.getValueAt(row, 1).toString();
    pengarangBuku = tbl_data.getValueAt(row, 2).toString();
    penerbitBuku = tbl_data.getValueAt(row, 7).toString();

    dispose();
}
}

private void actionBar(){
    txt_cari.addKeyListener(new KeyAdapter(){
        public void keyReleased(KeyEvent e){
            searchData();
        }
    });
}
}

```

```

private int getTotalData() {
    int totalData = 0;

    try{
        String sql = "SELECT COUNT(*) AS total FROM buku";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            ResultSet rs = st.executeQuery();
            if(rs.next()){
                totalData = rs.getInt("total");
            }
        }
    }
    catch(Exception e){
        Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void loadData(){
    calculateTotalPages();
    int totalData = getTotalData();
    jlbl_halaman.setText(String.valueOf("Halaman " + halamanSaatIni + " dari Total Data" +
    totalData));

    int startIndex = (halamanSaatIni - 1) * dataPerHalaman;
    getData(startIndex, dataPerHalaman, (DefaultTableModel)tbl_data.getModel());
}

private void getData(int startIndex, int entriesPage, DefaultTableModel model) {
    model.setRowCount(0);

    try {
        String sql = "SELECT bk.ID_Buku, bk.Judul_Buku, bk.Pengarang_Buku, bk.Tahun_Terbit, "
        + "bk.ID_Kategori, ktg.Nama_Kategori, bk.ID_Penerbit, pnb.Nama_Penerbit,
        bk.Stok "
        + "FROM buku bk "
        + "INNER JOIN kategori ktg ON ktg.ID_Kategori = bk.ID_Kategori "
        + "INNER JOIN penerbit pnb ON pnb.ID_Penerbit = bk.ID_Penerbit "
        + "ORDER BY bk.ID_Buku ASC LIMIT ?,?";

        try (PreparedStatement st = conn.prepareStatement(sql)) {
            st.setInt(1, startIndex);
            st.setInt(2, entriesPage);
            ResultSet rs = st.executeQuery();

            while (rs.next()) {
                String idBuku = rs.getString("ID_Buku");
                String namaBuku = rs.getString("Judul_Buku");
                String pengarangBuku = rs.getString("Pengarang_Buku");
                String tahunTerbit = rs.getString("Tahun_Terbit");
                String idKategori = rs.getString("ID_Kategori");
                String namaKategori = rs.getString("Nama_Kategori");
                String idPenerbit = rs.getString("ID_Penerbit");
                String namaPenerbit = rs.getString("Nama_Penerbit");
                String stokBuku = rs.getString("Stok");

                Object[] rowData = {idBuku, namaBuku, pengarangBuku, tahunTerbit, idKategori,
                idPenerbit, namaKategori, namaPenerbit, stokBuku};
                model.addRow(rowData);
            }
        }
    }
    catch (SQLException e) {
        Logger.getLogger(MenuBuku.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void setTabelModel(){
    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.addColumn("ID Buku");
    model.addColumn("NAMA");
    model.addColumn("PENGARANG BUKU");
    model.addColumn("TAHUN TERBIT");
    model.addColumn("ID KATEGORI");
    model.addColumn("ID PENERBIT");
    model.addColumn("NAMA KATEGORI");
    model.addColumn("NAMA PENERBIT");
    model.addColumn("STOK");
}

```

Penjelasan :

1. Enkapsulasi:

- Konsep enkapsulasi terlihat dalam penggunaan variabel-variabel private seperti idBuku, judulBuku, pengarangBuku, dan penerbitBuku, yang hanya dapat diakses melalui metode getter yang disediakan.

2. Polimorfisme:

- Tidak ada contoh polimorfisme yang jelas dalam kode tersebut. Polimorfisme terjadi ketika suatu objek dapat memiliki banyak bentuk, seperti pada overriding metode atau menggunakan parameter polymorphic.

3. Event-Driven Programming:

- Kode ini mengimplementasikan event-driven programming dengan menanggapi aksi pengguna seperti klik tombol atau input keyboard. Contohnya adalah saat tombol-tombol seperti btn_first, btn_previous, btn_next, dan btn_last diklik, fungsi actionPerformed dijalankan untuk mengatur navigasi halaman data buku.

4. Exception Handling:

- Exception handling digunakan untuk menangani kemungkinan terjadinya kesalahan saat menjalankan kode. Contohnya, blok try-catch digunakan untuk menangani SQLException yang mungkin muncul saat berinteraksi dengan database.

5. Fungsi Setiap Method:

- closeDialog(): Method ini menutup dialog.
- paginationBuku(): Method ini mengatur navigasi halaman data buku.
- calculateTotalPages(): Method ini menghitung jumlah total halaman berdasarkan jumlah data.
- getTotalData(): Method ini mengambil jumlah total data dari database.
- loadData(): Method ini memuat data buku berdasarkan halaman saat ini.
- getData(): Method ini mengambil data buku dari database berdasarkan indeks awal dan jumlah entri per halaman.
- setTableModel(): Method ini mengatur model tabel untuk tampilan data buku.
- searchData(): Method ini mencari data buku berdasarkan kata kunci yang dimasukkan pengguna.
- pilihData(): Method ini menangani pemilihan data buku saat pengguna mengklik pada baris tertentu dalam tabel.
- actionButton(): Method ini menangani aksi pengguna pada tombol pencarian dengan menambahkan event listener untuk mengaktifkan pencarian saat tombol ditekan.

H. Menu Kategori

Tampilan :

- 1) Tampilan Data Kategori

 **Data Kategori Buku** MASTER DATA > KATEGORI 

Pencarian

TAMBAH HAPUS BATAL

Halaman of Total Halaman
First Page < > Last Page 10 ▼

- 2) Tampilan Tambah Kategori

MASTER DATA > PENERBIT 

 **Tambah Data Kategori Buku**

SIMPAN BATAL

ID

NAMA

DESKRIPSI BUKU

Kode :

```
package Kode;

import Koneksi.Koneksi;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
import javax.swing.table.DefaultTableModel;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class MenuKategori extends javax.swing.JPanel {

    private static final String ID_Kategori = "id_Kategori";
    private static final String Nama_Kategori = "nama_Kategori";
    private static final String Deskripsi_Kategori =
"Deskripsi_Kategori";
    private int halamanSaatIni = 1;
    private int dataPerHalaman = 10;
    private int totalPages;

    private final Connection conn;
    private String setIDKategori;

    public MenuKategori() {
        JTextField jtxt_id = new JTextField();
        initComponents();

        conn = Koneksi.getConnection();
        setTableModel();
        loadData();
        pagination();
    }
}
```

```
private void btn_tambahActionPerformed(java.awt.event.ActionEvent evt) {
    jpn_main.removeAll();
    jpn_main.add(jpn_add);
    jpn_main.repaint();
    jpn_main.revalidate();
    jtxt_id.setText(setIDKategori());
    if(btn_tambah.getText().equals("UBAH")){
        dataTabel();
        btn_simpan.setText("PERBARUI");
    }
}

private void btn_hapusActionPerformed(java.awt.event.ActionEvent evt) {
    deleteData();
}

private void btn_batalActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void tbl_dataMouseClicked(java.awt.event.MouseEvent evt) {
    if(btn_tambah.getText().equals("TAMBAH")){
        btn_tambah.setText("UBAH");
        btn_hapus.setVisible(true);
        btn_batal.setVisible(true);
    }
}

private void txt_cariKeyTyped(java.awt.event.KeyEvent evt) {
    searchData();
}

private void btn_btlActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_simpanActionPerformed(java.awt.event.ActionEvent evt) {
    if(btn_simpan.getText().equals("TAMBAH") || btn_simpan.getText().equals("SIMPAN")){
        insertData();
    }
    else if (btn_simpan.getText().equals("PERBARUI")){
        updateData();
    }
}
```

```

private void loadData(){
    calculateTotalPages();
    int totalData = getTotalData();
    jlbl_halaman.setText(String.valueOf("Halaman" + halamanSaatIni + " dari Total Data
alData"));

    int startIndex = (halamanSaatIni - 1) * dataPerHalaman;
    getData(startIndex, dataPerHalaman, (DefaultTableModel) tbl_data.getModel());
    btn_hapus.setVisible(false);
    btn_batal.setVisible(false);
}

private void showPanel(){
    jpn_main.removeAll();
    jpn_main.add(new MenuKategori());
    jpn_main.repaint();
    jpn_main.revalidate();
}

private void setTabelModel(){
    DefaultTableModel model = (DefaultTableModel) tbl_data.getModel();
    model.addColumn("ID KATEGORI");
    model.addColumn("NAMA");
    model.addColumn("DESKRIPSI KATEGORI ");
}

private void getData(int startIndex, int entriesPage, DefaultTableModel model) {
    model.setRowCount(0);

    try{
        String sql = "SELECT * FROM kategori LIMIT ?,?";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.setInt(1, startIndex);
            st.setInt(2, entriesPage);
            ResultSet rs = st.executeQuery();

            while(rs.next()){
                String idKategori = rs.getString(ID_Kategori);
                String namaKategori = rs.getString(Nama_Kategori);
                String deskripsiKategori = rs.getString(Deskripsi_Kategori);

                Object[] rowData = {idKategori, namaKategori, deskripsiKategori};
                model.addRow(rowData);
            }
        }
    }

    catch(Exception e){
        Logger.getLogger(MenuKategori.class.getName()).log(Level.SEVERE, null, e);
    }
}

private String setIDKategori(){
    String urutan = null;
    Date now = new Date();
    SimpleDateFormat noFormat = new SimpleDateFormat("yyMM");
    String no = noFormat.format(now);

    String sql = "SELECT RIGHT(ID_Kategori, 3) As Nomor " +
        "FROM kategori " +
        "WHERE ID_Kategori LIKE 'KTG' + no + '%' " +
        "ORDER BY ID_Kategori DESC " +
        "LIMIT 1";

    try(PreparedStatement st = conn.prepareStatement(sql)){
        ResultSet rs = st.executeQuery();

        if (rs.next()){
            int nomor = Integer.parseInt(rs.getString("Nomor"));
            nomor++;
            urutan = "KTG" + no + String.format("%03d", nomor);
        }
        else{
            urutan = "KTG" + no + "001";
        }
    }

    catch(SQLException e){
        Logger.getLogger(MenuKategori.class.getName()).log(Level.SEVERE, null, e);
    }
    return urutan;
}

```

```

public boolean validasNama() {
    boolean valid = true;

    String idKategori = jtxt_id.getText();
    String namaKategori = jtxt_nama.getText();

    String sql = "SELECT Nama_Kategori FROM kategori WHERE ID_Kategori != ? AND Nama_Kategori LIKE
BINAR ?";
    try (PreparedStatement st = conn.prepareStatement(sql)) {
        st.setString(1, idKategori);
        st.setString(2, namaKategori);

        ResultSet rs = st.executeQuery();

        if (rs.next()) {
            valid = false;
            JOptionPane.showMessageDialog(this, "Nama Kategori Sudah Ada\nSilahkan Input yang
Lain", "Peringatan", JOptionPane.WARNING_MESSAGE);
        }
    } catch (SQLException e) {
        Logger.getLogger(MenuKategori.class.getName()).log(Level.SEVERE, null, e);
    }

    return valid;
}

private void insertData() {
    String idKategori = selIDKategori();
    String namaKategori = jtxt_nama.getText();
    String deskripsiKategori = jtxt_deskripsi.getText();

    if(idKategori.isEmpty() || namaKategori.isEmpty() || deskripsiKategori.isEmpty()){
        JOptionPane.showMessageDialog(this, "Semua Kolom Wajib Diisi !!!", "Validasi",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    if(!validasNama()){
        return;
    }

    try{
        String sqlCheck = "SELECT ID_Kategori FROM Kategori WHERE ID_Kategori = ?";
        try (PreparedStatement stCheck = conn.prepareStatement(sqlCheck)) {
            stCheck.setString(1, idKategori);
            ResultSet rsCheck = stCheck.executeQuery();
            if (rsCheck.next()) {
                JOptionPane.showMessageDialog(this, "ID Kategori sudah ada dalam database. Silakan
coba lagi.", "Validasi", JOptionPane.ERROR_MESSAGE);
                return;
            }
        }

        String sql = "INSERT INTO Kategori (ID_Kategori, Nama_Kategori, Deskripsi_Kategori) VALUES (?, ?, ?)";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.setString(1, idKategori);
            st.setString(2, namaKategori);
            st.setString(3, deskripsiKategori);

            int rowInserted = st.executeUpdate();

            if(rowInserted > 0){
                JOptionPane.showMessageDialog(this, "Data Berhasil Ditambahkan");
                resetForm();
                loadData();
                showPanel();
            }
        }

        catch(SQLException e){
            Logger.getLogger(MenuKategori.class.getName()).log(Level.SEVERE, null, e);
        }
    }

    private void updateData() {
        String idKategori = jtxt_id.getText();
        String namaKategori = jtxt_nama.getText();
        String deskripsiKategori = jtxt_deskripsi.getText();

        if (idKategori.isEmpty() || namaKategori.isEmpty() || deskripsiKategori.isEmpty()){
            JOptionPane.showMessageDialog(this, "Semua kolom wajib diisi.", "Validasi",
JOptionPane.ERROR_MESSAGE);
            return;
        }

        try {
            String sqlCheck = "SELECT ID_Kategori FROM kategori WHERE ID_Kategori = ?";
            try (PreparedStatement stCheck = conn.prepareStatement(sqlCheck)) {
                stCheck.setString(1, idKategori);
                ResultSet rsCheck = stCheck.executeQuery();
                if (!rsCheck.next()){
                    JOptionPane.showMessageDialog(this, "ID Kategori tidak ditemukan dalam database.
Silakan tambahkan data baru.", "Validasi", JOptionPane.ERROR_MESSAGE);
                    return;
                }
            }

            String sql = "UPDATE kategori SET Nama_Kategori=? , Deskripsi_Kategori=? WHERE
ID_Kategori=?";
            try (PreparedStatement st = conn.prepareStatement(sql)) {
                st.setString(1, namaKategori);
                st.setString(2, deskripsiKategori);
                st.setString(3, idKategori);

                int rowUpdated = st.executeUpdate();
                if (rowUpdated > 0) {
                    JOptionPane.showMessageDialog(this, "Data berhasil diperbarui.");
                    resetForm();
                    loadData();
                    showPanel();
                } else {
                    JOptionPane.showMessageDialog(this, "Gagal memperbarui data.", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }

            catch (SQLException e) {
                JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat memperbarui data: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}

```

```

private void resetForm() {
    jtxt_id.setText("");
    jtxt_nama.setText("");
    jtxt_deskripsi.setText("");
}

private void dataTabel() {
    jpn_view.setVisible(false);
    jpn_add.setVisible(true);

    int row = tbl_data.getSelectedRow();
    jLabel7.setText("Perbarui Data Kategori Perpustakaan");

    txt_id.setEnabled(false);
    jtxt_id.setText(tbl_data.getValueAt(row, 0).toString());
    jtxt_nama.setText(tbl_data.getValueAt(row, 1).toString());
    jtxt_deskripsi.setText(tbl_data.getValueAt(row, 2).toString());
}

private void deleteData() {
    int selectedRow = tbl_data.getSelectedRow();
    int confirm = JOptionPane.showConfirmDialog(
        this, "Apakah Anda ingin Menghapus Data ??",
        "Konfirmasi Hapus Data", JOptionPane.YES_NO_OPTION);

    if(confirm == JOptionPane.YES_OPTION){
        String id = tbl_data.getValueAt(selectedRow, 0).toString();
        try{
            String sql = "DELETE FROM kategori WHERE ID_Kategori = ?";
            tryPreparedStatement st = conn.prepareStatement(sql);
            st.setString(1, id);

            int rowDeleted = st.executeUpdate();
            if(rowDeleted > 0){
                JOptionPane.showMessageDialog(this, "Data Berhasil Dihapus");
            } else{
                JOptionPane.showMessageDialog(this, "Data Gagal Ditambahkan");
            }
        }
        catch(Exception e){
            Logger.getLogger(MenuKategori.class.getName()).log(Level.SEVERE, null, e);
        }
    }
    resetForm();
    loadData();
    showPanel();
}

private void searchData() {
    String keyword = txt_cari.getText();

