GUI dan Database

1. Kompetensi

Setelah menempuh materi percobaan ini, mahasiswa mampu:

1. Menggunakan paradigma berorientasi objek untuk interaksi dengan database
2. Membuat Graphical User Interface (GUI)

2. Pendahuluan

Kali ini kita akan menggunakan paradigma berorientasi objek yang telah kita pelajari untuk membuat aplikasi yang melakukan manipulasi data di database melalui Graphical User Interface (GUI).

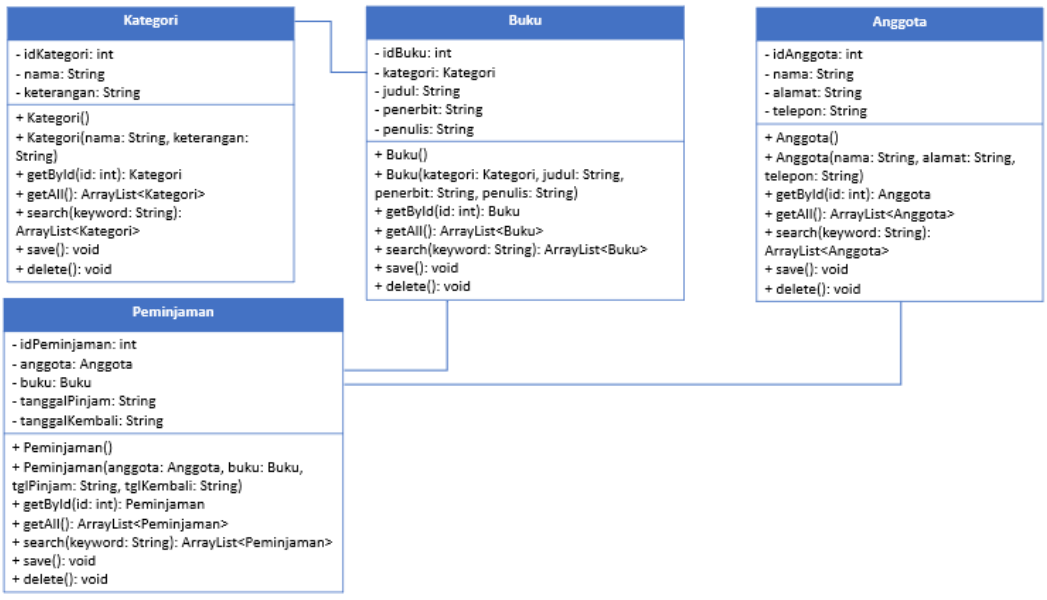
Secara umum, tahapan yang akan kita lakukan adalah sebagai berikut:

1. Membuat database yang berisi tabel-tabel yang diperlukan.
2. Membuat backend yang berisi class-class yang mewakili data yang ada pada database, dan class helper untuk melakukan eksekusi query database.
3. Membuat frontend sebagai antarmuka dengan pengguna. Frontend ini bisa berbasis teks (console), GUI, web, mobile, dan sebagainya.

Library yang digunakan untuk project ini antara lain:

1. JDBC (Java Data Base Connectivity), untuk melakukan interaksi ke database.
2. ArrayList, untuk menampung data hasil query ke database.
3. Swing, untuk membuat GUI.

Untuk percobaan, kita akan membuat sistem informasi Perpustakaan, yang memiliki data antara lain: Buku, Kategori, Anggota dan Peminjaman. Fitur yang ada pada aplikasi ini adalah anggota dapat melakukan peminjaman dan pengembalian buku. Berikut adalah class diagram untuk sistem informasi ini:



Dapat dilihat dari class diagram di atas, terdapat relasi antar class. Class Buku berelasi dengan Kategori dikarenakan terdapat atribut bertipe data Kategori di dalam class Buku. Demikian juga class Peminjaman yang berelasi dengan class Buku dan Anggota.

3. Percobaan

3.1 Percobaan 1 – Import Database

Import database Perpustakaan yang sudah disediakan.

3.2 Percobaan 2 – Mempersiapkan Project

1. Buat project baru, beri nama **Perpustakaan**.
2. Pada project explorer, klik kanan pada Libraries → Add Library, pilih MySQL JDBC Driver.
3. Jika tidak ada pilihan MySQL JDBC Driver:
    a. Buka link https://dev.mysql.com/downloads/connector/j/
    b. Pilih “Platform Independent”pada opsi Operating System
    c. Download ZIP Archive kemudian extract
    d. Pada project yang dibuat, klik kanan pada Libraries, kemudian add JAR/Folder… dan pilih file jar yang telah diextract sebelumnya
4. Buat package **frontend** dan **backend**. Cara membuat package adalah, pada project explorer, klik kanan pada Source Packages → New → Java Package, beri nama package nya (frontend, backend).

