GUI dan Database

1. Kompetensi

Setelah menempuh materi percobaan ini, mahasiswa mampu mengenal:

1. Menggunakan paradigma berorientasi objek untuk interaksi dengan database
2. Membuat backend dan frontend
3. Membuat form sebagai frontend

2. Pendahuluan

Kali ini kita akan menggunakan paradigma berorientasi objek yang telah kita pelajari untuk membuat aplikasi berbasis database dan dilengkapi dengan form sebagai Graphical User Interface (GUI).

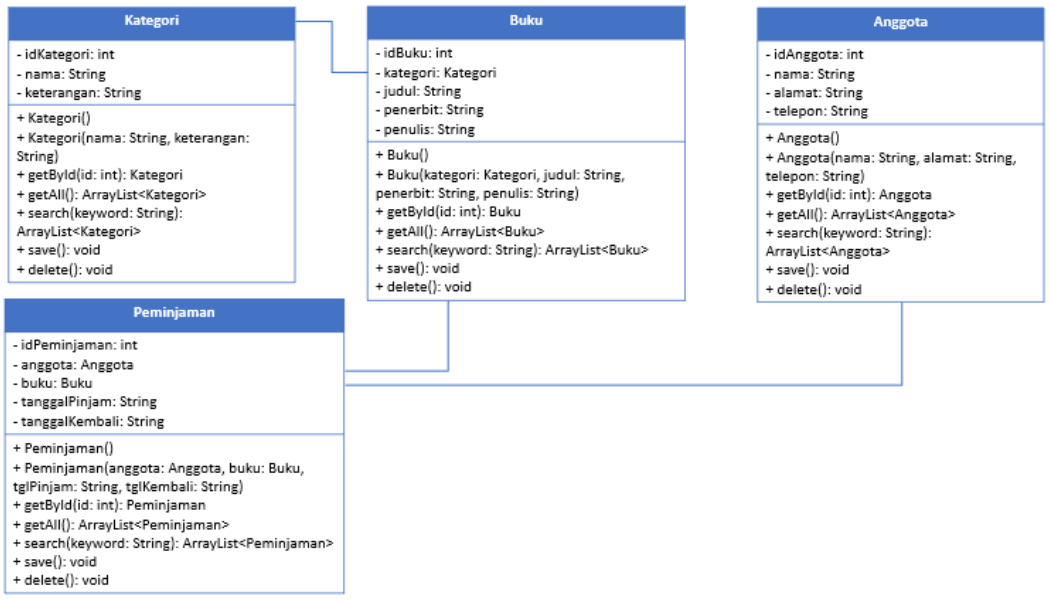
Secara umum, tahapan yang akan kita lakukan adalah sebagai berikut:

1. Membuat database dan tabel-tabelnya.
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 yang merupakan antarmuka kepada pengguna. Frontend ini bisa berbasis teks (console), GUI, web, mobile, dan sebagainya.

Library yang digunakan untuk project ini antara lain:

1. JDBC, untuk melakukan interaksi ke database.
2. ArrayList, untuk menampung data hasil query ke database.
3. Swing, untuk membuat tampilan 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:



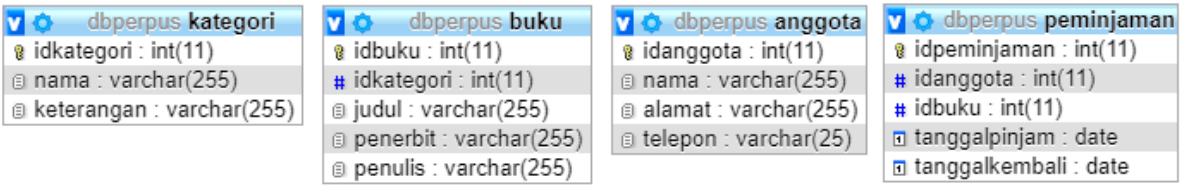
Dapat dilihat dari class diagram diatas, terdapat relasi antar class. Class Buku berelasi dengan Kategori dikarenakan terdapat atribut bertipe data Kategori di dalam class buku. Begitu juga class Peminjaman yang berelasi dengan class Buku dan Anggota.

3. Percobaan

3.1 Percobaan 1

Membuat database.

1. Langkah pertama untuk percobaan ini adalah membuat database. Install XAMPP, buka phpMyAdmin, buat database **dbperpus**, dan tabel-tabelnya:



   Set semua primary key id pada tiap tabel (idanggota, idkategori, idpeminjaman, idbuku) dengan **Auto Increment**.

3.2 Percobaan 2

Mempersiapkan project.

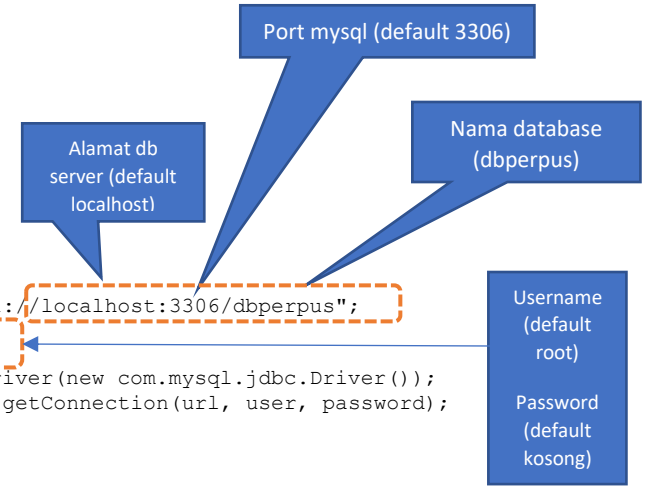
1. Buat project baru, beri nama **Perpustakaan**.
2. Pada project explorer, klik kanan pada Libraries → Add Library, pilih MySQL JDBC Driver.
3. Buat package **frontend** dan **backend**. Cara membuat package adalah, pada project explorer, klik kanan pada Source Packages → New → Java Package, beri nama package nya (frontend, backend).