    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.setRowCount(0);

    try{
        String sql = "SELECT * FROM kategori WHERE ID_Kategori LIKE ? OR Nama_Kategori LIKE ? OR
Deskripsi_Kategori LIKE ?";
        tryPreparedStatement st = conn.prepareStatement(sql);
        st.setString(1, "%" + keyword + "%");
        st.setString(2, "%" + keyword + "%");
        st.setString(3, "%" + keyword + "%");

        ResultSet rs = st.executeQuery();

        while(rs.next()){
            String idKategori = rs.getString(ID_Kategori);
            String namaKategori = rs.getString(Nama_Kategori);
            String deskripsiKategori = rs.getString(Deskripsi_Kategori);

            Object[] rowData = {idKategori, namaKategori, deskripsiKategori};
            model.addRow(rowData);
        }
    }
    catch(Exception e){
        Logger.getLogger(MenuKategori.class.getName()).log(Level.SEVERE, null, e);
    }
}

```

```

private void pagination() {
    btn_firts.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = 1;
            loadData();
        }
    });

    btn_previous.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(halamanSaatIni > 1){
                halamanSaatIni--;
                loadData();
            }
        }
    });

    btn_next.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (halamanSaatIni < totalPages){
                halamanSaatIni++;
                loadData();
            }
        }
    });

    btn_last.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = totalPages;
            loadData();
        }
    });

    jcbx_data.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            dataPerHalaman = Integer.parseInt(jcbx_data.getSelectedItem().toString());
            halamanSaatIni = 1;
            loadData();
        }
    });
}

private void calculateTotalPages(){
    int totalData = getTotalData();
    totalPages = (int) Math.ceil((double) totalData / dataPerHalaman);
}

private int getTotalData() {
    int totalData = 0;

    try{
        String sql = "SELECT COUNT(*) AS total FROM kategori";
        tryPreparedStatement st = conn.prepareStatement(sql);
        ResultSet rs = st.executeQuery();
        if(rs.next()){
            totalData = rs.getInt("total");
        }
    }
    catch(Exception e){
        Logger.getLogger(MenuKategori.class.getName()).log(Level.SEVERE, null, e);
    }

    return totalData;
}

```

Penjelasan :

1. Enkapsulasi:

- enkapsulasi dalam kode tersebut adalah penggunaan variabel instance private seperti halamanSaatIni, dataPerHalaman, dan totalPages, yang hanya dapat diakses di dalam kelas MenuKategori.

2. Pewarisan:

- Pewarisan adalah konsep dimana sebuah kelas dapat mewarisi atribut dan metode dari kelas lain. Dalam kode tersebut, kelas MenuKategori tidak mewarisi dari kelas lain secara langsung, namun dapat dianggap mewarisi fungsi-fungsi dasar dari kelas javax.swing.JPanel.
- Karena kelas MenuKategori adalah sebuah JPanel, ia dapat menggunakan atau mewarisi metode-metode yang ada dalam kelas JPanel seperti removeAll(), repaint(), dan revalidate().

3. Event-Driven Programming:

- menanggapi aksi pengguna seperti mengklik tombol, mengetik di bidang teks, atau memilih opsi dari daftar.

4. Exception Handling:

- terdapat blok try-catch yang digunakan untuk menangani pengecualian yang mungkin terjadi saat berinteraksi dengan database atau melakukan operasi IO (Input/Output). Hal ini terlihat dalam metode insertData(), updateData(), validasiNama(), dan lainnya.

5. Fungsi dari setiap metode dalam kode sebagai berikut:

- MenuKategori() : Konstruktor untuk kelas MenuKategori, digunakan untuk inisialisasi komponen GUI dan koneksi database.
- btn_tambahActionPerformed() : Aksi yang dilakukan ketika tombol "Tambah" di-klik, mengganti panel tampilan ke panel tambah data.
- btn_hapusActionPerformed() : Aksi yang dilakukan ketika tombol "Hapus" di-klik, menghapus data yang dipilih dari tabel.
- btn_batalActionPerformed() : Aksi yang dilakukan ketika tombol "Batal" di-klik, menampilkan panel utama dan memuat ulang data.
- tbl_dataMouseClicked() : Aksi yang dilakukan ketika baris di tabel data diklik, mengubah teks tombol "Tambah" menjadi "Ubah" dan menampilkan tombol "Hapus" dan "Batal".
- txt_cariKeyTyped() : Aksi yang dilakukan ketika teks dalam bidang pencarian digubah, melakukan pencarian data sesuai dengan kata kunci yang dimasukkan.
- btn_previousActionPerformed() : Aksi yang dilakukan ketika tombol "Sebelumnya" di-klik pada navigasi halaman, memuat data halaman sebelumnya.
- jtxt_deskripsiActionPerformed() : Aksi yang dilakukan ketika teks dalam bidang deskripsi kategori di-enter, tidak melakukan apapun dalam implementasi yang diberikan.

- jtxt_idActionPerformed() : Aksi yang dilakukan ketika teks dalam bidang ID kategori di-enter, tidak melakukan apapun dalam implementasi yang diberikan.
- btn_btlActionPerformed() : Aksi yang dilakukan ketika tombol "Batal" di-klik pada panel tambah data, menampilkan panel utama dan memuat ulang data.
- btn_simpanActionPerformed() : Aksi yang dilakukan ketika tombol "Simpan" di-klik, memilih antara menyimpan data baru atau memperbarui data yang ada.
- loadData() : Memuat data dari database ke dalam tabel dengan memperhitungkan halaman yang sedang ditampilkan.
- showPanel() : Menampilkan panel utama.
- setTabelModel() : Mengatur model tabel untuk tabel data.
- getData() : Mengambil data dari database berdasarkan halaman yang ditentukan.
- setIDKategori() : Menghasilkan ID kategori baru berdasarkan urutan terakhir dari database.
- validasiNama() : Memvalidasi nama kategori untuk memastikan tidak ada duplikasi.
- insertData() : Memasukkan data kategori baru ke dalam database.
- updateData() : Memperbarui data kategori yang ada di dalam database.
- resetForm() : Mengosongkan formulir input.
- dataTabel() : Mengisi formulir input dengan data kategori yang dipilih untuk diedit.
- deleteData() : Menghapus data kategori dari database.
- searchData() : Mencari data kategori berdasarkan kata kunci yang dimasukkan.
- pagination() : Mengatur navigasi halaman untuk menampilkan data dalam beberapa halaman.
- calculateTotalPages() : Menghitung total halaman yang diperlukan untuk menampilkan semua data.
- getTotalData() : Menghitung total jumlah data yang ada di dalam database.

I. Menu Penerbit

Tampilan :

- 1) Tampilan Data Penerbit

Data Penerbit Buku

Pencarian

TAMBAH HAPUS BATAL

MASTER DATA > PENERBIT

Halaman of Total Halaman

First Page < > Last Page 10

- 2) Tampilan Tambah Penerbit

MASTER DATA > PENERBIT

Tambah Data Penerbit Buku

SIMPAN BATAL

ID
Masukkan Id Kategori

NAMA
Masukkan Id Kategori

SITUS PENERBIT
Masukkan Situs Penerbit

Kode :

```
package Kode;

import Koneksi.Koneksi;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
import javax.swing.table.DefaultTableModel;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class MenuPenerbit extends javax.swing.JPanel {

    private static final String ID_Penerbit = "id_Penerbit";
    private static final String Nama_Penerbit = "nama_Penerbit";
    private static final String Situs_Penerbit =
"Situs_Penerbit";
    private int halamanSaatIni = 1;
    private int dataPerHalaman = 10;
    private int totalPages;

    private final Connection conn;
    private String setIDPenerbit;

    public MenuPenerbit() {
        JTextField jtxt_id = new JTextField();
        initComponents();

        conn = Koneksi.getConnection();
        setTabelModel();
        loadData();
        pagination();
    }

}
```

```
private void btn_tambahActionPerformed(java.awt.event.ActionEvent evt) {
    jpn_main.removeAll();
    jpn_main.add(jpn_add);
    jpn_main.repaint();
    jpn_main.revalidate();
    jtxt_id.setText(setIDPenerbit());
    if(btn_tambah.getText().equals("UBAH")){
        dataLabel();
        btn_simpan.setText("PERBARUI");
    }
}

private void btn_hapusActionPerformed(java.awt.event.ActionEvent evt) {
    deleteData();
}

private void btn_batalActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void tbl_dataMouseClicked(java.awt.event.MouseEvent evt) {
    if(btn_tambah.getText().equals("TAMBAH")){
        btn_tambah.setText("UBAH");
        btn_hapus.setVisible(true);
        btn_batal.setVisible(true);
    }
}

private void txt_cariKeyTyped(java.awt.event.KeyEvent evt) {
    searchData();
}

private void btn_btlActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_simpanActionPerformed(java.awt.event.ActionEvent evt) {
    if(btn_simpan.getText().equals("TAMBAH") || btn_simpan.getText().equals("SIMPAN")){
        insertData();
    }
    else if (btn_simpan.getText().equals("PERBARUI")){
        updateData();
    }
}
```

```

private void loadData(){
    calculateTotalPages();
    int totalData = getTotalData();
    jlbl_halaman.setText(String.valueOf("Halaman" + halamanSaatIni + " dari Total Data"
    + totalData));
}

int startIndex = (halamanSaatIni - 1) * dataPerHalaman;
getData(startIndex, dataPerHalaman, (DefaultTableModel) tbl_data.getModel());
    btn_hapus.setVisible(false);
    btn_batal.setVisible(false);
}

private void showPanel(){
    jpn_main.removeAll();
    jpn_main.add(new MenuPenerbit());
    jpn_main.repaint();
    jpn_main.revalidate();
}

private void setTableModel(){
    DefaultTableModel model = (DefaultTableModel) tbl_data.getModel();
    model.addColumn("ID PENERBIT");
    model.addColumn("NAMA");
    model.addColumn("SITUS PENERBIT ");
}

private void getData(int startIndex, int entriesPage, DefaultTableModel model) {
    model.setRowCount(0);

    try{
        String sql = "SELECT * FROM penerbit LIMIT ?,?";
        PreparedStatement st = conn.prepareStatement(sql){
            st.setInt(1, startIndex);
            st.setInt(2, entriesPage);
        }
        ResultSet rs = st.executeQuery();

        while(rs.next()){
            String idPenerbit = rs.getString(ID_Penerbit);
            String namaPenerbit = rs.getString(Nama_Penerbit);
            String situsPenerbit = rs.getString(Situs_Penerbit);

            Object[] rowData = {idPenerbit, namaPenerbit, situsPenerbit};
            model.addRow(rowData);
        }
    }
    catch(Exception e){
        Logger.getLogger(MenuPenerbit.class.getName()).log(Level.SEVERE, null, e);
    }
}

private String setIdPenerbit() {
    String urutan = null;
    Date now = new Date();
    SimpleDateFormat noFormat = new SimpleDateFormat("yyMM");
    String no = noFormat.format(now);

    String sql = "SELECT RIGHT(ID_Penerbit, 3) AS Nomor " +
                "FROM penerbit " +
                "WHERE ID_Penerbit LIKE 'PBT' + no + '%' " +
                "ORDER BY ID_Penerbit DESC " +
                "LIMIT 1";

    try (PreparedStatement st = conn.prepareStatement(sql)) {
        ResultSet rs = st.executeQuery();

        if (rs.next()) {
            int nomor = Integer.parseInt(rs.getString("Nomor"));
            nomor++;
            urutan = "PBT" + no + String.format("%03d", nomor);
        } else {
            urutan = "PBT" + no + "001";
        }
    }
    catch (SQLException e) {
        Logger.getLogger(MenuPenerbit.class.getName()).log(Level.SEVERE, null, e);
    }
    return urutan;
}

```

```

public boolean validasiNama() {
    boolean valid = true;

    String idPenerbit = jtxt_id.getText();
    String namaPenerbit = jtxt_nama.getText();

    String sql = "SELECT Nama_Penerbit FROM penerbit WHERE ID_Penerbit != ? AND Nama_Penerbit LIKE
    BINARY ?";
    try (PreparedStatement st = conn.prepareStatement(sql)) {
        st.setString(1, idPenerbit);
        st.setString(2, namaPenerbit);

        ResultSet rs = st.executeQuery();

        if (rs.next()) {
            valid = false;
            JOptionPane.showMessageDialog(this, "Nama Penerbit Sudah Ada\nSilahkan Input yang Lain",
            "Peringatan", JOptionPane.WARNING_MESSAGE);
        }
    }
    catch (SQLException e) {
        Logger.getLogger(MenuPenerbit.class.getName()).log(Level.SEVERE, null, e);
    }
    return valid;
}

private void insertData() {
    String idPenerbit = setIdPenerbit();
    String namaPenerbit = jtxt_nama.getText();
    String situsPenerbit = jtxt_situs.getText();

    if(idPenerbit.isEmpty() || namaPenerbit.isEmpty() || situsPenerbit.isEmpty()){
        JOptionPane.showMessageDialog(this, "Semua Kolom Wajib Diisi !!!", "Validasi",
        JOptionPane.ERROR_MESSAGE);
        return;
    }

    if(!validasiNama()){
        return;
    }

    try{
        String sqlCheck = "SELECT ID_Penerbit FROM penerbit WHERE ID_Penerbit = ?";
        try (PreparedStatement stCheck = conn.prepareStatement(sqlCheck)) {
            stCheck.setString(1, idPenerbit);
            ResultSet rsCheck = stCheck.executeQuery();
            if (rsCheck.next()) {
                JOptionPane.showMessageDialog(this, "ID Penerbit sudah ada dalam database. Silakan
coba lagi.", "Valisan", JOptionPane.ERROR_MESSAGE);
                return;
            }
        }
        String sql = "INSERT INTO penerbit (ID_Penerbit, Nama_Penerbit, Situs_Penerbit) VA

```

```

private void updateData() {
    String idPenerbit = jtxt_id.getText();
    String namaPenerbit = jtxt_nama.getText();
    String situsPenerbit = jtxt_situs.getText();

    if (idPenerbit.isEmpty() || namaPenerbit.isEmpty() || situsPenerbit.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Semua kolom wajib diisi.", "Validasi",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    try {
        String sqlCheck = "SELECT ID_Penerbit FROM penerbit WHERE ID_Penerbit = ?";
        try (PreparedStatement stCheck = conn.prepareStatement(sqlCheck)) {
            stCheck.setString(1, idPenerbit);
            ResultSet rsCheck = stCheck.executeQuery();
            if (!rsCheck.next()) {
                JOptionPane.showMessageDialog(this, "ID Penerbit tidak ditemukan dalam database.
Silakan tambahkan data baru.", "Validasi", JOptionPane.ERROR_MESSAGE);
                return;
            }
        }