    Beberapa istilah dalam JDBC:
    a. **DriverManager**: class yang mengelola daftar driver JDBC dan bertanggung jawab memilih driver yang sesuai dan menyediakan objek Connection berdasarkan URL database.
    b. **Driver**: interface yang harus diimplementasikan oleh vendor database (MySQL, PostgreSQL, dll) untuk menangani mekanisme koneksi dan komunikasi antara Java dan database tersebut.
    c. **Connection**: interface yang mewakili sesi koneksi aktif ke database dan digunakan untuk membuat statement serta mengelola transaksi
    d. **Statement**: interface untuk mengeksekusi SQL command
    e. **PreparedStatement**: statement yang mendukung parameter
    f. **ResultSet**: interface yang merepresentasikan hasil query dan menyediakan metode untuk membaca data baris demi baris

3.3 Percobaan 3 – DBHelper

1. Pada package **backend**, buat class **DBHelper**.
2. Import java.sql.*
3. Berikut adalah kode dari class DBHelper. Sesuaikan nilai url, user, password, dan nama database dengan database yang Anda gunakan. DriverManager.getConnection() dipanggil untuk membangun koneksi ke database.

```
package backend;
import java.sql.*;

public class DBHelper {
    private static String url = "jdbc:mysql://localhost:3306/perpustakaan";
    private static String user = "root";
    private static String pass = "";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(url, user, pass);
    }
}
```

3.4 Percobaan 4 – Class Kategori

1. Pada package **backend**, buat class baru yaitu **Kategori**.
2. Import java.util.ArrayList dan java.sql.*

```
import java.util.ArrayList;
import java.sql.*;
```

3. Deklarasikan atribut sesuai field pada tabel kategori.

```
private int idKategori;
private String nama;
private String keterangan;
```

4. Buatlah setter getter untuk setiap atribut. Anda bisa gunakan fasilitas **Insert Code** pada NetBeans. Caranya adalah, klik kanan sembarang tempat di editor, pilih Insert Code, pilih Setter and Getter, lalu pilih atributnya.

5. Tambahkan method **getAll()** untuk mendapatkan data Kategori yang ada di database. Method executeQuery() akan mengembalikan data dari table Kategori. Selanjutnya setiap data dari ResultSet akan dibuatkan objek kategori baru yang ditampung oleh listKategori bertipe ArrayList<Kategori>.

```
public static ArrayList<Kategori> getAll(String keyword) {
    ArrayList<Kategori> listKategori = new ArrayList<>();

    String sql = "SELECT * FROM kategori "
               + "WHERE nama LIKE ? OR keterangan LIKE ?";

    try (Connection conn = DBHelper.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql)
    ){
        String likeKeyword = "%" + keyword + "%";
        ps.setString(1, likeKeyword);
        ps.setString(2, likeKeyword);

        try (ResultSet rs = ps.executeQuery()) {
            while (rs.next()) {
                Kategori kat = new Kategori();
                kat.setIdKategori(rs.getInt("id_kategori"));
                kat.setNama(rs.getString("nama"));
                kat.setKeterangan(rs.getString("keterangan"));

                listKategori.add(kat);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    return listKategori;
}
```

6. Buat method getById() untuk memperoleh data kategori berdasarkan id tertentu.

```
public static Kategori getById(int id) {
    Kategori kat = null;
    String sql = "SELECT * FROM kategori WHERE id_kategori = ?";

    try (
            Connection conn = DBHelper.getConnection();
            PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setInt(1, id);

        try (ResultSet rs = ps.executeQuery()) {
            if (rs.next()) {
                kat = new Kategori();
                kat.setIdKategori(rs.getInt("id_kategori"));
                kat.setNama(rs.getString("nama"));
                kat.setKeterangan(rs.getString("keterangan"));
            }
        }

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

    return kat;
}
```

7. Tambahkan method **save()**. Method ini memiliki dua fungsi, yaitu insert dan update. Jika idkategori bernilai 0 maka data yang diinputkan belum ada di database sehingga akan diinsert. Sedangkan jika idkategori tidak bernilai 0 maka akan dilakukan proses update.

```
public void save() {
    if (this.idKategori == 0) {
        String sql = "INSERT INTO kategori (nama, keterangan) VALUES (?, ?)";

        try (Connection conn = DBHelper.getConnection();
             PreparedStatement ps = conn.prepareStatement(
                     sql, Statement.RETURN_GENERATED_KEYS)) {

            ps.setString(1, this.nama);
            ps.setString(2, this.keterangan);
            ps.executeUpdate();

            try (ResultSet rs = ps.getGeneratedKeys()) {
                if (rs.next()) {
                    this.idKategori = rs.getInt(1);
                }
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }
    } else {
        String sql = "UPDATE kategori SET nama = ?, keterangan = ? WHERE id_kategori = ?";

        try (Connection conn = DBHelper.getConnection();
            PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setString(1, this.nama);
            ps.setString(2, this.keterangan);
            ps.setInt(3, this.idKategori);

            ps.executeUpdate();

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

8. Buat method **delete()** untuk melakukan operasi penghapusan data pada tabel kategori pada database.

```
public static boolean delete(int idKategori) {
    String sql = "DELETE FROM kategori WHERE id_kategori = ?";

    try (Connection conn = DBHelper.getConnection();
         PreparedStatement ps = conn.prepareStatement(sql)) {
        ps.setInt(1, idKategori);
        ps.executeUpdate();
        return true;

    } catch (SQLException e) {
        if (e.getErrorCode() == 1451) {
            System.out.println("Masih ada buku pada kategori ini");
        }

        e.printStackTrace();
        return false;
    }
}
```

3.5 Percobaan 5

Uji method-method yang sudah dibuat dengan memanggilnya dari frontend berbasis teks (console). Percobaan ini dapat di-skip jika Anda telah yakin bahwa method-method yang digunakan untuk manipulasi data sudah berfungsi dengan baik.

1. Pada package **frontend**, buat class **TestBackend**. Tambahkan import backend.*
2. Berikut kode lengkap untuk class TestBackend.