        String sql = "UPDATE penerbit SET Nama_Penerbit=?, Situs_Penerbit=? WHERE ID_Penerbit=?";
        try (PreparedStatement st = conn.prepareStatement(sql)) {
            st.setString(1, namaPenerbit);
            st.setString(2, situsPenerbit);
            st.setString(3, idPenerbit);

            int rowUpdated = st.executeUpdate();
            if (rowUpdated > 0) {
                JOptionPane.showMessageDialog(this, "Data berhasil diperbarui.");
                resetForm();
                loadData();
                showPanel();
            } else {
                JOptionPane.showMessageDialog(this, "Gagal memperbarui data.", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat memperbarui data: " +
e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    private void resetForm() {
        jtxt_id.setText("");
    }
}

```

```

private void calculateTotalPages(){
    int totalData = getTotalData();
    totalPages = (int) Math.ceil((double) totalData / dataPerHalaman);
}

private int getTotalData() {
    int totalData = 0;

    try{
        String sql = "SELECT COUNT(*) AS total FROM penerbit";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            ResultSet rs = st.executeQuery();
            if(rs.next()){
                totalData = rs.getInt("total");
            }
        }
    } catch(Exception e){
        Logger.getLogger(MenuPenerbit.class.getName()).log(Level.SEVERE, null,
e);
    }
    return totalData;
}

```

```

private void searchData() {
    String keyword = txt_cari.getText();

    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.setRowCount(0);

    try{
        String sql = "SELECT * FROM penerbit WHERE ID_Penerbit LIKE ? OR Nama_Penerbit LIKE ? OR
Situs_Penerbit LIKE ?";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.setString(1, "%" + keyword + "%");
            st.setString(2, "%" + keyword + "%");
            st.setString(3, "%" + keyword + "%");

            ResultSet rs = st.executeQuery();

            while(rs.next()){
                String idPenerbit = rs.getString(ID_Penerbit);
                String namaPenerbit = rs.getString(Nama_Penerbit);
                String situsPenerbit = rs.getString(Situs_Penerbit);

                Object[] rowData = {idPenerbit, namaPenerbit, situsPenerbit};
                model.addRow(rowData);
            }
        }
    } catch(Exception e){
        logger.getLogger(MenuPenerbit.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void pagination() {
    btn_firts.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = 1;
            loadData();
        }
    });

    btn_previous.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(halamanSaatIni > 1){
                halamanSaatIni--;
                loadData();
            }
        }
    });

    btn_next.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (halamanSaatIni < totalPages){
                halamanSaatIni++;
                loadData();
            }
        }
    });

    btn_last.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = totalPages;
            loadData();
        }
    });

    jcbx_data.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            dataPerHalaman = Integer.parseInt(jcbx_data.getSelectedItem().toString());
            halamanSaatIni = 1;
            loadData();
        }
    });
}

```

Penjelasan :

1. Enkapsulasi:

- Private Fields: Variabel-variabel seperti ID_Penerbit, Nama_Penerbit, dan Situs_Penerbit dideklarasikan sebagai private untuk membatasi akses langsung dari luar kelas.
- Getter dan Setter: Meskipun tidak terlihat dalam kode yang diberikan, biasanya dalam enkapsulasi juga ada penggunaan getter dan setter untuk mengakses dan mengubah nilai dari variabel-variabel tersebut.

2. Event-Driven:

- ActionListener: ActionListener digunakan untuk menangani peristiwa-peristiwa yang terjadi pada komponen-komponen Swing, seperti tombol (button) atau teks (textfield). Contohnya adalah pada method btn_tambahActionPerformed, btn_hapusActionPerformed, btn_batalActionPerformed, dan lain-lain. Ketika tombol tersebut diklik, method-method ini akan dieksekusi.
- MouseEvent: Dalam method tbl_dataMouseClicked, terdapat penanganan event mouse-click pada tabel data. Ketika sebuah baris dalam tabel diklik, aksi tertentu dijalankan.

3. Exception Handling:

- Try-Catch Blocks: Terdapat penggunaan try-catch blocks untuk menangani pengecualian (exceptions) yang mungkin terjadi dalam operasi-operasi database atau eksekusi query.
- Logging: Penggunaan kelas Logger untuk mencatat pesan log yang berisi informasi tentang pengecualian yang terjadi.

4. Fungsi dari Setiap Method:

- loadData(): Method ini digunakan untuk memuat data dari database ke dalam tabel, menghitung jumlah total halaman, dan menampilkan informasi halaman saat ini.
- showPanel(): Method ini digunakan untuk menampilkan panel tertentu di dalam aplikasi.
- setTabelModel(): Method ini digunakan untuk mengatur model tabel dengan menambahkan kolom-kolom yang sesuai.
- getData(): Method ini digunakan untuk mengambil data dari database dan memasukkannya ke dalam model tabel.
- setIDPenerbit(): Method ini digunakan untuk menghasilkan ID penerbit baru berdasarkan urutan terakhir dalam database.
- validasiNama(): Method ini digunakan untuk memvalidasi apakah nama penerbit sudah ada dalam database.
- insertData(): Method ini digunakan untuk memasukkan data baru ke dalam database.
- updateData(): Method ini digunakan untuk memperbarui data yang sudah ada dalam database.

- `resetForm()`: Method ini digunakan untuk mengosongkan nilai dari semua field input.
- `dataTabel()`: Method ini digunakan untuk menampilkan data yang ada dalam tabel ke dalam field input untuk diedit.
- `deleteData()`: Method ini digunakan untuk menghapus data yang dipilih dari database.
- `searchData()`: Method ini digunakan untuk mencari data berdasarkan kata kunci yang dimasukkan.
- `pagination()`: Method ini digunakan untuk menangani aksi-aksi pagination seperti navigasi ke halaman pertama, halaman sebelumnya, halaman berikutnya, dan halaman terakhir, serta menentukan jumlah data yang ditampilkan per halaman.

J. Menu Petugas

Tampilan :

- 1) Tampilan Data Petugas

 **Data Petugas Perpustakaan**

Pencarian

TAMBAH HAPUS BATAL

MASTER DATA > PETUGAS 

Halaman of Total Halaman

First Page < > Last Page 10 ▾

- 2) Tampilan Tambah Petugas

 **Tambah Data Petugas Perpustakaan**

SIMPAN BATAL

ID
Masukkan Id Petugas

NAMA
Masukkan Nama Petugas

USERNAME
Masukkan Username

EMAIL
Masukkan Email

PASSWORD
Masukkan Password

ROLE
Pilih Role : ▾

MASTER DATA > PETUGAS 

Kode :

```
package Kode;

import Koneksi.Koneksi;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
import javax.swing.table.DefaultTableModel;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class MenuPetugas extends javax.swing.JPanel {

    private static final String ID_User = "id_User";
    private static final String Nama_User = "nama_User";
    private static final String Username = "Username";
    private static final String Email = "Email";
    private static final String Role = "Role";
    private static final String Jenis_Kelamin =
"jenis_kelamin";
    private int halamanSaatIni = 1;
    private int dataPerHalaman = 10;
    private int totalPages;

    private final Connection conn;
    private FormLogin formlogin;

    public MenuPetugas() {
        JTextField jtxt_id = new JTextField();
        initComponents();

        conn = Koneksi.getConnection();
        formlogin = new FormLogin();
        setTableModel();
        loadData();
        paginationUser();
    }

}
```

```
private void btn_tambahActionPerformed(java.awt.event.ActionEvent evt) {
    jpn_main.removeAll();
    jpn_main.add(jpn_add);
    jpn_main.repaint();
    jpn_main.revalidate();

    jtxt_id.setText(setIDUser());

    if(btn_tambah.getText().equals("UBAH")){
        dataTabel();
        btn_simpan.setText("PERBARUI");
    }
}

private void btn_hapusActionPerformed(java.awt.event.ActionEvent evt) {
    deleteData();
}

private void btn_batalActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_btlActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_simpanActionPerformed(java.awt.event.ActionEvent evt) {
    if(btn_simpan.getText().equals("TAMBAH") || btn_simpan.getText().equals("SIMPAN")){
        insertData();
    } else if (btn_simpan.getText().equals("PERBARUI")){
        updateData();
    }
}

private void tbl_dataMouseClicked(java.awt.event.MouseEvent evt) {
    if(btn_tambah.getText().equals("TAMBAH")){
        btn_tambah.setText("UBAH");
        btn_hapus.setVisible(true);
        btn_batal.setVisible(true);
    }
}

private void txt_cariKeyTyped(java.awt.event.KeyEvent evt) {
    searchData();
}
```

```

private void btn_tambahActionPerformed(java.awt.event.ActionEvent evt) {
    jpn_main.removeAll();
    jpn_main.add(jpn_add);
    jpn_main.repaint();
    jpn_main.revalidate();
    jtxt_id.setText(setIDUser());
    if(btn_tambah.getText().equals("UBAH")){
        dataLabel();
        btn_simpan.setText("PERBARUI");
    }
}

private void btn_hapusActionPerformed(java.awt.event.ActionEvent evt) {
    deleteData();
}

private void btn_batalActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_btActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_simpanActionPerformed(java.awt.event.ActionEvent evt) {
    if(btn_simpan.getText().equals("TAMBAH") || btn_simpan.getText().equals("SIMPAN")){
        insertData();
    } else if (btn_simpan.getText().equals("PERBARUI")){
        updateData();
    }
}

private void tbl_dataMouseClicked(java.awt.event.MouseEvent evt) {
    if(btn_tambah.getText().equals("TAMBAH")){
        btn_tambah.setText("UBAH");
        btn_hapus.setVisible(true);
        btn_batal.setVisible(true);
    }
}

private void txt_cariKeyPressed(java.awt.event.KeyEvent evt) {
    searchData();
}
}

private String setIDUser() {
    String urutan = null;
    Date now = new Date();
    SimpleDateFormat noformat = new SimpleDateFormat("yyMM");
    String no = noformat.format(now);

    String sql = "SELECT RIGHT(ID_User, 3) AS Nomor " +
                "FROM User " +
                "WHERE ID_User LIKE 'USR' + no + '%' " +
                "ORDER BY Nomor DESC " +
                "LIMIT 1";

    try (PreparedStatement st = conn.prepareStatement(sql)) {
        ResultSet rs = st.executeQuery();

        if (rs.next()) {
            int nomor = Integer.parseInt(rs.getString("Nomor"));
            nomor++;
            urutan = "USR" + no + String.format("%03d", nomor);
        } else {
            urutan = "USR" + no + "001";
        }
    } catch (SQLException e) {
        Logger.getLogger(MenuPetugas.class.getName()).log(Level.SEVERE, null, e);
    }
    return urutan;
}

private void insertData() {
    String lUser = setIDUser();
    String namaUser = jtxt_nama.getText();
    String userName = jtxt_username.getText();
    String emailUser = jtxt_email.getText();
    String password = formLogin.getMd5Java(jtxt_password.getText());
    String roleUser = jcb_role.getSelectedItem().toString();

    if(namaUser.isEmpty() || userName.isEmpty() || emailUser.isEmpty() || jcb_role.getSelectedItem().toString().equals("Pilih Role")){
        JOptionPane.showMessageDialog(this, "Semua Kolom Wajib Diisi !!!", "Validasi", JOptionPane.ERROR_MESSAGE);
        return;
    }

    String sqlCheck = "SELECT ID_User FROM User WHERE ID_User = ?";
    try (PreparedStatement stCheck = conn.prepareStatement(sqlCheck)) {
        stCheck.setString(1, lUser);
        ResultSet rsCheck = stCheck.executeQuery();
        if (!rsCheck.next()) {
            JOptionPane.showMessageDialog(this, "ID User sudah ada dalam database. Silakan coba lagi.", "Validasi", JOptionPane.ERROR_MESSAGE);
            return;
        }
    }

    String sql = "INSERT INTO user (ID_User, Nama_User, Username, Email, Password, Role) VALUES (?, ?, ?, ?, ?, ?)";
    try (PreparedStatement st = conn.prepareStatement(sql)){
        st.setString(1, lUser);
        st.setString(2, namaUser);
        st.setString(3, userName);
        st.setString(4, emailUser);
        st.setString(5, password);
        st.setString(6, roleUser);

        int rowInserted = st.executeUpdate();
        if(rowInserted > 0){
            JOptionPane.showMessageDialog(this, "Data Berhasil Ditambahkan");
            resetForm();
            loadData();
            showPanel();
        }
    }
    catch(SQLException e){
        Logger.getLogger(MenuPetugas.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void updateData() {
    String lUser = jtxt_id.getText();
    String namaUser = jtxt_nama.getText();
    String userName = jtxt_username.getText();
    String emailUser = jtxt_email.getText();
    String roleUser = jcb_role.getSelectedItem().toString();

    if (namaUser.isEmpty() || userName.isEmpty() || emailUser.isEmpty() || jcb_role.getSelectedItem().toString().equals("Pilih Role")){
        JOptionPane.showMessageDialog(this, "Semua kolom wajib diisi.", "Validasi", JOptionPane.ERROR_MESSAGE);
        return;
    }

    String sqlCheck = "SELECT ID_User FROM User WHERE ID_User = ?";
    try (PreparedStatement stCheck = conn.prepareStatement(sqlCheck)) {
        stCheck.setString(1, lUser);
        ResultSet rsCheck = stCheck.executeQuery();
        if (!rsCheck.next()){
            JOptionPane.showMessageDialog(this, "ID User tidak ditemukan dalam database. Silakan tambahkan data baru.", "Validasi", JOptionPane.ERROR_MESSAGE);
            return;
        }
    }

    String sql = "UPDATE User SET Nama_User=?, Username=?, Email=?, Role=? WHERE ID_User=?";
    try (PreparedStatement st = conn.prepareStatement(sql)) {
        st.setString(1, namaUser);
        st.setString(2, userName);
        st.setString(3, emailUser);
        st.setString(4, roleUser);
        st.setString(5, lUser);

        int rowUpdated = st.executeUpdate();
        if (rowUpdated > 0){
            JOptionPane.showMessageDialog(this, "Data berhasil diperbarui.");
            resetForm();
            loadData();
            showPanel();
        } else {
            JOptionPane.showMessageDialog(this, "Gagal memperbarui data.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat memperbarui data: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}
}

```

```

private void resetForm() {
    jtxt_id.setText("");
    jtxt_nama.setText("");
    jtxt_username.setText("");
    jtxt_email.setText("");
    jtxt_password.setText("");
    jcb_role.setSelectedItem("Pilih Role");
}

private void dataTable() {
    jpn_view.setVisible(false);
    jpn_add.setVisible(true);

    int row = tbl_data.getSelectedRow();
    jLabel7.setText("Perbarui Data Petugas Perpustakaan");

    jtxt_id.setEnabled(false);
    jlb_password.setEnabled(false);
    jtxt_password.setEnabled(false);

    jtxt_id.setText(tbl_data.getValueAt(row, 0).toString());
    jtxt_nama.setText(tbl_data.getValueAt(row, 1).toString());
    jtxt_username.setText(tbl_data.getValueAt(row, 2).toString());
    jtxt_email.setText(tbl_data.getValueAt(row, 3).toString());
    jcb_role.setSelectedItem(tbl_data.getValueAt(row, 4).toString());
}

private void deleteData() {
    int selectedRow = tbl_data.getSelectedRow();
    int confirm = JOptionPane.showConfirmDialog(this, "Apakah Anda yakin ingin Menghapus Data?", "Konfirmasi Hapus Data", JOptionPane.YES_NO_OPTION);

    if(confirm == JOptionPane.YES_OPTION){
        String id = tbl_data.getValueAt(selectedRow, 0).toString();
        try{
            String sql = "DELETE FROM user WHERE ID_User = ?";
            PreparedStatement st = conn.prepareStatement(sql);
            st.setString(1, id);

            int rowDeleted = st.executeUpdate();
            if(rowDeleted > 0){
                JOptionPane.showMessageDialog(this, "Data Berhasil Dihapus");
            } else{
                JOptionPane.showMessageDialog(this, "Data Gagal Ditambahkan");
            }
        }
        catch(Exception e){
            Logger.getLogger(MenuPetugas.class.getName()).log(Level.SEVERE, null, e);
        }
    }
    resetForm();
    loadData();
    showPanel();
}

private void searchData() {
    String keyWord = txt_cari.getText();