```
import backend.*;

public class TestBackend {
    public static void main(String[] args)
    {
        Kategori kat1 = new Kategori("Novel", "Koleksi buku novel");
        Kategori kat2 = new Kategori("Referensi", "Buku referensi ilmiah");
        Kategori kat3 = new Kategori("Komik", "Komik anak-anak");

        //test insert
        kat1.save();
        kat2.save();
        kat3.save();

        // test update
        kat2.setKeterangan("Koleksi buku referensi ilmiah");
        kat2.save();

        // test delete
        Kategori.delete(kat1.getIdKategori());

        // test get all
        for(Kategori k : Kategori.getAll())
        {
            System.out.println("Nama: " + k.getNama() + ", Ket: " + k.getKeterangan());
        }

        // test search
        for(Kategori k : Kategori.getAll("referensi"))
        {
            System.out.println("Nama: " + k.getNama() + ", Ket: " + k.getKeterangan());
        }
    }
}
```

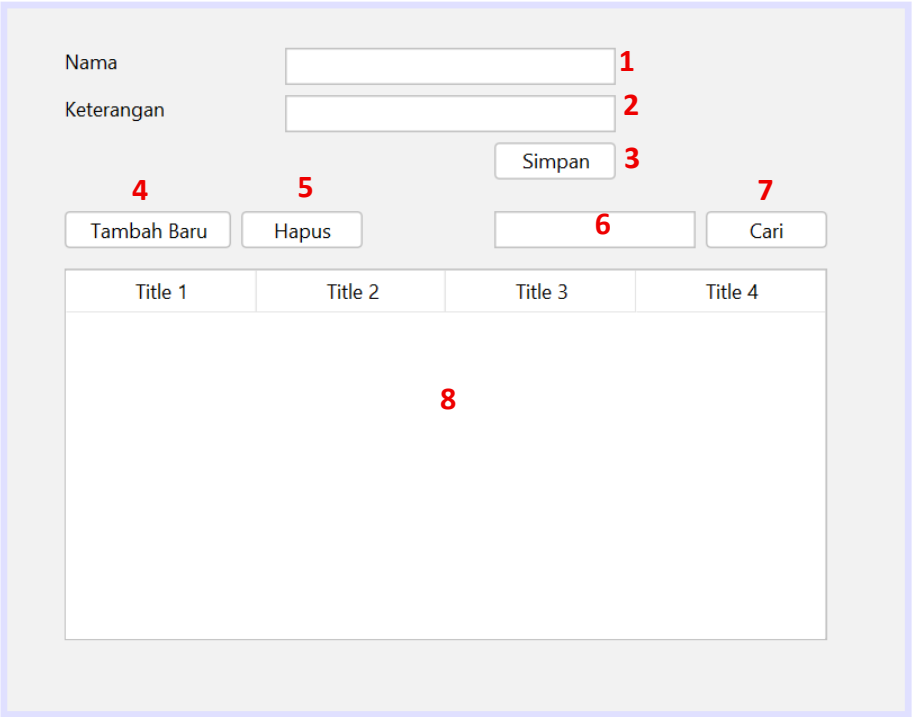
3. Jalankan TestBackend dengan klik kanan, Run File. Cocokkan outputnya:

```
run:
Nama: Referensi, Ket: Koleksi buku referensi ilmiah
Nama: Komik, Ket: Komik anak-anak
Nama: Referensi, Ket: Koleksi buku referensi ilmiah
BUILD SUCCESSFUL (total time: 1 second)
```

3.6 Percobaan 6

Pada percobaan ini kita akan membuat interface GUI untuk class **Kategori**.

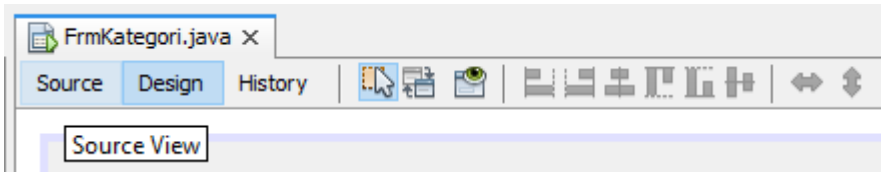
1. Pada package **frontend**, buat JFrame dengan nama FrmKategori. Caranya adalah, klik kanan pada package frontend → New →JFrame Form.
2. Susun form sehingga seperti Gambar berikut dengan cara *drag & drop* komponennya dari "Palette" di sisi kanan atas.



3. Perhatikan tabel di bawah ini kemudian atur properti setiap komponen dengan klik pada komponen, lalu ubah properties nya pada window **Properties** (umunya di sebelah kanan). Jika window belum terbuka, aktifkan melalui menu Window > IDE Tools > Properties. Properti *text* dan *enable* ada pada tab **Properties**, sedangkan Variable Name ada pada tab **Code**. Variable Name juga dapat diset dengan klik kanan pada komponen kemudian pilih "Change Variable Name"

Nomor	Tipe Komponen	Variable Name	Properties
1	JTextField	txtNama	text: kosong
2	JTextField	txtKeterangan	text: kosong
3	JButton	btnSimpan	text: Simpan
4	JButton	btnHapus	text: Hapus
5	JButton	btnTambahBaru	text: Tambah Baru
6	JTextField	txtCari	text: kosong
7	JButton	btnCari	text: Cari
8	JTable	tblKategori	

4. Edit kode program dengan klik Source tab



5. Tambahkan import backend.*, java.swing.JOptionPane, java.util.ArrayList, dan javax.swing.table.DefaultTableModel;

```
import backend.*;
import java.util.ArrayList;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

6. Tambahkan atribut idSelected pada FrmKategori untuk menyimpan nilai idKategori yang sedang dipilih.

```
public class FrmKategori extends javax.swing.JFrame {

    private int idSelected = 0;
```

7. Tambahkan method **kosongkanForm()** untuk mengosongkan isian textbox pada form dan mereset idSelected.

```
public void kosongkanForm(){
    txtNama.setText("");
    txtKeterangan.setText("");
    idSelected = 0;
}
```

8. Pada tab "Design", double klik pada **btnTambahBaru.** Netbeans akan secara otomatis menambahkan actionListener baru yaitu method btnTambahBaruActionPerformed(). Method ini akan dipanggil setiap kali btnTambahBaru diklik.

   Panggil method kosongkanForm() sehingga form siap digunakan untuk menginputkan data baru.

```
private void btnTambahBaruActionPerformed(java.awt.event.ActionEvent evt) {
    kosongkanForm();
}
```

9. Tambahkan method **tampilkanData()** untuk memperoleh semua data kategori dari database dengan method Kategori.getAll() kemudian menampilkannya ke JTable tblKategori.

   Keterangan:
   a. **Object**: class Object merupakan root class dari hierarki class. Setiap class merupakan turunan dari class Objek.
   b. **DefaultTableModel**: Class ini berfungsi untuk menyimpan nilai setiap cell yang akan ditampilkan dalam JTable. DefaultTableModel akan terdiri dari beberapa rowData. Setiap kolom dalam masing-masing rowData diisi dengan nilai dari listKategori yang diperoleh dari database.

Langkah:

a. Buat array of object baru bernama namaKolom untuk menyimpan nama-nama kolom
b. Buat objek bertipe DefaultTableModel baru bernama model yang akan menyimpan nilai setiap cell
c. Set namaKolom sebagai columnIdentifier(header) pada model
d. Dapatkan semua data kategori dari database menggunakan method getAll() berdasarkan keyword pada txtCari kemudian simpan dalam listKategori
e. Buat array of object bernama rowData dengan jumlah elemen 3 sesuai kolom yang akan ditampilkan
f. Untuk setiap objek kat dalam listKategori, buat rowData dengan informasi id, nama, dan keterangan yang sesuai kemudian tambahkan ke model
g. Set model sebagai nilai yang akan ditampilkan oleh tblKategori

```
public void tampilkanData(){
    String[] namaKolom = new String[] {"ID", "Nama", "Kategori"};

    DefaultTableModel model = new DefaultTableModel();
    model.setColumnIdentifiers(namaKolom);

    String keyword = txtCari.getText();
    ArrayList<Kategori> listKategori = Kategori.getAll(keyword);
    Object[] rowData = new Object[3];

    for (Kategori kat : listKategori) {
        rowData[0] = kat.getIdKategori();
        rowData[1] = kat.getNama();
        rowData[2] = kat.getKeterangan();

        model.addRow(rowData);
    }

    tblKategori.setModel(model);
}
```

10. Pada tab “Design”, double klik pada **btnCari.** Netbeans akan secara otomatis menambahkan actionListener baru berupa method btnCariActionPerformed(). Method ini akan dipanggil setiap kali btnCari diklik.

    Panggil method tampilkanData() sehingga tblKategori akan di-refresh setiap kali btnCari diklik.

```
private void btnCariActionPerformed(java.awt.event.ActionEvent evt) {
    tampilkanData();
}
```

11. Agar row data pada **tblKategori** dapat dipilih untuk diedit atau dihapus, maka perlu ditambahkan event mouse click pada **tblKategori**. Caranya, klik kanan pada **tblKategori**, pilih Events → Mouse → MouseClicked. Netbeans akan menambahkan actionListener berupa method tblKategoriMouseClicked(). Method ini akan dipanggil setiap ada klik pada tblKateogori.

Tambahkan kode berikut agar data yang diklik oleh user akan ditampilkan pada form di atas tabel.

```
private void tblKategoriMouseClicked(java.awt.event.MouseEvent evt) {
    DefaultTableModel model = (DefaultTableModel)tblKategori.getModel();
    int rowIndex = tblKategori.getSelectedRow();

    idSelected = (int) model.getValueAt(rowIndex, 0);
    txtNama.setText(model.getValueAt(rowIndex, 1).toString());
    txtKeterangan.setText(model.getValueAt(rowIndex, 2).toString());
}
```

Pengisian form secara lengkap dengan data yang berasal dari data table memungkinkan jika table menampilkan seluruh data. Jika tidak, tentukan idKategori kemudian dapatkan data lengkapnya menggunakan method getById().

12. Pada tab "Design", double click pada **btnSimpan**. Netbeans akan secara otomatis menambahkan actionListener baru berupa method btnSimpanActionPerformed(). Method ini akan dieksekusi setiap kali btnSimpan diklik.