    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.setRowCount(0);

    try{
        String sql = "SELECT * FROM user WHERE ID_User LIKE ? OR Nama_User LIKE ? OR Email LIKE ?";
        PreparedStatement st = conn.prepareStatement(sql);
        st.setString1("%" + keyWord + "%");
        st.setString2("%" + keyWord + "%");
        st.setString3("%" + keyWord + "%");

        ResultSet rs = st.executeQuery();
        while(rs.next()){
            String idUser = rs.getString(ID_User);
            String namaUser = rs.getString>Nama_User;
            String userName = rs.getString(username);
            String emailUser = rs.getStringEmail();
            String roleUser = rs.getString(role);

            Object[] rowData = {idUser, namaUser, userName, emailUser, roleUser};
            model.addRow(rowData);
        }
    }
    catch(Exception e){
        Logger.getLogger(MenuPetugas.class.getName()).log(Level.SEVERE, null, e);
    }
}

```

```

private void paginationUser() {
    btn_firts.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = 1;
            loadData();
        }
    });

    btn_previous.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(halamanSaatIni > 1){
                halamanSaatIni--;
                loadData();
            }
        }
    });

    btn_next.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (halamanSaatIni < totalPages){
                halamanSaatIni++;
                loadData();
            }
        }
    });

    btn_last.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = totalPages;
            loadData();
        }
    });

    jcbx_data.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            dataPerHalaman = Integer.parseInt(jcbx_data.getSelectedItem().toString());
            halamanSaatIni = 1;
            loadData();
        }
    });
}

private void calculateTotalPages(){
    int totalData = getTotalData();
    totalPages = (int) Math.ceil((double) totalData / dataPerHalaman);
}

private int getTotalData() {
    int totalData = 0;

    try{
        String sql = "SELECT COUNT(*) AS total FROM user";
        PreparedStatement st = conn.prepareStatement(sql);
        ResultSet rs = st.executeQuery();
        if(rs.next()){
            totalData = rs.getInt("total");
        }
    }
    catch(Exception e){
        Logger.getLogger(MenuPetugas.class.getName()).log(Level.SEVERE, null, e);
    }
    return totalData;
}

```

Penjelasan :

1. Enkapsulasi:

- Konsep enkapsulasi terlihat dalam penggunaan atribut private dan metode public untuk mengatur akses ke data dan fungsionalitas objek. Misalnya, penggunaan variabel private seperti ID_User, Nama_User, Username, Email, Role, dan Jenis_Kelamin untuk mencegah akses langsung dari luar kelas, dan metode public untuk mengakses dan memanipulasi data tersebut.

2. Pewarisan:

- Terdapat pewarisan dalam kelas MenuPetugas yang mewarisi sifat-sifat dari kelas javax.swing.JPanel. Hal ini memungkinkan kelas MenuPetugas untuk memiliki fungsionalitas dan karakteristik yang sama dengan JPanel, serta menambahkan fungsionalitas tambahan sesuai kebutuhan.

3. Polimorfisme:

- Polimorfisme terdapat pada Methos insertData();, terdapat pada Variable String password = formlogin.getMd5java(jtxt_password.getText());, Dimana Variabel password ini mengambil method Md5 pada FormLogin dengan cara membuat variable private FormLogin formlogin; untuk menginisiasi FormLogin dan melakukan pembuatan objek dari FormLogin ini formlogin = new FormLogin();

4. Event-Driven:

- Konsep event-driven merujuk pada aplikasi atau program yang merespons kejadian atau interaksi pengguna. Dalam kode ini, Anda menggunakan event-driven programming karena Anda menanggapi acara seperti klik tombol, klik mouse, dan ketikan keyboard. Contohnya adalah metode btn_tambahActionPerformed, btn_hapusActionPerformed, btn_simpanActionPerformed, dan tbl_dataMouseClicked, yang semuanya merespons acara dari antarmuka pengguna.

5. Fungsi dari setiap methodnya sebagai berikut:

- MenuPetugas() : Konstruktor untuk kelas MenuPetugas. Inisialisasi koneksi database dan form login, dan memanggil method untuk mengatur model tabel dan memuat data.
- btn_tambahActionPerformed() : Mengatur tindakan yang akan dilakukan ketika tombol tambah/ubah diklik, seperti menampilkan panel tambah/ubah data.
- btn_hapusActionPerformed() : Menangani tindakan pengguna ketika tombol hapus diklik, seperti menghapus data yang dipilih dari database.
- btn_batalActionPerformed() : Menangani tindakan pengguna ketika tombol batal diklik, seperti membatalkan operasi tambah/ubah data.
- btn_btlActionPerformed() : Menangani tindakan pengguna ketika tombol batal diklik, mirip dengan btn_batalActionPerformed.
- btn_simpanActionPerformed() : Menangani tindakan pengguna ketika tombol simpan diklik, seperti menyimpan data baru atau memperbarui data yang ada.

- `tbl_dataMouseClicked()` : Menanggapi tindakan pengguna ketika sebuah baris di tabel data diklik, seperti mengaktifkan tombol ubah dan hapus.
- `txt_cariKeyTyped()` : Menangani tindakan pengguna saat pengguna mengetikkan teks di bidang pencarian, seperti mencari data sesuai dengan kata kunci yang dimasukkan.
- `loadData()` : Memuat data dari database ke dalam tabel, termasuk mengatur pagination dan menampilkan informasi halaman saat ini.
- `showPanel()` : Menampilkan panel yang sesuai dalam antarmuka pengguna.
- `setTableModel()` : Mengatur model tabel dengan menambahkan kolom yang sesuai.
- `getData()` : Mengambil data dari database dengan batasan tertentu dan memasukkannya ke dalam model tabel.
- `setIDUser()` : Menghasilkan ID unik untuk pengguna berdasarkan tanggal saat ini dan urutan terakhir dari database.
- `insertData()` : Memasukkan data pengguna baru ke dalam database.
- `updateData()` : Memperbarui data pengguna yang ada di database.
- `resetForm()` : Mengatur ulang nilai-nilai input di panel tambah/ubah.
- `dataTabel()` : Memperbarui tampilan panel untuk operasi pengubahan data.
- `deleteData()` : Menghapus data yang dipilih dari database.
- `searchData()` : Mencari data berdasarkan kata kunci yang dimasukkan oleh pengguna.
- `paginationUser()` : Mengatur tindakan tombol untuk navigasi halaman dan menentukan jumlah data per halaman.
- `calculateTotalPages()` : Menghitung total halaman berdasarkan jumlah total data dan data per halaman.
- `getTotalData()` : Mengambil jumlah total data dari database.

K. Menu Peminjaman

Tampilan :

1) Tampilan Data Peminjaman

 **Data Peminjaman**

Pencarian

TAMBAH **BATAL**

TRANSAKSI > PEMINJAMAN  

Halaman 1 of Total Halaman 10

First Page < > Last Page 10

2) Tampilan Tambah Data Peminjaman

 **Tambah Data Peminjam**

TRANSAKSI > PEMINJAMAN  

SIMPAN **BATAL**

ID PEMINJAM	Masukkan Id	JUMLAH	Masukkan Jumlah Buku	TOTAL PINJAM
				TOTAL
TANGGAL PINJAM	Masukkan Tanggal Pinjam	TANGGAL KEMBALI	Masukkan Tanggal Pengembalian	
ANGGOTA	Masukkan Id Anggota	...	BUKU	Masukkan Id Buku
NAMA	Masukkan Nama		JUDUL	Masukkan Judul
EMAIL	Masukkan Email		PENGARANG	Masukkan Pengarang
TELEPHONE	Masukkan Nomor Telephone		PENERBIT	Masukkan Penerbit

UBAH **HAPUS** **BATAL**

Kode :

```
package Kode;

import Koneksi.Koneksi;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

public class MenuPeminjaman extends javax.swing.JPanel {
    private String userID;

    private int halamanSaatIni = 1;
    private int dataPerHalaman = 10;
    private int totalPages;

    private final Connection conn;

    public MenuPeminjaman(String userID) {
        this.userID = userID;
        initComponents();

        conn = Koneksi.getConnection();
        setTabelModel();
        setTabelModelDetail();
        setTabelModelSementara();
        loadData();
        loadDataSementara();
        paginationBuku();
    }

}
```

```
private void btn_tambahActionPerformed(java.awt.event.ActionEvent evt) {
    jpn_main.removeAll();
    jpn_main.add(jpn_add);
    jpn_main.repaint();
    jpn_main.revalidate();

    jtxt_id.setText(setIdPeminjaman());
    jtxt_id.setEnabled(false);
    btn_ubah.setText("TAMBAH");
}

private void btn_batalActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_btActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void btn_simpanActionPerformed(java.awt.event.ActionEvent evt) {
    insertData();
}

private void tbl_dataMouseClicked(java.awt.event.MouseEvent evt) {
    pnDetail.setVisible(true);

    int row = tbl_data.getSelectedRow();
    String id = tbl_data.getValueAt(row, 0).toString();
    getDataDetail((DefaultTableModel)tbl_dataDetail.getModel(), id);
}

private void btn_setanggotaActionPerformed(java.awt.event.ActionEvent evt) {
    setAnggota();
}

private void btn_setbukuActionPerformed(java.awt.event.ActionEvent evt) {
    setBuku();
}

private void jtxt_jumlahbukuActionPerformed(java.awt.event.ActionEvent evt) {
    if(btn_ubah.getText().equals("TAMBAH")){
        insertDataSementara();
    } else if(btn_ubah.getText().equals("UBAH")){
        updateData();
    }
}

private void tbl_datasementaraMouseClicked(java.awt.event.MouseEvent evt) {
    btn_ubah.setText("UBAH");
    btn_hapus.setVisible(true);
    btn_btSementara.setVisible(true);
    dataTabelSementara();
}

private void btn_ubahActionPerformed(java.awt.event.ActionEvent evt) {
    if(btn_ubah.getText().equals("TAMBAH")){
        insertDataSementara();
    } else if(btn_ubah.getText().equals("UBAH")){
        updateData();
    }
}

private void btn_hapusActionPerformed(java.awt.event.ActionEvent evt) {
    deleteData();
}

private void btn_btSementaraActionPerformed(java.awt.event.ActionEvent evt) {
    loadDataSementara();
    resetFormBuku();
    btn_ubah.setText("TAMBAH");
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    showPanel();
    loadData();
}

private void txt_cariKeyReleased(java.awt.event.KeyEvent evt) {
    searchData();
}

private void btn_firtsActionPerformed(java.awt.event.ActionEvent evt) {
    halamanSaatIni = 1;
    loadData();
}

private void btn_previousActionPerformed(java.awt.event.ActionEvent evt) {
    if (halamanSaatIni > 1) {
        halamanSaatIni--;
        loadData();
    }
}

private void btn_nextActionPerformed(java.awt.event.ActionEvent evt) {
    if (halamanSaatIni < totalPages) {
        halamanSaatIni++;
        loadData();
    }
}

private void btn_lastActionPerformed(java.awt.event.ActionEvent evt) {
    halamanSaatIni = totalPages;
    loadData();
}

private void jcpx_dataActionPerformed(java.awt.event.ActionEvent evt) {
    dataPerHalaman = Integer.parseInt(jcpx_data.getSelectedItem().toString());
    halamanSaatIni = 1;
    loadData();
}
```

```

private void loadData(){
    calculateTotalPages();
    int totalData = getTotalData();
    lbl_halaman.setText(String.valueOf("Halaman " + halamanSaatIni + " dari Total Data " +
totalData));
}

int startIndex = (halamanSaatIni - 1) * dataPerHalaman;
getData(startIndex, dataPerHalaman, (DefaultTableModel) tbl_data.getModel());
btn_batal.setVisible(false);
pnDetail.setVisible(false);
}

private void loadDataSementara(){
    getDataSementara((DefaultTableModel)tbl_dataSementara.getModel());
}

btn_ubah.setText("TAMBAH");
btn_hapus.setVisible(false);
btn_btLSementara.setVisible(false);
}

private void showPanel(){
    jpn_main.removeAll();
    jpn_main.add(new MenuPeminjaman(userID));
    jpn_main.repaint();
    jpn_main.revalidate();
}

private void setTabelModel(){
    DefaultTableModel model = (DefaultTableModel) tbl_data.getModel();
    model.addColumn("ID PEMINJAMAN");
    model.addColumn("TANGGAL PEMINJAMAN");
    model.addColumn("TANGGAL PENGEMBALIAN");
    model.addColumn("TOTAL PINJAM");
    model.addColumn("NAMA ANGGOTA");
    model.addColumn("PETUGAS");
}

private void getData(int startIndex, int entriesPage, DefaultTableModel model) {
    model.setRowCount(0);

    try {
        String sql = "SELECT pm.ID_Peminjaman, pm.Tanggal_Peminjaman, pm.Tanggal_Pengembalian,
pm.Total_Pinjam, agt.Nama_Anggota, usr.Nama_User\n" +
"FROM peminjaman pm\n" +
"INNER JOIN anggota agt ON agt.ID_Anggota = pm.ID_Anggota\n" +
"INNER JOIN USER usr ON usr.ID_User = pm.ID_User\n" +
"ORDER BY pm.ID_Peminjaman ASC LIMIT ?, ?";
        PreparedStatement st = conn.prepareStatement(sql);
        st.setInt(1, startIndex);
        st.setInt(2, entriesPage);
        ResultSet rs = st.executeQuery();

        while (rs.next()) {
            String idPeminjam = rs.getString("ID_Peminjaman");
            String tanggalPinjam = rs.getString("Tanggal_Peminjaman");
            String tanggalKembali = rs.getString("Tanggal_Pengembalian");
            String totalPinjam = rs.getString("Total_Pinjam");
            String namaAnggota = rs.getString("Nama_Anggota");
            String namaUser = rs.getString("Nama_User");

            Object[] rowData = {idPeminjam, tanggalPinjam, tanggalKembali, totalPinjam,
namaAnggota, namaUser};
            model.addRow(rowData);
        }
    } catch (SQLException e) {
        Logger.getLogger(MenuPeminjaman.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void setTabelModelSementara(){
    DefaultTableModel model = (DefaultTableModel) tbl_dataSementara.getModel();
    model.addColumn("ID PEMINJAMAN");
    model.addColumn("JUDUL BUKU");
    model.addColumn("PENGARANG");
    model.addColumn("PENERBIT");
    model.addColumn("JUMLAH PINJAM");
}

```



```

private void getDataSementara(DefaultTableModel model) {
    model.setRowCount(0);

    try {
        String sql = "SELECT * FROM sementara";
        PreparedStatement st = conn.prepareStatement(sql);
        ResultSet rs = st.executeQuery();

        while (rs.next()) {
            String idBuku = rs.getString("ID_Buku");
            String judulBuku = rs.getString("Judul_Buku");
            String pengarangBuku = rs.getString("Pengarang_Buku");
            String penerbitBuku = rs.getString("Penerbit_Buku");
            String jumlahPinjam = rs.getString("Jumlah_Pinjam");

            Object[] rowData = {idBuku, judulBuku, pengarangBuku, penerbitBuku, jumlahPinjam};
            model.addRow(rowData);
        }
    } catch (SQLException e) {
        Logger.getLogger(MenuPeminjaman.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void setTabelModelDetail() {
    DefaultTableModel model = (DefaultTableModel) tbl_dataDetail.getModel();
    model.addColumn("ID");
    model.addColumn("ID BUKU");
    model.addColumn("JUDUL");
    model.addColumn("JUMLAH");
    model.addColumn("STATUS PEMINJAMAN");
}

public void getDataDetail(DefaultTableModel model, String id) {
    model.setRowCount(0);

    try {
        String sql = "SELECT pmd.ID_Peminjaman, bk.ID_Buku, bk.Judul_Buku, pmd.Jumlah_Pinjam,
model.Status_Peminjaman\n" +
"FROM detail_peminjaman pmd\n" +
"INNER JOIN buku bk ON bk.ID_Buku = pmd.ID_Buku\n" +
"WHERE pmd.ID_Peminjaman = '"+id+"'"+
"ORDER BY pmd.ID_Peminjaman ASC";
        PreparedStatement st = conn.prepareStatement(sql);
        ResultSet rs = st.executeQuery();

        while (rs.next()) {
            String idPeminjam = rs.get

```

```

private void setAnggota(){
    boolean closeable = true;
    DataAnggota anggota = new DataAnggota(null, closeable);
    anggota.setVisible(true);

    jtxt_idanggota.setText(anggota.getIdAnggota());
    jtxt_nama.setText(anggota.getNamaAnggota());
    jtxt_email.setText(anggota.getEmailAnggota());
    jtxt_telephone.setText(anggota.getTeleponAnggota());

    jtxt_idanggota.setEnabled(false);
    jtxt_nama.setEnabled(false);
    jtxt_email.setEnabled(false);
    jtxt_telephone.setEnabled(false);
}

private void setBuku(){
    boolean closeable = true;
    DataBuku buku = new DataBuku(null, closeable);
    buku.setVisible(true);

    jtxt_idbuku.setText(buku.getIdBuku());
    jtxt_judul.setText(buku.getJudulBuku());
    jtxt_pengarang.setText(buku.getPengarangBuku());
    jtxt_penerbit.setText(buku.getPenerbitBuku());

    jtxt_idbuku.setEnabled(false);
    jtxt_judul.setEnabled(false);
    jtxt_pengarang.setEnabled(false);
    jtxt_penerbit.setEnabled(false);
}

private void insertData() {
    String idPeminjam = jtxt_id.getText();
    String tglPinjam = jtxt_tanggalPinjam.getText();
    String tglKembali = jtxt_tanggalPengembalian.getText();
    String totalPinjam = lb_total.getText();
    String idAnggota = jtxt_idanggota.getText();

    if(idPeminjam.isEmpty() || tglPinjam.isEmpty() || tglKembali.isEmpty() || totalPinjam.isEmpty()
    || idAnggota.isEmpty()){
        JOptionPane.showMessageDialog(this, "Semua kolom harus diisi !", "Validasi",
JOptionPane.ERROR_MESSAGE);
        return;
    }

    try {
        String sql = "INSERT INTO peminjaman (ID_Peminjaman, Tanggal_Peminjaman,
Tanggal_Pengembalian, Total_Pinjam, ID_User, ID_Anggota) VALUES (?, ?, ?, ?, ?, ?)";
        try (PreparedStatement st = conn.prepareStatement(sql)) {
            st.setString(1, idPeminjam);
            st.setString(2, tglPinjam);
            st.setString(3, tglKembali);
            st.setString(4, totalPinjam);
            st.setString(5, userID);
            st.setString(6, idAnggota);

            int rowInserted = st.executeUpdate();

            if (rowInserted > 0) {
                JOptionPane.showMessageDialog(this, "Data Berhasil Ditambahkan");
                insertDataDetail();
                deleteDataSementara();
                resetForm();
                loadData();
                showPanel();
            }
        } catch (SQLException e) {
            Logger.getLogger(MenuPeminjaman.class.getName()).log(Level.SEVERE, null, e);
        }
    }

    private void insertDataDetail() {
        String idPeminjam = jtxt_id.getText();

        try {
            String sql = "INSERT INTO detail_peminjaman (ID_Peminjaman, ID_Buku, Jumlah_Pinjam,
Status_Peminjaman) SELECT '"+idPeminjam+"',ID_Buku,Jumlah_Pinjam,Status_Pinjam FROM sementara";
            try (PreparedStatement st = conn.prepareStatement(sql)) {
                st.executeUpdate();
            } catch (SQLException e) {
                Logger.getLogger(MenuPeminjaman.class.getName()).log(Level.SEVERE, null, e);
            }
        }
    }
}

```

```

private void getDataSementara(DefaultTableModel model) {
    model.setRowCount(0);

    try {
        String sql = "SELECT * FROM sementara";
        try (PreparedStatement st = conn.prepareStatement(sql)) {
            ResultSet rs = st.executeQuery();

            while (rs.next()) {
                String idBuku = rs.getString("ID_Buku");
                String judulBuku = rs.getString("Judul_Buku");
                String pengarangBuku = rs.getString("Pengarang_Buku");
                String penerbitBuku = rs.getString("Penerbit_Buku");
                String jumlahPinjam = rs.getString("Jumlah_Pinjam");

                Object[] rowData = {idBuku, judulBuku, pengarangBuku, penerbitBuku, jumlahPinjam};
                model.addRow(rowData);
            }
        } catch (SQLException e) {
            Logger.getLogger(MenuPeminjaman.class.getName()).log(Level.SEVERE, null, e);
        }
    }

    private void setTableModelDetail() {
        DefaultTableModel model = (DefaultTableModel) tbl_dataDetail.getModel();
        model.addColumn("ID");
        model.addColumn("ID BUKU");
        model.addColumn("JUDUL");
        model.addColumn("JUMLAH");
        model.addColumn("STATUS PEMINJAMAN");
    }

    public void getDataDetail(DefaultTableModel model, String id) {
        model.setRowCount(0);

        try {
            String sql = "SELECT pmd.ID_Peminjaman, bk.ID_Buku, bk.Judul_Buku, pmd.Jumlah_Pinjam,
pmd.Status_Peminjaman\n" +
                "FROM detail_peminjaman pmd\n" +
                "INNER JOIN buku bk ON bk.ID_Buku = pmd.ID_Buku\n" +
                "WHERE pmd.ID_Peminjaman = '"+id+"'"+
                "ORDER BY pmd.ID_Peminjaman ASC";
            try (PreparedStatement st = conn.prepareStatement(sql)) {
                ResultSet rs = st.executeQuery();

                while (rs.next()) {
                    String idPeminjaman = rs.get

```

```

private void resetForm() {
    jtxt_id.setText("");
    jtxt_nama.setText("");
    jtxt_pengarang.setText("");
}

private void resetFormBuku(){
    jtxt_idbuku.setText("");
    jtxt_judul.setText("");
    jtxt_pengarang.setText("");
    jtxt_penerbit.setText("");
    jtxt_jumlahbuku.setText("");
}

private void dataTabelSementara() {
    int row = tbl_datasementara.getSelectedRow();
    if(row!=1){
        jtxt_idbuku.setText(tbl_datasementara.getValueAt(row, 0).toString());
        jtxt_judul.setText(tbl_datasementara.getValueAt(row, 1).toString());
        jtxt_pengarang.setText(tbl_datasementara.getValueAt(row, 2).toString());
        jtxt_penerbit.setText(tbl_datasementara.getValueAt(row, 3).toString());
        jtxt_jumlahbuku.setText(tbl_datasementara.getValueAt(row, 4).toString());

        jtxt_jumlahbuku.requestFocus();
    }
}

private void deleteData() {
    int selectedRow = tbl_datasementara.getSelectedRow();
    int confirm = JOptionPane.showConfirmDialog(this,
        "Apakah yakin ingin menghapus data ini ?",
        "Konfirmasi Hapus Data",
        JOptionPane.YES_NO_OPTION);

    if(confirm == JOptionPane.YES_OPTION){
        String id = tbl_datasementara.getValueAt(selectedRow, 0).toString();
        try {
            String sql = "DELETE FROM sementara WHERE ID_Buku=?";
            try(PreparedStatement st = conn.prepareStatement(sql)){
                st.setString(1, id);
                int rowDeleted = st.executeUpdate();

                int totalPinjam = getTotalPinjam();
                if(totalPinjam > 0){
                    lb_total.setText(String.valueOf(totalPinjam));
                }else{
                    lb_total.setText("Total");
                }

                if(rowDeleted > 0){
                    JOptionPane.showMessageDialog(this, "Data Berhasil Dihapus");
                }else{
                    JOptionPane.showMessageDialog(this, "Data Gagal Dihapus");
                }

                btn_ubah.setText("TAMBAH");
            }
        } catch (SQLException e) {
            Logger.getLogger(MenuPeminjaman.class.getName()).log(Level.SEVERE,null,e);
        }
        resetFormBuku();
        loadDataSementara();
    }
}

private void deleteDataSementara(){
    try{
        String sql = "DELETE FROM sementara";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.executeUpdate();
        }
    } catch(Exception e){
        Logger.getLogger(MenuPeminjaman.class.getName()).log(Level.SEVERE, null, e);
    }
}

```

```

private void searchData() {
    String keyWord = txt_cari.getText();

    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.setRowCount(0);

    try{
        String sql = "SELECT pm.ID_Peminjaman, pm.Tanggal_Peminjaman, pm.Tanggal_Pengembalian, pm.Total_Pinjam, agt.Nama_Anggota, usr.Nama_User\n" +
                    "FROM peminjaman pm\n" +
                    "INNER JOIN anggota agt ON agt.ID_Anggota = pm.ID_Anggota\n" +
                    "INNER JOIN USER usr ON usr.ID_User = pm.ID_User\n" +
                    "WHERE pm.ID_Peminjaman LIKE ? OR agt.Nama_Anggota LIKE ?";

        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.setString(1, "%" + keyword + "%");
            st.setString(2, "%" + keyword + "%");

            ResultSet rs = st.executeQuery();

            while (rs.next()) {
                String idPeminjam = rs.getString("ID_Peminjaman");
                String tanggalPinjam = rs.getString("Tanggal_Peminjaman");
                String tanggalKembali = rs.getString("Tanggal_Pengembalian");
                String totalPinjam = rs.getString("Total_Pinjam");
                String namaAnggota = rs.getString("Nama_Anggota");
                String namaUser = rs.getString("Nama_User");

                Object[] rowData = {idPeminjam, tanggalPinjam, tanggalKembali, totalPinjam, namaAnggota, namaUser};
                model.addRow(rowData);
            }
        }
    } catch(Exception e){
        Logger.getLogger(MenuPeminjaman.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void paginationBuku() {
    btn_firts.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = 1;
            loadData();
        }
    });
}

```

```

private void searchData() {
    String keyWord = txt_cari.getText();

    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.setRowCount(0);

    try{
        String sql = "SELECT pm.ID_Peminjaman, pm.Tanggal_Peminjaman, pm.Tanggal_Pengembalian, pm.Total_Pinjam, agt.Nama_Anggota, usr.Nama_User\n" +
                    "FROM peminjaman pm\n" +
                    "INNER JOIN anggota agt ON agt.ID_Anggota = pm.ID_Anggota\n" +
                    "INNER JOIN USER usr ON usr.ID_User = pm.ID_User\n" +
                    "WHERE pm.ID_Peminjaman LIKE ? OR agt.Nama_Anggota LIKE ?";

        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.setString(1, "%" + keyword + "%");
            st.setString(2, "%" + keyword + "%");

            ResultSet rs = st.executeQuery();

            while (rs.next()) {
                String idPeminjam = rs.getString("ID_Peminjaman");
                String tanggalPinjam = rs.getString("Tanggal_Peminjaman");
                String tanggalKembali = rs.getString("Tanggal_Pengembalian");
                String totalPinjam = rs.getString("Total_Pinjam");
                String namaAnggota = rs.getString("Nama_Anggota");
                String namaUser = rs.getString("Nama_User");

                Object[] rowData = {idPeminjam, tanggalPinjam, tanggalKembali, totalPinjam, namaAnggota, namaUser};
                model.addRow(rowData);
            }
        }
    } catch(Exception e){
        Logger.getLogger(MenuPeminjaman.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void paginationBuku() {
    btn_firts.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = 1;
            loadData();
        }
    });
}

```

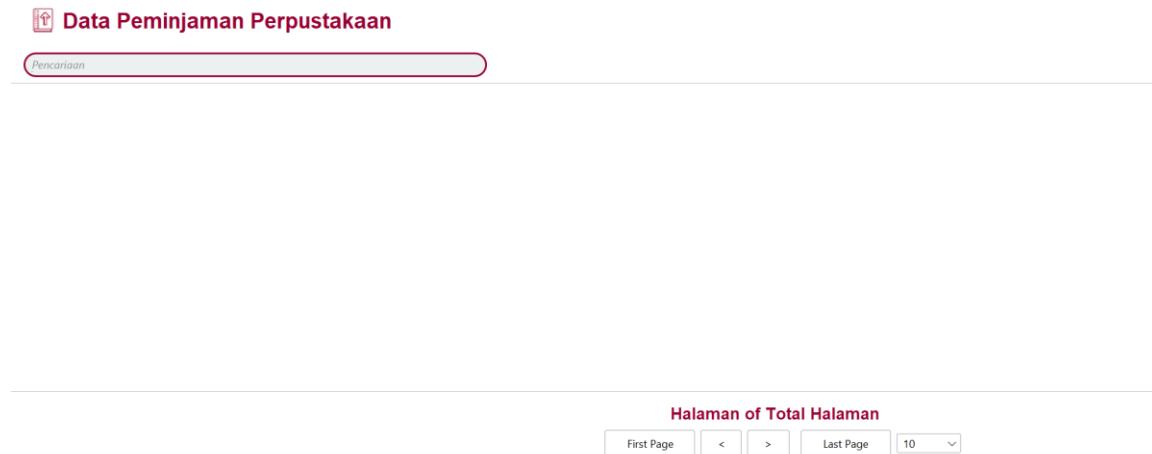
Penjelasan :

1. Enkapsulasi:
 - Variabel yang dinyatakan sebagai private seperti userID tidak bisa diakses secara langsung dari luar kelas.
 - Metode-metode yang digunakan untuk mengakses dan mengubah nilai variabel tersebut, seperti getData, insertData, dan deleteData, merupakan contoh implementasi dari enkapsulasi.
2. Pewarisan (Inheritance):
 - Kelas MenuPeminjaman memiliki pewarisan secara tidak langsung dari kelas javax.swing.JPanel.
 - Konstruktor kelas MenuPeminjaman memanggil konstruktor kelas induknya menggunakan super().
3. Polimorfisme:
 - Konsep polimorfisme dapat dilihat dalam penggunaan method overloading di mana terdapat beberapa metode dengan nama insertData() dan insertDataSementara().
4. Event Driven:
 - Kode tersebut menggunakan paradigma event-driven dalam penggunaan komponen GUI Swing. Misalnya, ketika tombol-tombol ditekan (ActionPerformed), metode-metode tertentu dipanggil untuk menangani event tersebut. Contohnya adalah btn_tambahActionPerformed, btn_simpanActionPerformed, dan lain-lain.
5. Exception Handling:
 - Exception handling digunakan untuk menangani kondisi error yang mungkin terjadi selama eksekusi program. Dalam kode tersebut, exception handling terlihat dalam blok try-catch di beberapa metode seperti insertData, insertDataSementara, updateData, dan lain-lain. Ketika terjadi kesalahan eksekusi, pesan error akan ditangkap dan ditampilkan melalui objek JOptionPane.
6. Fungsi dari setiap method dalam kelas MenuPeminjaman:
 - MenuPeminjaman: Konstruktor kelas yang menginisialisasi objek dan elemen antarmuka pengguna.
 - btn_tambahActionPerformed, btn_batalActionPerformed, btn_btlActionPerformed, btn_simpanActionPerformed, dan lain-lain: Metode-metode ini menangani event yang terjadi saat tombol-tombol di antarmuka pengguna ditekan.
 - loadData, loadDataSementara, searchData, dan lain-lain: Metode-metode ini digunakan untuk memuat, mencari, dan menampilkan data dari database ke dalam tabel pada antarmuka pengguna.

- insertData, insertDataDetail, insertDataSementara, dan lain-lain: Metode-metode ini bertanggung jawab untuk menyisipkan data ke dalam database, baik untuk tabel utama maupun tabel detail.
- resetForm, resetFormBuku: Metode-metode ini digunakan untuk mengosongkan formulir atau form input pada antarmuka pengguna.
- paginationBuku: Metode ini mengatur tindakan yang dilakukan ketika tombol navigasi halaman ditekan untuk mengatur tampilan data dalam beberapa halaman.
- calculateTotalPages, getTotalData, getIDKategori, getIDPenerbit, dan lain-lain: Metode-metode ini membantu dalam perhitungan dan pengambilan data yang dibutuhkan, seperti total halaman, total data, dan ID kategori atau penerbit dari database.

L. Halaman Data Peminjaman

Tampilan :



Kode :