    Langkah:
    a. Simpan data yang diinputkan user di text field ke variabel
    b. Cek apakah variabel untuk nama dan keterangan tidak kosong
    c. Jika data nama dan keterangan kosong, munculkan warning dengan JOptionPane
    d. Jika nama dan keterangan tidak kosong, dilakukan pembuatan objek kategori baru berdasarkan data tersebut.
    e. Lakukan pemanggilan method save() untuk menyimpan data kategori ke database. Method save() sudah secara otomatis menentukan apakah data akan di-insert atau di-update.
    f. Panggil method kosongkanForm() dan tampilkanData() untuk mengosongkan form dan memperbarui isi table

```
private void btnSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    String nama = txtNama.getText();
    String keterangan = txtKeterangan.getText();

    if (!(nama.isEmpty() && keterangan.isEmpty())) {
        Kategori kat = new Kategori();
        kat.setIdKategori(idSelected);
        kat.setNama(nama);
        kat.setKeterangan(keterangan);
        kat.save();

        tampilkanData();
        kosongkanForm();
    }
    else{
        JOptionPane.showMessageDialog(null, "Silakan isi nama dan keterangan");
    }
}
```

13. Double klik pada **btnHapus.** Netbeans akan secara otomatis menambahkan actionListener baru berupa method btnHapusActionPerformed(). Method ini akan dieksekusi setiap kali btnHapus diklik.

```
private void bthHapusActionPerformed(java.awt.event.ActionEvent evt) {
    boolean deleteSuccess = Kategori.delete(idSelected);

    if (deleteSuccess) {
        kosongkanForm();
        tampilkanData();
    }
    else{
        JOptionPane.showMessageDialog(null, "Terdapat data buku pada kategori tersebut");
    }
}
```

14. Pada constructor, tambahkan pemanggilan method kosongkanForm() dan tampilkanData(), agar ketika form ditampilkan pertama kali, maka form isian akan kosong dan list kategori langsung ditampilkan.

```
public FrmKategori() {
    initComponents();
    kosongkanForm();
    tampilkanData();
}
```

15. Run form dengan klik kanan pada FrmKategori → Run File. Kemudian uji coba tambah baru, edit, hapus, dan cari data kategori.

3.7 Percobaan 7

Lakukan hal yang sama untuk data **Anggota**

1. Buat class **Anggota** pada package **backend**, lengkapi atribut dan method-nya.
2. Lakukan test pada class TestBackend pada package **frontend** (opsional)
3. Buat **FrmAnggota** pada package **frontend** dan lengkapi komponen, method, serta event-nya.

3.9 Percobaan 8

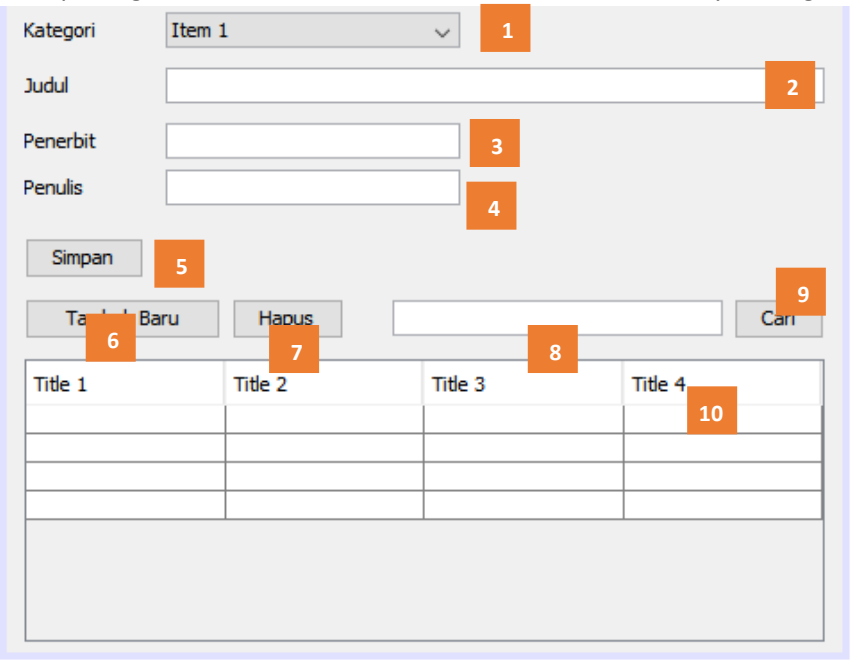
Untuk data **Buku**, caranya kurang lebih sama seperti data Kategori dan Anggota. Hanya saja yang berbeda adalah:

a. Pemanggilan **getKategori().getIdKategori()** pada query insert dan update untuk mengeset **idkategori** pada tabel **buku**
b. Query select yang melibatkan join table pada method getById() dan getAll().

*Kode lengkap class Buku dapat Anda lihat di **Lampiran 1**. Untuk test buku pada **frontend**, bisa Anda lihat di **Lampiran 2.***

Membuat GUI untuk data Buku, yang dilengkapi dengan combo box untuk memilih kategori yang terhubung dengan tabel kategori.

1. Pada package **frontend**, buat JFrame FrmBuku. Susun formnya sebagai berikut:



Nomor	Tipe Komponen	Variable Name	Properties
1	JComboBox	cmbKategori	
2	JTextField	txtJudul	text: kosong
3	JTextField	txtPenerbit	text: kosong
4	JTextField	txtPenulis	text: kosong
5	JButton	btnSimpan	text: Simpan
6	JButton	btnHapus	text: Hapus
7	JButton	btnTambahBaru	text: Tambah Baru
8	JTextField	txtCari	text: kosong
9	JButton	btnCari	text: Cari
10	JTable	tblBuku	

2. Tambahkan import backend.*, java.swing.JOptionPane, java.util.ArrayList, dan javax.swing.table.DefaultTableModel;

```
import backend.*;
import java.util.ArrayList;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

3. Tambahkan method **kosongkanForm()** untuk mengosongkan isian dan reset pilihan pada form.