```
package Kode;

import Koneksi.Koneksi;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.table.DefaultTableModel;

public class DataPeminjaman extends java.awt.Dialog {

    private int halamanSaatIni = 1;
    private int dataPerHalaman = 10;
    private int totalPages;

    private final Connection conn;

    private String idPeminjaman;
    private String tanggalPinjam;
    private String tanggalKembali;
    private String idAnggota;
    private String namaAnggota;
    private String idBuku;
    private String judulBuku;
    private String pengarangBuku;
    private String penerbitBuku;
    private String jumlahPinjam;

    public String getIdPeminjaman() {
        return idPeminjaman;
    }

    public String getTanggalPinjam() {
        return tanggalPinjam;
    }

    public String getTanggalKembali() {
        return tanggalKembali;
    }

    public String getIdAnggota() {
        return idAnggota;
    }

    public String getNamaAnggota() {
        return namaAnggota;
    }

    public String getIdBuku() {
        return idBuku;
    }

    public String getJudulBuku() {
        return judulBuku;
    }

    public String getPengarangBuku() {
        return pengarangBuku;
    }

    public String getPenerbitBuku() {
        return penerbitBuku;
    }

    public String getJumlahPinjam() {
        return jumlahPinjam;
    }

    public DataPeminjaman(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
        conn = Koneksi.getConnection();
        setTabelModel();
        loadData();
        paginationAnggota();
    }
}
```

```
private void closeDialog(java.awt.event.WindowEvent evt) {
    setVisible(false);
    dispose();
}

private void tbl_dataMouseClicked(java.awt.event.MouseEvent evt) {
    pilihData();
}

private void txt_cariKeyReleased(java.awt.event.KeyEvent evt) {
    searchData();
}

public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            DataPeminjaman dialog = new DataPeminjaman(new java.awt.Frame(), true);
            dialog.addWindowListener(new java.awt.event.WindowAdapter() {
                public void windowClosing(java.awt.event.WindowEvent e) {
                    System.exit(0);
                }
            });
            dialog.setVisible(true);
        }
    });
}
```

```

private void paginationAnggota() {
    btn_firts.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = 1;
            loadData();
        }
    });
    btn_previous.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if(halamanSaatIni > 1){
                }
                halamanSaatIni--;
                loadData();
            }
        });
    btn_next.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (halamanSaatIni < totalPages){
                halamanSaatIni++;
                loadData();
            }
        }
    });
    btn_last.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = totalPages;
            loadData();
        }
    });
    jcbx_data.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            dataPerHalaman =
Integer.parseInt(jcbx_data.getSelectedItem().toString());
            loadData();
        }
    });
}

private void calculateTotalPages(){
    int totalData = getTotalData();
    totalPages = (int) Math.ceil((double) totalData / dataPerHalaman);
}

```

```

private int getTotalData() {
    int totalData = 0;

    try{
        String sql = "SELECT COUNT(*) AS total FROM peminjaman";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            ResultSet rs = st.executeQuery();
            if(rs.next()){
                totalData = rs.getInt("total");
            }
        }
    } catch(Exception e){
        Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
    }
    return totalData;
}

private void loadData(){
    calculateTotalPages();
    int totalData = getTotalData();
    jtbl_halaman.setText(String.valueOf("Halaman" + halamanSaatIni + " dari Total Data" +
totalData));
    int startIndex = (halamanSaatIni - 1) * dataPerHalaman;
    getData(startIndex, dataPerHalaman, (DefaultTableModel) tbl_data.getModel());
}

private void getData(int startIndex, int entriesPage, DefaultTableModel model) {
    model.setRowCount(0);

    try {
        String sql = "SELECT * FROM "
+ "detail_peminjaman pmd "
+ "INNER JOIN peminjaman pm ON pm.ID_Peminjaman = pmd.ID_Peminjaman "
+ "INNER JOIN anggota agt ON agt.ID_Anggota = pm.ID_Anggota "
+ "INNER JOIN buku bk ON bk.ID_Buku = pmd.ID_Buku "
+ "INNER JOIN penerbit pnb ON pnb.ID_Penerbit = bk.ID_Penerbit "
+ "WHERE Status_Peminjaman = 'Sedang dipinjam' LIMIT ?, ?";

        try (PreparedStatement st = conn.prepareStatement(sql)) {
            st.setInt(1, startIndex);
            st.setInt(2, entriesPage);
            ResultSet rs = st.executeQuery();

            while (rs.next()) {
                String idPeminjam = rs.getString("ID_Peminjaman");
                String tanggalPinjam = rs.getString("Tanggal_Peminjaman");
                String tanggalKembali = rs.getString("Tanggal_Pengembalian");
                String idAnggota = rs.getString("ID_Anggota");
                String namaAnggota = rs.getString("Nama_Anggota");
                String idBuku = rs.getString("ID_Buku");
                String judulBuku = rs.getString("Judul_Buku");
                String pengarangBuku = rs.getString("Pengarang_Buku");
                String penerbitBuku = rs.getString("Nama_Penerbit");
                String jumlahPinjam = rs.getString("Jumlah_Pinjam");

                Object[] rowData = {idPeminjam, tanggalPinjam, tanggalKembali, idAnggota,
                    namaAnggota, idBuku, judulBuku, pengarangBuku, penerbitBuku, jumlahPinjam};
                model.addRow(rowData);
            }
        }
    } catch (SQLException e) {
        Logger.getLogger(MenuAnggota.class.getName()).log(Level.SEVERE, null, e);
    }
}

private void setTabelModel(){
    DefaultTableModel model = (DefaultTableModel) tbl_data.getModel();
    model.addColumn("ID");
    model.addColumn("TANGGAL PINJAM");
    model.addColumn("TANGGAL KEMBALI");
    model.addColumn("ID ANGGOTA");
    model.addColumn("NAMA NAGGOTA");
    model.addColumn("ID BUKU");
    model.addColumn("JUDUL");
    model.addColumn("PENGARANG");
    model.addColumn("PENERBIT");
    model.addColumn("JUMLAH");
}

```

```

private void searchData() {
    String keyword = txt_cari.getText();

    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.setRowCount(0);

    try {
        String sql;

        if(!keyWord.isEmpty()){
            sql = "SELECT * FROM "
                + "detail_peminjaman pmd "
                + "INNER JOIN peminjaman pm ON pm.ID_Peminjaman = pmd.ID_Peminjaman "
                + "INNER JOIN anggota agt ON agt.ID_Anggota = pm.ID_Anggota "
                + "INNER JOIN buku bk ON bk.ID_Buku = pmd.ID_Buku "
                + "INNER JOIN penerbit pnb ON pnb.ID_Penerbit = bk.ID_Penerbit "
                + "WHERE Status_Peminjaman = 'Sedang dipinjam' AND (pm.ID_Peminjaman LIKE ? OR
                agt>Nama_Anggota LIKE ?)";
        }

        else{
            sql = "SELECT * FROM "
                + "detail_peminjaman pmd "
                + "INNER JOIN peminjaman pm ON pm.ID_Peminjaman = pmd.ID_Peminjaman "
                + "INNER JOIN anggota agt ON agt.ID_Anggota = pm.ID_Anggota "
                + "INNER JOIN buku bk ON bk.ID_Buku = pmd.ID_Buku "
                + "INNER JOIN penerbit pnb ON pnb.ID_Penerbit = bk.ID_Penerbit "
                + "WHERE Status_Peminjaman = 'Sedang dipinjam'";
        }
    }

    try (PreparedStatement st = conn.prepareStatement(sql)) {
        if(!keyWord.isEmpty()){
            st.setString(1, "%" + keyword + "%");
            st.setString(2, "%" + keyword + "%");
        }

        ResultSet rs = st.executeQuery();

        while (rs.next()) {
            String idPeminjam = rs.getString("ID_Peminjaman");
            String tanggalPinjam = rs.getString("Tanggal_Peminjaman");
            String tanggalKembali = rs.getString("Tanggal_Pengembalian");
            String idAnggota = rs.getString("ID_Anggota");
            String namaAnggota = rs.getString("Nama_Anggota");
            String idBuku = rs.getString("ID_Buku");
            String judulBuku = rs.getString("Judul_Buku");
            String pengarangBuku = rs.getString("Pengarang_Buku");
            String penerbitBuku = rs.getString("Nama_Penerbit");
            String jumlahPinjam = rs.getString("Jumlah_Pinjam");

            Object[] rowData = {idPeminjam, tanggalPinjam, tanggalKembali, idAnggota,
                               namaAnggota, idBuku, judulBuku, pengarangBuku, penerbitBuku, jumlahPinjam};
            model.addRow(rowData);
        }
    }

    catch (SQLException e) {
        e.printStackTrace();
    }
}

private void pilihData(){
    int row = tbl_data.getSelectedRow();

    idPeminjaman = tbl_data.getValueAt(row, 0).toString();
    tanggalPinjam = tbl_data.getValueAt(row, 1).toString();
    tanggalKembali = tbl_data.getValueAt(row, 2).toString();
    idAnggota = tbl_data.getValueAt(row, 3).toString();
    namaAnggota = tbl_data.getValueAt(row, 4).toString();
    idBuku = tbl_data.getValueAt(row, 5).toString();
    judulBuku = tbl_data.getValueAt(row, 6).toString();
    pengarangBuku = tbl_data.getValueAt(row, 7).toString();
    penerbitBuku = tbl_data.getValueAt(row, 8).toString();
    jumlahPinjam = tbl_data.getValueAt(row, 9).toString();

    tbl_data.setVisible(false);
    dispose();
}
}

```

Penjelasan :

1. Enkapsulasi:

- Terdapat penggunaan private fields seperti idPeminjaman, tanggalPinjam, tanggalKembali, dll., yang hanya dapat diakses melalui public methods yang disediakan. Misalnya, getIdPeminjaman(), getTanggalPinjam(), dan seterusnya.
- Data dalam objek DataPeminjaman terlindungi karena hanya dapat diakses melalui getter methods dan tidak dapat diubah langsung dari luar kelas.

2. Polimorfisme:

- Ada penggunaan polimorfisme melalui method overriding, yaitu ketika method tertentu seperti actionPerformed() dan keyReleased() diimplementasikan kembali

dalam kelas DataPeminjaman untuk menyesuaikan perilaku yang diinginkan dalam konteks kelas tersebut.

3. Event-Driven Programming:

- Kode ini menggunakan pendekatan pemrograman event-driven di mana tindakan pengguna, seperti mengklik tombol, mengetik di kotak teks, atau memilih opsi dari dropdown, memicu event yang ditangani oleh event listeners. Contohnya adalah event listeners yang terpasang pada tombol dan kotak teks untuk menangani tindakan seperti mengklik atau mengetik.

4. Exception Handling:

- Terdapat penggunaan exception handling di dalam metode getTotalData() dan getData(). Jika terjadi kesalahan dalam menjalankan kueri SQL atau mengambil data dari result set, penanganan exception dilakukan untuk memberikan informasi yang jelas tentang kesalahan yang terjadi.

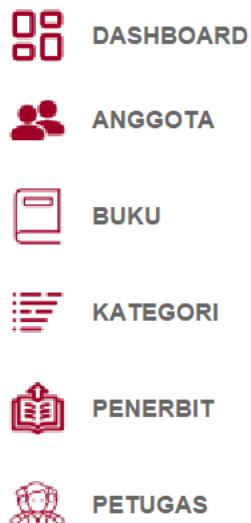
5. Fungsi dari Setiap Method:

- closeDialog(): Menutup dialog.
- tbl_dataMouseClicked(): Dipanggil ketika pengguna mengklik baris di tabel data.
- txt_cariKeyReleased(): Dipanggil ketika pengguna melepaskan tombol pada kotak pencarian, untuk mencari data berdasarkan kata kunci yang dimasukkan.
- paginationAnggota(): Metode yang mengatur event listeners untuk tombol navigasi halaman.
- calculateTotalPages(): Menghitung total halaman berdasarkan jumlah total data dan data per halaman.
- getTotalData(): Mengambil jumlah total data yang ada di tabel.
- loadData(): Memuat data sesuai dengan halaman saat ini dan data per halaman yang dipilih.
- getData(): Mengambil data dari database sesuai dengan halaman dan jumlah data per halaman yang ditentukan.
- setTabelModel(): Mengatur model tabel dengan kolom-kolom yang diperlukan.
- searchData(): Mencari data berdasarkan kata kunci yang dimasukkan dalam kotak pencarian.
- pilihData(): Menangani pemilihan data dari tabel untuk digunakan dalam proses lain dalam aplikasi.

M. Menu Master

Tampilan :

MASTER DATA



Kode :

```
● ● ●

package Main;

import Kode.MenuAnggota;
import Kode.MenuBuku;
import Kode.MenuDashboard;
import Kode.MenuKategori;
import Kode.MenuPenerbit;
import Kode.MenuPetugas;
import java.awt.Color;
import javax.swing.JPanel;

public class MenuMaster extends javax.swing.JPanel
{
    private MenuUtama menuUtama;

    public MenuMaster(MenuUtama menuUtama) {
        initComponents();
        this.menuUtama = menuUtama;
    }
}
```