```
public void kosongkanForm(){
    idSelected = 0;
    cmbKategori.setSelectedIndex(0);
    txtJudul.setText("");
    txtPenerbit.setText("");
    txtPenulis.setText("");
}
```

4. Tambahkan method **tampilkanData()** untuk mengambil data buku dari database berdasarkan keyword dari txtCari dan menampilkannya ke JTable tblBuku.

```
public void tampilkanData(){
    Object[] namaKolom = new Object[] {"ID", "Kategori", "Judul", "Penerbit", "Penulis"};

    DefaultTableModel model = new DefaultTableModel();
    model.setColumnIdentifiers(namaKolom);

    String keyword = txtCari.getText();
    ArrayList<Buku> listBuku = Buku.getAll(keyword);
    Object[] rowData = new Object[5];

    for (Buku buku : listBuku) {
        rowData[0] = buku.getIdbuku();
        rowData[1] = buku.getKategori().getNama();
        rowData[2] = buku.getJudul();
        rowData[3] = buku.getPenerbit();
        rowData[4] = buku.getPenulis();

        model.addRow(rowData);
    }

    tblBuku.setModel(model);
}
```

5. Tambahkan method **setCmbKategori().** Pada method ini semua kategori didapatkan dari database menggunakan method Kategori.getAll() kemudian dijadikan sebagai opsi untuk cmbKategori.

```
public void setCmbKategori(){
    DefaultComboBoxModel model = new DefaultComboBoxModel(Kategori.getAll().toArray());
    cmbKategori.setModel(model);
}
```

6. Agar cmbKategori menampilkan **nama kategori**, maka override method **toString()** pada class **Kategori.** Tambahkan kode berikut ini pada class **Kategori**.

```
@Override
public String toString() {
    return nama;
}
```

7. Pada constructor, tambahkan pemanggilan method kosongkanForm(), tampilkanCmbKategori() dan tampilkanData(), agar ketika form ditampilkan pertama kali, maka form isian akan kosong, daftar buku ditampilkan, dan opsi untuk kategori sudah tersedia.

```
public FrmBuku() {
    initComponents();
    kosongkanForm();
    tampilkanData();
    setCmbKategori();
}
```

8. Double klik pada **btnTambahBaru** untuk menambahkan method btnTambahBaruActionPerformed(). Panggil method kosongkanForm() agar form siap digunakan untuk menginputkan data baru.

```
private void btnTambahBaruActionPerformed(java.awt.event.ActionEvent evt) {
    kosongkanForm();
}
```

9. Double klik pada **btnCari** untuk menambahkan method btnCariActionPerformed(). Panggil method tampilkanData untuk me-refresh daftar buku pada JTable.

```
private void btnCariActionPerformed(java.awt.event.ActionEvent evt) {
    tampilkanData();
}
```

10. Agar row data pada **tblBuku** dapat dipilih untuk diedit atau dihapus, maka perlu ditambahkan event mouse click pada tblBuku. Ketika pengguna mengklik pada tblBuku, maka data tersebut akan ditampilkan pada form di atas tabel. Caranya, klik kanan pada tblBuku, pilih Events → Mouse → MouseClicked.

    Pada kode program event mouse click nya **tblKategori**, nilai pada text field diset berdasarkan data pada tabel. Namun untuk **tblBuku**, form memerlukan data idkategori yang tidak tersedia pada tabel. Oleh karena itu, langkah yang dilakukan adalah:

    a. Tentukan row yang diklik
    b. Tentukan idbuku
    c. Tentukan objek buku yang sesuai dengan menggunakan method getById()
    d. Isi form sesuai data buku

```
private void tblBukuMouseClicked(java.awt.event.MouseEvent evt) {
    DefaultTableModel model = (DefaultTableModel)tblBuku.getModel();
    int rowIndex = tblBuku.getSelectedRow();
    idSelected = Integer.parseInt(model.getValueAt(rowIndex, 0).toString());

    Buku buku = Buku.getById(idSelected);

    cmbKategori.getModel().setSelectedItem(buku.getKategori());
    txtJudul.setText(buku.getJudul());
    txtPenerbit.setText(buku.getPenerbit());
    txtPenulis.setText(buku.getPenulis());
}
```

11. Double klik pada **btnHapus** untuk menambahkan kode untuk menghapus data.

```
private void bthHapusActionPerformed(java.awt.event.ActionEvent evt) {
    boolean deleteSuccess = Buku.delete(idSelected);

    if (deleteSuccess) {
        kosongkanForm();
        tampilkanData();
    }
    else{
        JOptionPane.showMessageDialog(null, "Terdapat data buku pada kategori tersebut");
    }
}
```

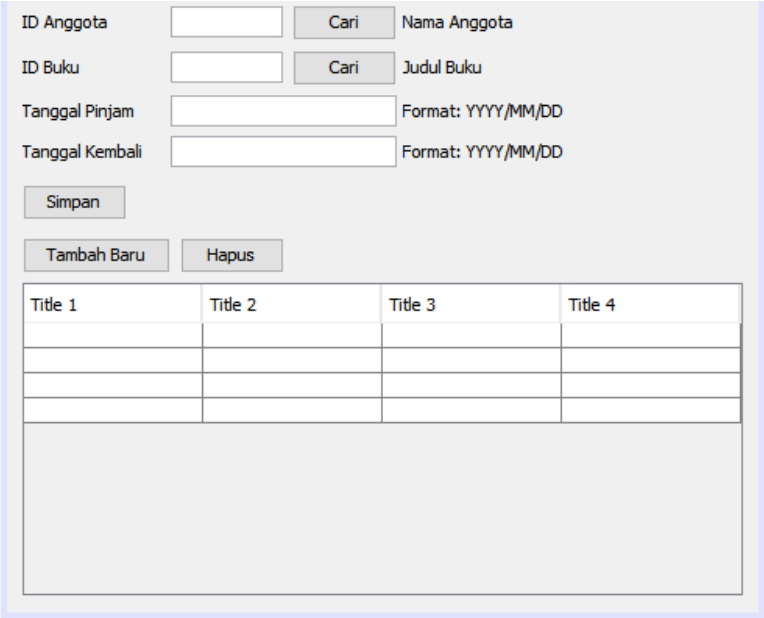
12. Double klik pada **btnSimpan** untuk menambahkan kode untuk menyimpan data. Method save() pada class Buku sudah secara otomatis menentukan apakah data akan di-insert atau di-update.