```
● ● ●
private void btn_dashboardMouseClicked(java.awt.event.MouseEvent evt) {
    jpn_btndhsbrd.setBackground(new Color(240, 240, 240));
    jpn_lne1.setBackground(new Color(128, 0, 0));
    JPanel jp = menuutama.getPanelUtama();
    jp.removeAll();
    jp.add(new Menubashboard());
    jp.repaint();
    jp.revalidate();
}

private void btn_dashboardMouseEntered(java.awt.event.MouseEvent evt) {
    jpn_btndhsbrd.setBackground(new Color(250, 250, 250));
    jpn_lne1.setBackground(new Color(128, 0, 0));
}

private void btn_dashboardMouseExited(java.awt.event.MouseEvent evt) {
    jpn_btndhsbrd.setBackground(new Color(255, 255, 255));
    jpn_lne1.setBackground(new Color(255, 255, 255));
}

private void btn_anggotaMouseClicked(java.awt.event.MouseEvent evt) {
    jpn_btnggota.setBackground(new Color(250, 250, 250));
    jpn_lne.setBackground(new Color(128, 0, 0));
    JPanel jp = menuutama.getPanelUtama();
    jp.removeAll();
    jp.add(new Menuanggota());
    jp.repaint();
    jp.revalidate();
}

private void btn_anggotaMouseEntered(java.awt.event.MouseEvent evt) {
    jpn_btnggota.setBackground(new Color(250, 250, 250));
    jpn_lne.setBackground(new Color(128, 0, 0));
}

private void btn_anggotaMouseExited(java.awt.event.MouseEvent evt) {
    jpn_btnggota.setBackground(new Color(255, 255, 255));
    jpn_lne.setBackground(new Color(255, 255, 255));
}

private void btn_bukuMouseClicked(java.awt.event.MouseEvent evt) {
    btn_buku.setBackground(new Color(240, 240, 240));
    jpn_line5.setBackground(new Color(128, 0, 0));
    JPanel jp = menuutama.getPanelUtama();
    jp.removeAll();
    jp.add(new Menubuku());
    jp.repaint();
    jp.revalidate();
}

private void btn_bukuMouseEntered(java.awt.event.MouseEvent evt) {
    jpn_btnbuku.setBackground(new Color(250, 250, 250));
    jpn_lne5.setBackground(new Color(128, 0, 0));
}

private void btn_bukuMouseExited(java.awt.event.MouseEvent evt) {
    jpn_btnbuku.setBackground(new Color(255, 255, 255));
    jpn_lne5.setBackground(new Color(255, 255, 255));
}

private void btn_kategoriMouseClicked(java.awt.event.MouseEvent evt) {
    jpn_btncategori.setBackground(new Color(250, 250, 250));
    jpn_lne11.setBackground(new Color(128, 0, 0));
    JPanel jp = menuutama.getPanelUtama();
    jp.removeAll();
    jp.add(new Menukategori());
    jp.repaint();
    jp.revalidate();
}

private void btn_kategoriMouseEntered(java.awt.event.MouseEvent evt) {
    jpn_btncategori.setBackground(new Color(250, 250, 250));
    jpn_lne11.setBackground(new Color(128, 0, 0));
}

private void btn_kategoriMouseExited(java.awt.event.MouseEvent evt) {
    jpn_btncategori.setBackground(new Color(255, 255, 255));
    jpn_lne11.setBackground(new Color(255, 255, 255));
}

private void btn_penerbitMouseClicked(java.awt.event.MouseEvent evt) {
    jpn_btnpenerbit.setBackground(new Color(250, 250, 250));
    jpn_lne12.setBackground(new Color(128, 0, 0));
    JPanel jp = menuutama.getPanelUtama();
    jp.removeAll();
    jp.add(new Menupenerbit());
    jp.repaint();
    jp.revalidate();
}

private void btn_penerbitMouseEntered(java.awt.event.MouseEvent evt) {
    jpn_btnpenerbit.setBackground(new Color(250, 250, 250));
    jpn_lne12.setBackground(new Color(128, 0, 0));
}

private void btn_penerbitMouseExited(java.awt.event.MouseEvent evt) {
    jpn_btnpenerbit.setBackground(new Color(255, 255, 255));
    jpn_lne12.setBackground(new Color(255, 255, 255));
}

private void btn_petugasMouseClicked(java.awt.event.MouseEvent evt) {
    jpn_btntpgs.setBackground(new Color(240, 240, 240));
    jpn_lne6.setBackground(new Color(128, 0, 0));
    JPanel jp = menuutama.getPanelUtama();
    jp.removeAll();
    jp.add(new Menupetugas());
    jp.repaint();
    jp.revalidate();
}

private void btn_petugasMouseEntered(java.awt.event.MouseEvent evt) {
    jpn_btntpgs.setBackground(new Color(250, 250, 250));
    jpn_lne6.setBackground(new Color(128, 0, 0));
}

private void btn_petugasMouseExited(java.awt.event.MouseEvent evt) {
    jpn_btntpgs.setBackground(new Color(255, 255, 255));
    jpn_lne6.setBackground(new Color(255, 255, 255));
}
```

Penjelasan :

1. Enkapsulasi:

- penggunaan kelas MenuMaster bersama dengan metode-metodenya seperti btn_dashboardMouseClicked, btn_anggotaMouseClicked, dll., merupakan contoh dari enkapsulasi. Data (variabel) dan metode-metode tersebut berada dalam satu wadah yang sama.

2. Polimorfisme:

- Dalam kode tersebut, terdapat polimorfisme dalam penggunaan metode-metode yang diaktifkan oleh berbagai event mouse, seperti btn_dashboardMouseClicked, btn_anggotaMouseClicked, dll. Meskipun metode-metode ini memiliki nama yang sama, perilaku mereka berbeda tergantung pada tombol mana yang diklik.

3. Event-driven:

- Dalam kode tersebut, setiap metode mengatur perilaku objek (panel) terkait dengan event tertentu, misalnya, btn_dashboardMouseClicked dipanggil ketika tombol dashboard diklik.

4. Fungsi setiap method dalam kode tersebut adalah sebagai berikut:

- Setiap metode yang dimulai dengan btn_ dihubungkan dengan event dari tombol-tombol di antarmuka pengguna (GUI).
- Setiap metode yang dimulai dengan jpn_ adalah metode untuk mengatur tampilan visual seperti perubahan warna panel atau garis sebagai umpan balik terhadap interaksi pengguna.
- Metode removeAll(), add(), repaint(), dan revalidate() digunakan untuk memanipulasi panel utama (JPanel) tempat konten baru ditampilkan ketika pengguna berinteraksi dengan tombol-tombol di menu master.

N. Menu Transaksi

Tampilan :



Kode :

```
package Main;

import Kode.MenuPeminjaman;
import Kode.MenuPengembalian;
import java.awt.Color;
import javax.swing.JPanel;

public class MenuTransaksi extends javax.swing.JPanel {
    private MenuUtama menuUtama;

    public MenuTransaksi(MenuUtama menuUtama) {
        initComponents();
        this.menuUtama = menuUtama;
    }
}
```

```
private void btn_peminjamanMouseClicked(java.awt.event.MouseEvent evt) {
    jpn_btnpeminjaman.setBackground(new Color(240, 240, 240));
    jpn_line7.setBackground(new Color(128, 0, 0));

    JPanel jp = menuUtama.getPanelUtama();
    jp.removeAll();
    jp.add(new MenuPeminjaman(menuUtama.getUserID()));
    jp.repaint();
    jp.revalidate();
}

private void btn_peminjamanMouseEntered(java.awt.event.MouseEvent evt) {
    jpn_btnpeminjaman.setBackground(new Color(250, 250, 250));
    jpn_line7.setBackground(new Color(128, 0, 0));
}

private void btn_peminjamanMouseExited(java.awt.event.MouseEvent evt) {
    jpn_btnpeminjaman.setBackground(new Color(255, 255, 255));
    jpn_line7.setBackground(new Color(255, 255, 255));
}

private void btn_pengembalianMouseClicked(java.awt.event.MouseEvent evt) {
    btn_pengembalian.setBackground(new Color(240, 240, 240));
    jpn_line8.setBackground(new Color(128, 0, 0));

    JPanel jp = menuUtama.getPanelUtama();
    jp.removeAll();
    jp.add(new MenuPengembalian(menuUtama.getUserID()));
    jp.repaint();
    jp.revalidate();
}

private void btn_pengembalianMouseEntered(java.awt.event.MouseEvent evt) {
    jpn_btnpengembalian.setBackground(new Color(250, 250, 250));
    jpn_line8.setBackground(new Color(128, 0, 0));
}

private void btn_pengembalianMouseExited(java.awt.event.MouseEvent evt) {
    jpn_btnpengembalian.setBackground(new Color(255, 255, 255));
    jpn_line8.setBackground(new Color(255, 255, 255));
}
```

Penjelasan :

1. Enkapsulasi:

- Dalam kode kelas MenuTransaksi menggabungkan panel dan logika pengaturan tampilan ke dalam satu kesatuan yang terorganisir.

2. Polimorfisme:

- Terdapat polimorfisme dalam penggunaan metode-metode yang dihubungkan dengan event mouse, seperti btn_peminjamanMouseClicked, btn_pengembalianMouseClicked, dll. Meskipun metode-metode ini memiliki nama yang sama, perilaku mereka berbeda tergantung pada tombol mana yang diklik.

3. Pemrograman Berbasis Acara (Event-driven):

- Seperti sebelumnya, kode ini menggunakan pemrograman berbasis acara di mana metode-metode dipanggil ketika ada interaksi dengan elemen GUI, seperti klik mouse. Misalnya, metode btn_peminjamanMouseClicked akan dipanggil ketika tombol peminjaman diklik.

4. Fungsi setiap method dalam kode tersebut adalah sebagai berikut:

- Setiap metode yang dimulai dengan btn_ dan jpn_ bertanggung jawab untuk menangani event mouse (klik, masuk, keluar) pada tombol-tombol dan panel yang ditampilkan di antarmuka pengguna.
- Metode removeAll(), add(), repaint(), dan revalidate() digunakan untuk memanipulasi panel utama (JPanel) tempat konten baru ditampilkan ketika pengguna berinteraksi dengan tombol-tombol pada menu transaksi.
- Metode btn_peminjamanMouseClicked dan btn_pengembalianMouseClicked diaktifkan saat pengguna mengklik tombol peminjaman dan pengembalian, masing-masing. Mereka mengganti konten panel utama dengan panel untuk menu peminjaman atau pengembalian.

O. Menu Laporan

Tampilan :



Kode :

```
package Main;

import Kode.LaporanPeminjaman;
import Kode.MenuAnggota;
import java.awt.Color;
import javax.swing.JPanel;

public class MenuLaporan extends javax.swing.JPanel {
    private MenuUtama menuUtama;

    public MenuLaporan(MenuUtama menuUtama) {
        this.menuUtama = menuUtama;
        initComponents();
    }
}
```

```
private void btn_lappeminjamanMouseClicked(java.awt.event.MouseEvent evt) {
    jpn_btnlappeminjaman.setBackground(new Color(240, 240, 240));
    jpn_line9.setBackground(new Color(128, 0, 0));

    JPanel jp = menuUtama.getPanelUtama();

    jp.removeAll();
    jp.add(new LaporanPeminjaman());
    jp.repaint();
    jp.revalidate();
}

private void btn_lappeminjamanMouseEntered(java.awt.event.MouseEvent evt) {
    jpn_btnlappeminjaman.setBackground(new Color(250, 250, 250));
    jpn_line9.setBackground(new Color(128, 0, 0));
}

private void btn_lappeminjamanMouseExited(java.awt.event.MouseEvent evt) {
    jpn_btnlappeminjaman.setBackground(new Color(255, 255, 255));
    jpn_line9.setBackground(new Color(255, 255, 255));
}
```

Penjelasan :

1. Enkapsulasi:

- Atribut-atribut (variabel) kelas dideklarasikan sebagai private seperti jLabel9, jpn_btnlappeminjaman, jpn_line9, jpn_lappeminjaman, btn_lappeminjaman, yang artinya hanya dapat diakses secara langsung oleh kelas itu sendiri.
- Metode-metode yang mengoperasikan atribut tersebut juga didefinisikan dalam kelas yang sama, seperti initComponents(), btn_lappeminjamanMouseClicked(), btn_lappeminjamanMouseEntered(), btn_lappeminjamanMouseExited().

2. Event-driven:

- Kode tersebut menggunakan pendekatan event-driven untuk menanggapi tindakan pengguna. Ini terlihat dari penanganan event mouse yang dipicu oleh pengguna ketika tombol diklik, dimasukkan, atau dikeluarkan. Misalnya, btn_lappeminjamanMouseClicked(), btn_lappeminjamanMouseEntered(), dan btn_lappeminjamanMouseExited() adalah contoh metode yang dipanggil ketika pengguna melakukan tindakan mouse tertentu pada tombol "PEMINJAMAN".

3. Fungsi dari setiap method:

- `initComponents()`: Metode ini digunakan untuk menginisialisasi semua komponen GUI yang diperlukan, seperti label, panel, dan lainnya. Ini biasanya dihasilkan oleh editor GUI seperti NetBeans IDE.
- `btn_lappeminjamanMouseClicked()`: Metode ini dipanggil ketika tombol "PEMINJAMAN" diklik. Ketika tombol ini diklik, latar belakang panel berubah warna, dan panel utama diganti dengan panel yang menampilkan laporan peminjaman.
- `btn_lappeminjamanMouseEntered()`: Metode ini dipanggil ketika kurSOR mouse masuk ke area tombol "PEMINJAMAN". Ini mengubah tampilan tombol untuk memberi umpan balik visual kepada pengguna.
- `btn_lappeminjamanMouseExited()`: Metode ini dipanggil ketika kurSOR mouse meninggalkan area tombol "PEMINJAMAN". Ini mengembalikan tampilan tombol ke kondisi semula setelah kurSOR mouse meninggalkan tombol.

P. Halaman Laporan Peminjaman

Tampilan :

Data Laporan Peminjaman Buku

LAPORAN > PEMINJAMAN

Pencarian

TANGGAL MULAI

Masukkan Tanggal bergabung

TANGGAL SAMPAI

Masukkan Tanggal bergabung

TAMPILKAN

BATAL

CETAK

Halaman of Total Halaman

First Page < > Last Page 10

Kode :

```
● ● ●

package Kode;

import Koneksi.Koneksi;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
import java.util.HashMap;
import javax.swing.table.DefaultTableModel;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.view.JasperViewer;

public class LaporanPeminjaman extends javax.swing.JPanel {

    private int halamanSaatIni = 1;
    private int dataPerHalaman = 10;
    private int totalPages;

    private final Connection conn;

    private SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-
dd");
    private Date tanggalMulai;
    private Date tanggalAkhir;

    private String setIDAnggota;

    public LaporanPeminjaman() {
        JTextField jtxt_id = new JTextField();
        initComponents();

        conn = Koneksi.getConnection();
        setTabelModel();
        paginationAnggota();
        actionButton();
    }

}
```

```

    private void loadData(){
        calculateTotalPages();
        int totalData = getTotalData();
        jlbl_halaman.setText(String.valueOf("Halaman " + halamanSaatIni + " dari Total Data " +
        totalData));
    }

    int startIndex = (halamanSaatIni - 1) * dataPerHalaman;
    getData(startIndex, dataPerHalaman, (DefaultTableModel) tbl_data.getModel());
}

private void showPanel(){
    jpn_main.removeAll();
    jpn_main.add(new LaporanPeminjaman());
    jpn_main.repaint();
    jpn_main.revalidate();
}

private void setTableModel(){
    DefaultTableModel model = (DefaultTableModel) tbl_data.getModel();
    model.addColumn("ID PEMINJAM");
    model.addColumn("ID ANGGOTA");
    model.addColumn("NAMA ANGGOTA");
    model.addColumn("ID BUKU");
    model.addColumn("PINJAM");
    model.addColumn("KEMBALI");
    model.addColumn("DIKEMBALIKAN");
    model.addColumn("STATUS");
    model.addColumn("DENDA");
}

private void actionButton(){
    btn_tampilkan.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e){
            loadData();
        }
    });
    btn_batal.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e){
            showPanel();
            loadData();
        }
    });
    btn_print.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e){
            cetakLaporan();
        }
    });
    txt_cari.addKeyListener(new KeyAdapter(){
        @Override
        public void keyReleased(KeyEvent e) {
            searchData();
        }
    });
}

private void getData(int startIndex, int entriesPage, DefaultTableModel model) {
    model.setRowCount(0);
    String tanggalMulai = jtxt_tanggalmulai.getText();
    String tanggalAkhir = jtxt_tanggalakhir.getText();

    try{
        this.tanggalMulai = dateFormat.parse(tanggalMulai);
        this.tanggalAkhir = dateFormat.parse(tanggalAkhir);

        String sql = "SELECT "
            + "pmj.ID_Peminjaman, "
            + "agt.ID_Anggota, "
            + "agt.Nama_Anggota, "
            + "bk.ID_Buku, "
            + "pmj.Tanggal_Peminjaman, "
            + "pmj.Tanggal_Pengembalian, "
            + "png.Tanggal_Dikembalikan, "
            + "dpm.Status_Peminjaman, "
            + "dpn.Jumlah_Denda "
            + "FROM pengembalian png "
            + "INNER JOIN detail_pengembalian dpn ON dpn.ID_Pengembalian =
png.ID_Pengembalian "
            + "INNER JOIN pinjaman pmj ON pmj.ID_Peminjaman = png.ID_Peminjaman "
            + "INNER JOIN detail_peminjaman dpm ON dpm.ID_Peminjaman = pmj.ID_Peminjaman "
            + "INNER JOIN anggota agt ON agt.ID_Anggota = pmj.ID_Anggota "
            + "INNER JOIN buku bk ON bk.ID_Buku = dpm.ID_Buku "
            + "WHERE pmj.Tanggal_Peminjaman BETWEEN ? AND ? "
            + "AND dpm.Status_Peminjaman = 'Sudah Dikembalikan' "
            + "GROUP BY "
            + "bk.ID_Buku, "
            + "pmj.ID_Peminjaman, "
            + "png.ID_Pengembalian, "
            + "pmj.Tanggal_Peminjaman, "
            + "png.Tanggal_Dikembalikan "
            + "ORDER BY png.ID_Pengembalian ASC LIMIT ?, ?";

        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.setDate(1, new java.sql.Date(this.tanggalMulai.getTime()));
            st.setDate(2, new java.sql.Date(this.tanggalAkhir.getTime()));
            st.setInt(3, startIndex);
            st.setInt(4, entriesPage);
            ResultSet rs = st.executeQuery();

            while(rs.next()){
                String idPeminjaman = rs.getString("ID_Peminjaman");
                String idAnggota = rs.getString("ID_Anggota");
                String namaAnggota = rs.getString("Nama_Anggota");
                String idBuku = rs.getString("ID_Buku");
                String tanggalPinjam = rs.getString("Tanggal_Peminjaman");
                String tanggalKembali = rs.getString("Tanggal_Pengembalian");
                String tanggalDikembalikan = rs.getString("Tanggal_Dikembalikan");
                String statusPeminjaman = rs.getString("Status_Peminjaman");
                String jumlah_Denda = rs.getString("Jumlah_Denda");

                Object[] rowData = {idPeminjaman, idAnggota, namaAnggota, idBuku,
                tanggalPinjam, tanggalKembali, tanggalDikembalikan, statusPeminjaman,
                jumlah_Denda};
                model.addRow(rowData);
            }
        }
    } catch(Exception e){
        Logger.getLogger(LaporanPeminjaman.class.getName()).log(Level.SEVERE, null, e);
    }
}

```

```

private void searchData() {
    String keyWord = txt_cari.getText();
    DefaultTableModel model = (DefaultTableModel)tbl_data.getModel();
    model.setRowCount(0);
    try{
        String sql = "SELECT "
                    + "pmj.ID_Peminjaman, "
                    + "agt.ID_Anggota, "
                    + "agt.Nama_Anggota, "
                    + "bk.ID_Buku, "
                    + "pmj.Tanggal_Peminjaman, "
                    + "pmj.Tanggal_Pengembalian, "
                    + "png.Tanggal_Dikembalikan, "
                    + "dpm.Status_Peminjaman, "
                    + "dpn.Jumlah_Denda "
                    + "FROM pengembalian png "
                    + "INNER JOIN detail_pengembalian dpn ON dpn.ID_Pengembalian = "
                    + "INNER JOIN peminjaman pmj ON pmj.ID_Peminjaman = png.ID_Peminjaman "
                    + "INNER JOIN detail_peminjaman dpm ON dpm.ID_Peminjaman = pmj.ID_Peminjaman "
                    + "INNER JOIN anggota agt ON agt.ID_Anggota = pmj.ID_Anggota "
                    + "INNER JOIN buku bk ON bk.ID_Buku = dpm.ID_Buku "
                    + "WHERE (pmj.ID_Peminjaman LIKE ? OR agt>Nama_Anggota LIKE ? OR bk.ID_Buku "
                    + "AND pmj.Tanggal_Peminjaman BETWEEN ? AND ? AND dpm.Status_Peminjaman = "
                    + "'Sudah Dikembalikan' "
                    + "GROUP BY "
                    + "bk.ID_Buku, "
                    + "pmj.ID_Peminjaman, "
                    + "png.ID_Pengembalian, "
                    + "pmj.Tanggal_Peminjaman, "
                    + "pmj.Tanggal_Pengembalian, "
                    + "png.Tanggal_Dikembalikan ";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            st.setString(1, "%" + keyWord + "%");
            st.setString(2, "%" + keyWord + "%");
            st.setString(3, "%" + keyWord + "%");
            st.setString(4, dateFormat.format(tanggalMulai));
            st.setString(5, dateFormat.format(tanggalAkhir));
            ResultSet rs = st.executeQuery();
            while(rs.next()){
                String idPeminjaman = rs.getString("ID_Peminjaman");
                String idAnggota = rs.getString("ID_Anggota");
                String namaAnggota = rs.getString("Nama_Anggota");
                String idBuku = rs.getString("ID_Buku");
                String tanggalPinjam = rs.getString("Tanggal_Peminjaman");
                String tanggalKembali = rs.getString("Tanggal_Pengembalian");
                String tanggalDikembalikan = rs.getString("Tanggal_Dikembalikan");
                String statusPeminjaman = rs.getString("Status_Peminjaman");
                String jumlah_Denda = rs.getString("Jumlah_Denda");
                Object[] rowData = {idPeminjaman, idAnggota, namaAnggota, idBuku,
                    tanggalPinjam, tanggalKembali, tanggalDikembalikan, statusPeminjaman,
                    jumlah_Denda};
                model.addRow(rowData);
            }
        }
        catch(Exception e){
            Logger.getLogger(LaporanPeminjaman.class.getName()).log(Level.SEVERE, null, e);
        }
    }
}

private void paginationAnggotta() {
    btn_firts.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = 1;
            loadData();
        }
    });
    btn_previous.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
            if(halamanSaatIni > 1){
            }
            halamanSaatIni--;
            loadData();
        }
    });
    btn_next.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
            if (halamanSaatIni < totalPages){
                halamanSaatIni++;
                loadData();
            }
        }
    });
    btn_last.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
            halamanSaatIni = totalPages;
            loadData();
        }
    });
    jcbx_data.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
            dataPerHalaman = Integer.parseInt(jcbx_data.getSelectedItem().toString());
            halamanSaatIni = 1;
            loadData();
        }
    });
}

```

```

private void calculateTotalPages(){
    int totalData = getTotalData();
    totalPages = (int) Math.ceil((double) totalData / dataPerHalaman);
}

private int getTotalData() {
    int totalData = 0;

    try{
        String sql = "SELECT COUNT(*) AS total FROM pengembalian";
        try(PreparedStatement st = conn.prepareStatement(sql)){
            ResultSet rs = st.executeQuery();
            if(rs.next()){
                totalData = rs.getInt("total");
            }
        }
    } catch(Exception e){
        Logger.getLogger(LaporanPeminjaman.class.getName()).log(Level.SEVERE, null, e);
    }

    return totalData;
}

private void cetakLaporan() {
    String tanggalMulai = jtxt_tanggalmulai.getText();
    String tanggalAkhir = jtxt_tanggalkahir.getText();

    try {
        String reportPath = "src/Reports/LaporanPeminjaman.jasper";

        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        Date tglMulai = dateFormat.parse(tanggalMulai);
        Date tglAkhir = dateFormat.parse(tanggalAkhir);

        HashMap<String, Object> parameters = new HashMap<>();
        parameters.put("tanggalMulai", tglMulai);
        parameters.put("tanggalAkhir", tglAkhir);

        JasperPrint print = JasperFillManager.fillReport(reportPath, parameters, conn);
        JasperViewer viewer = new JasperViewer(print, false);
        viewer.setExtendedState(JasperViewer.MAXIMIZED_BOTH);
        viewer.setVisible(true);
    } catch(Exception e) {
        e.printStackTrace();
    }
}
}

```

Penjelasan :

1. Enkapsulasi:

- Dalam kode beberapa contoh enkapsulasi termasuk penggunaan variabel instance private seperti conn dan dateFormat, yang hanya dapat diakses dalam kelas itu sendiri.

2. Pewarisan:

- konsep pewarisan dapat dilihat dalam penggunaan metode extends untuk menggabungkan fungsionalitas dari kelas lain, seperti JPanel.

3. Polimorfisme:

- dalam kode terdapat dalam penanganan acara (event handling), di mana metode actionPerformed() dipanggil tergantung pada tindakan pengguna yang dipicu oleh tombol atau komponen lainnya.

4. Event-driven:

- pada metode actionPerformed() yang menambahkan pendengar acara untuk tombol-tombol.

5. Fungsi Setiap Method :

- `initComponents()`: Menginisialisasi komponen GUI.
- `loadData()`: Memuat data ke dalam tabel berdasarkan tanggal yang dipilih.
- `showPanel()`: Menampilkan panel.
- `setTabelModel()`: Mengatur model tabel.
- `actionButton()`: Menambahkan pendengar acara untuk tombol.
- `getData()`: Mengambil data dari database.
- `resetForm()`: Mereset formulir.
- `searchData()`: Mencari data berdasarkan kata kunci.
- `paginationAnggota()`: Menangani navigasi halaman untuk data anggota.
- `calculateTotalPages()`: Menghitung total halaman.
- `getTotalData()`: Menghitung total data yang tersedia.
- `cetakLaporan()`: Mencetak laporan berdasarkan tanggal yang dipilih.

Q. Tampilan Halaman Laporan Peminjaman Menggunakan Jasper

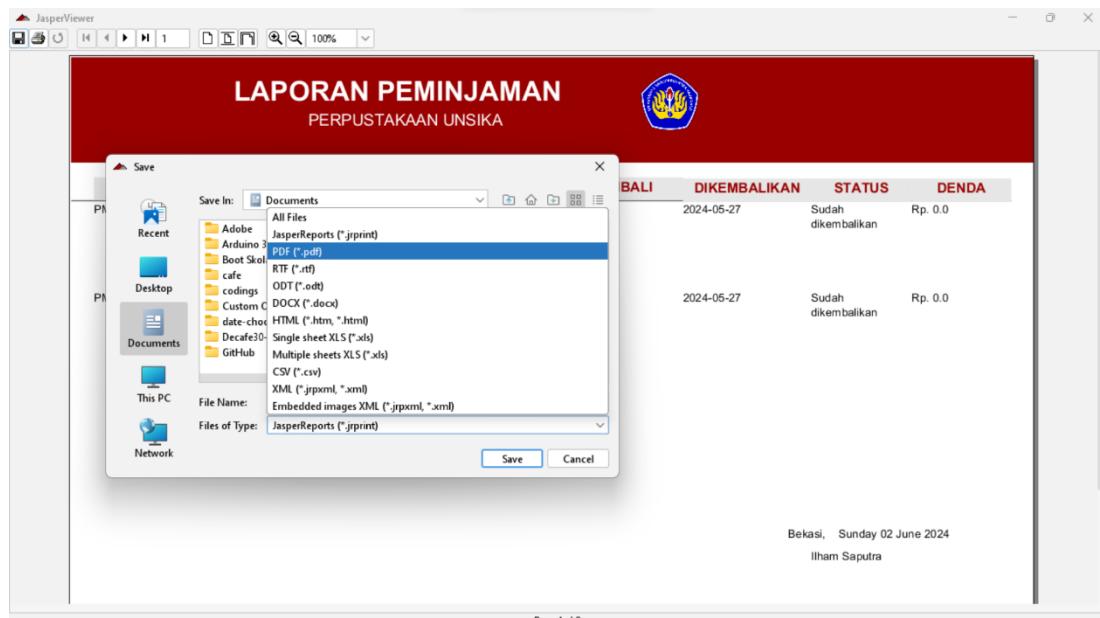
ID	ID	NAMA	ID BUKU	PINJAM	KEMBALI	DIKEMBALIKAN	STATUS	DENDA
PMJ2405001	AGT2310010	Nama10	BKU2405004	2024/05/27	2024/05/27	2024-05-27	Sudah dikembalikan	Rp. 0.0
PMJ2405002	AGT2310009	Nama44	BKU2405003	2024/05/27	2024/05/27	2024-05-27	Sudah dikembalikan	Rp. 0.0

Bekasi, Thursday 30 May
Ilham Saputra

Yang Ngedevelop

Pada Laporan Peminjaman ini, Menggunakan Jasper yaitu bisa diekstrak menjadi berbagai Macam Type File.

Contoh Gambar Ingin Mengekstrak Laporan Peminjaman “



BAB IV

KESIMPULAN

1. Tujuan Program:

Program perpustakaan bertujuan untuk mengelola data peminjaman dan pengembalian buku dalam suatu perpustakaan. Hal ini dilakukan dengan menyediakan fungsionalitas untuk melihat, mencari, dan mencetak laporan peminjaman buku berdasarkan rentang tanggal tertentu.

2. Fitur Utama:

- Memuat data peminjaman buku dari database berdasarkan rentang tanggal yang dipilih.
- Menampilkan data peminjaman buku dalam tabel, termasuk informasi seperti ID peminjaman, ID anggota, nama anggota, ID buku, tanggal pinjam, tanggal kembali, tanggal dikembalikan, status peminjaman, dan denda (jika ada).
- Memungkinkan pencarian data berdasarkan kata kunci.
- Mendukung navigasi halaman untuk menampilkan data dalam jumlah tertentu per halaman.
- Mencetak laporan peminjaman buku berdasarkan rentang tanggal yang dipilih.
- Menggunakan UI atau Tampilan yang Menarik

3. Konsep PBO yang Diterapkan:

- Menggunakan Class, Object, Methode dan Package.
- Program ini menerapkan konsep PBO seperti enkapsulasi, pewarisan, polimorfisme, dan penanganan acara (event-driven). Penggunaan kelas, metode, dan atribut membantu dalam mengatur dan mengelola logika program secara terstruktur.
- Enkapsulasi digunakan untuk menyembunyikan detail implementasi dari pengguna, sehingga meningkatkan keamanan dan modularitas kode.
- Pewarisan program ini mungkin memanfaatkan kelas turunan untuk memperluas fungsionalitas salah satunya menggunakan extend JPanel ataupun JForm.
- Polimorfisme tercermin dalam penggunaan antarmuka yang sama untuk menangani berbagai jenis peristiwa yang terjadi, seperti klik tombol atau perubahan nilai.
- Penanganan acara (event-driven) digunakan untuk menanggapi tindakan pengguna, seperti memuat data saat tombol "Tampilkan" ditekan.
- exception handling digunakan untuk menangani situasi yang tidak terduga atau kesalahan yang mungkin terjadi saat menjalankan kueri SQL atau operasi lainnya.
- Menggunakan Dasar – dasar GUI, dengan Property yang dicustom.
- Terkoneksi dengan Database menggunakan MySQL.
- Terintegrasi dengan Create, Read, Update, Edit dan Delete