```
private void btnSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    String judul = txtJudul.getText();
    String penerbit = txtPenerbit.getText();
    String penulis = txtPenulis.getText();

    if (!(judul.isEmpty() && penerbit.isEmpty() && penulis.isEmpty())) {
        Buku buku = new Buku();
        buku.setIdBuku(idSelected);
        buku.setKategori((Kategori)cmbKategori.getSelectedItem());
        buku.setJudul(judul);
        buku.setPenerbit(penerbit);
        buku.setPenulis(penulis);
        buku.save();

        kosongkanForm();
        tampilkanData();
    }
    else{
        JOptionPane.showMessageDialog(null, "Silakan isi form dengan lengkap");
    }
}
```

13. Jalankan form dengan opsi Run File. Kemudian ujicoba tambah baru, edit, hapus, dan cari.

4. Tugas

1. Buatlah class **Peminjaman.**
2. Buatlah form **FrmPeminjaman** dan susun sebagai berikut:



3. Atur kode program agar dapat menangani transaksi peminjaman dan pengembalian.

   Note:
   Pada textbox ID Anggota, pengguna tinggal memasukkan ID anggota, kemudian menekan tombol Cari. Jika ketemu, maka label **"Nama Anggota"** yang ada di samping tombol Cari tersebut akan menampilkan nama anggota dari ID yang dimasukkan tadi. Begitu juga dengan ID Buku.

Lampiran

Lampiran 1. Kode lengkap class Buku

```
package backend;
import java.util.ArrayList;
import java.sql.*;

public class Buku {
    private int idBuku;
    private Kategori kategori;
    private String judul;
    private String penerbit;
    private String penulis;

    public Buku() {
    }

    public Buku(Kategori kategori, String judul, String penerbit, String penulis) {
        this.kategori = kategori;
        this.judul = judul;
        this.penerbit = penerbit;
        this.penulis = penulis;
    }

    public int getIdbuku() {
        return idBuku;
    }

    public void setIdBuku(int idBuku) {
        this.idBuku = idBuku;
    }

    public String getJudul() {
        return judul;
    }

    public void setJudul(String judul) {
        this.judul = judul;
    }

    public Kategori getKategori() {
        return kategori;
    }

    public void setKategori(Kategori kategori) {
        this.kategori = kategori;
    }

    public String getPenerbit() {
        return penerbit;
    }

    public void setPenerbit(String penerbit) {
        this.penerbit = penerbit;
    }

    public String getPenulis() {
        return penulis;
    }

    public void setPenulis(String penulis) {
        this.penulis = penulis;
```

```
    }

    public static ArrayList<Buku> getAll(String keyword) {
        ArrayList<Buku> listBuku = new ArrayList<>();

        String sql =
            "SELECT buku.*, kategori.nama, kategori.keterangan FROM buku " +
            "LEFT JOIN kategori ON buku.id_kategori = kategori.id_kategori " +
            "WHERE judul LIKE ? OR penerbit LIKE ? OR penulis LIKE ?";

        try (Connection conn = DBHelper.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            String likeKeyword = "%" + keyword + "%";

            ps.setString(1, likeKeyword);
            ps.setString(2, likeKeyword);
            ps.setString(3, likeKeyword);

            try (ResultSet rs = ps.executeQuery()) {
                while (rs.next()) {

                    Kategori kat = new Kategori();
                    kat.setIdKategori(rs.getInt("id_kategori"));
                    kat.setNama(rs.getString("nama"));
                    kat.setKeterangan(rs.getString("keterangan"));

                    Buku buku = new Buku();
                    buku.setIdBuku(rs.getInt("id_buku"));
                    buku.setKategori(kat);
                    buku.setJudul(rs.getString("judul"));
                    buku.setPenerbit(rs.getString("penerbit"));
                    buku.setPenulis(rs.getString("penulis"));

                    listBuku.add(buku);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return listBuku;
    }

    public static Buku getById(int id) {
        Buku buku = null;

        String sql =
            "SELECT buku.*, kategori.nama, kategori.keterangan FROM buku " +
            "LEFT JOIN kategori ON buku.id_kategori = kategori.id_kategori " +
            "WHERE id_buku = ?";

        try (Connection conn = DBHelper.getConnection();
             PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setInt(1, id);

            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    Kategori kat = new Kategori();
                    kat.setIdKategori(rs.getInt("id_kategori"));
                    kat.setNama(rs.getString("nama"));
                    kat.setKeterangan(rs.getString("keterangan"));

                    buku = new Buku();
                    buku.setIdBuku(rs.getInt("id_buku"));
```

```
                    buku.setKategori(kat);
                    buku.setJudul(rs.getString("judul"));
                    buku.setPenerbit(rs.getString("penerbit"));
                    buku.setPenulis(rs.getString("penulis"));
                }
            }

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

        return buku;
    }

    public void save() {
        if (this.idBuku == 0) {
            String sql = "INSERT INTO buku (id_kategori, judul, penerbit, penulis) "
                       + "VALUES (?, ?, ?, ?)";

            try (Connection conn = DBHelper.getConnection();
                 PreparedStatement ps = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS)) {

                ps.setInt(1, this.kategori.getIdKategori());
                ps.setString(2, this.judul);
                ps.setString(3, this.penerbit);
                ps.setString(4, this.penulis);

                ps.executeUpdate();

                try (ResultSet rs = ps.getGeneratedKeys()) {
                    if (rs.next()) {
                        this.idBuku = rs.getInt(1);
                    }
                }

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

        } else {
            String sql = "UPDATE buku SET "
                    + "id_kategori = ?, "
                    + "judul = ?, "
                    + "penerbit = ?, "
                    + "penulis = ? "
                    + "WHERE id_buku = ?";

            try (Connection conn = DBHelper.getConnection();
                 PreparedStatement ps = conn.prepareStatement(sql)) {

                ps.setInt(1, this.kategori.getIdKategori());
                ps.setString(2, this.judul);
                ps.setString(3, this.penerbit);
                ps.setString(4, this.penulis);
                ps.setInt(5, this.idBuku);

                ps.executeUpdate();

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

    public static boolean delete(int id) {
```

```
        String sql = "DELETE FROM buku WHERE id_buku = ?";

        try (Connection conn = DBHelper.getConnection();
                PreparedStatement ps = conn.prepareStatement(sql)) {
            ps.setInt(1, id);
            ps.executeUpdate();
            return true;

        } catch (SQLException e) {
            if (e.getErrorCode() == 1451) {
                System.out.println("Masih ada peminjaman pada kategori ini");
            }

            e.printStackTrace();
            return false;
        }
    }
}
```

Lampiran 2. Kode untuk TestBackend

```
import backend.*;

public class TestBackend {
    public static void main(String[] args)
    {
        Kategori novel = new Kategori("Novel", "Koleksi buku novel");
        Kategori referensi = new Kategori("Referensi", "Buku referensi ilmiah");

        //test getById
        Kategori kat1 = Kategori.getById(25);

        if (kat1 != null) {
            System.out.println(kat1.getNama());
        }

        novel.save();
        referensi.save();

        Buku buku1 = new Buku(novel, "Timun Mas", "Elex Media", "Bang Supit");
        Buku buku2 = new Buku(referensi, "Metode Linier", "Springer", "Alex Baldwin");
        Buku buku3 = new Buku(novel, "Bintang Terang", "Erlangga", "Mat Sewoot");

        // test insert
        buku1.save();
        buku2.save();
        buku3.save();

        // test update
        buku2.setJudul("Aljabar Linier");
        buku2.save();

        //test getById
        Buku buku = Buku.getById(1);

        if (buku != null) {
            System.out.println("Kategori: " + buku.getKategori().getNama() + ", Judul: " +
buku.getJudul());
        }

        // test delete
        Buku.delete(buku3.getIdBuku());

        // test select all
        for(Buku b : Buku.getAll(""))
        {
            System.out.println("Kategori: " + b.getKategori().getNama() + ", Judul: " +
b.getJudul());
        }

        // test search
        for(Buku b : Buku.search("timun"))
        {
            System.out.println("Kategori: " + b.getKategori().getNama() + ", Judul: " +
b.getJudul());
        }
    }
}
```

Lampiran 3. Static keyword

Static attribute → attribute yang hanya terkait pada class, tetapi tidak terkait pada suatu objek tertentu
Static method → method yang hanya terkait pada class, tetapi tidak terkait pada suatu objek tertentu

Method getJudul() bukan merupakan method statis, artinya method ini terkait pada objek.

- buku1.getJudul() → mengembalikan nilai atribut judul dari buku1
- buku2.getJudul() → mengembalikan nilai atribut judul dari buku2
- buku3.getJudul() → mengembalikan nilai atribut judul dari buku3
- dst

Terkadang kita ingin membuat method yang terkait pada class Buku, tetapi tidak ada kaitannya dengan objek buku tertentu. Misal method getAll() pada class Buku berfungsi untuk mengembalikan semua data buku dari database. Jika method getAll() bukan static method, pemanggilan method harus melalui objek seperti biasanya, misalnya:

- buku4.getAll() → mengembalikan semua data buku dari database
- buku5.getAll() → mengembalikan semua data buku dari database
- buku6.getAll() → mengembalikan semua data buku dari database
- dst

Ketiga method tersebut melakukan hal yang persis sama dan tidak bergantung pada objek yang memanggilnya. Jika suatu method bersifat demikian, maka method tersebut sebaiknya dibuat sebagai static method sehingga dapat diakses langsung menggunakan nama class nya tanpa membuat objek terlebih dahulu.

- Buku.getAll()
- Buku.search(String keyword)
- Buku.getById(int id)

Pada beberapa konsep pemrograman, method-method helper yang berfungsi untuk melakukan manipulasi database, yaitu method untuk Create Read Update Delete, umumnya dipisah dari class Buku menjadi class tersendiri, misalnya BukuHelper.