3.3 Percobaan 3

Membuat class helper untuk mengeksekusi query SQL.

1. Pada package **backend**, buat class **DBHelper**.
2. Import java.sql.*
3. Didalam class ini ada method-method antara lain:
   a. **bukaKoneksi()**, untuk membuka koneksi ke database
   b. **insertQueryGetId(String query)**, untuk melakukan insert ke tabel dan mengembalikan nilai ID yang digenerate oleh database (hasil Auto Increment).
   c. **executeQuery(String query)**, untuk mengeksekusi query yang tidak mengembalikan nilai (misal: insert, update, delete).
   d. **selectQuery(String query)**, untuk mengeksekusi select query yang mengembalikan nilai hasil query.
4. Berikut adalah kode dari class DBHelper. Anda perlu sesuaikan method **bukaKoneksi()** dengan setting database yang terinstall di sistem anda. Namun jika anda install XAMPP secara default, maka setting ini tidak perlu diubah, mungkin nama database saja yang perlu disesuaikan. Silahkan COPAS kode berikut ini.

```
package backend;

import java.sql.*;

public class DBHelper
{
    private static Connection koneksi;

    public static void bukaKoneksi()
    {
        if(koneksi == null)
        {
            try
            {
                String url = "jdbc:mysql://localhost:3306/dbperpus";
                String user = "root";
                String password = "";
                DriverManager.registerDriver(new com.mysql.jdbc.Driver());
                koneksi = DriverManager.getConnection(url, user, password);
            }
            catch (SQLException t)
            {
                System.out.println("Error koneksi!");
            }
        }
    }

    public static int insertQueryGetId(String query)
    {
        bukaKoneksi();
        int num = 0;
        int result = -1;

        try
        {
            Statement stmt = koneksi.createStatement();
            num = stmt.executeUpdate(query, Statement.RETURN_GENERATED_KEYS);

            ResultSet rs = stmt.getGeneratedKeys();

            if (rs.next())
            {
                result = rs.getInt(1);
            }

            rs.close();
            stmt.close();
        }
        catch (Exception e)
        {
            e.printStackTrace();
            result = -1;
        }

        return result;
    }

    public static boolean executeQuery(String query)
    {
        bukaKoneksi();
        boolean result = false;

        try
        {
            Statement stmt = koneksi.createStatement();
            stmt.executeUpdate(query);

            result = true;

            stmt.close();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
```



```
        return result;
    }

    public static ResultSet selectQuery(String query)
    {
        bukaKoneksi();
        ResultSet rs = null;

        try
        {
            Statement stmt = koneksi.createStatement();
            rs = stmt.executeQuery(query);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }

        return rs;
    }
}
```

3.4 Percobaan 4

Membuat class **Kategori** untuk menghandle CRUD pada tabel kategori.

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

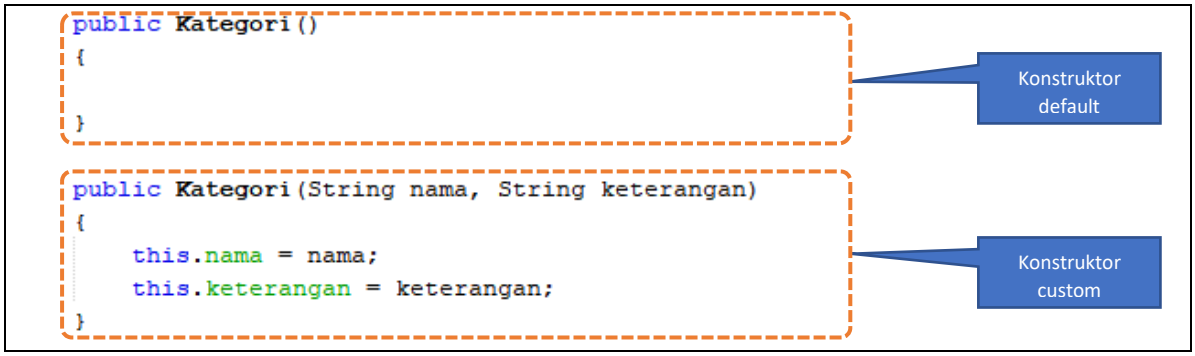
```
import java.util.ArrayList;
import java.sql.*;
```

3. Tambahkan atribut sesuai field pada tabel kategori.

```
    private int idkategori;
    private String nama;
    private String keterangan;
```

4. Tambahkan getter setter 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, centang semua atribut yang ada (idkategori, nama, keterangan).

5. Tambahkan konstruktor default dan konstruktor custom, yang digunakan untuk mengeset atribut nama dan keterangan. Atribut idkategori tidak boleh diset, karena id ini akan digenerate secara otomatis lewat fitur AutoIncrement pada MySQL.



6. Tambahkan method **getById()** untuk mendapatkan objek Kategori yang ada di database berdasarkan id-nya.

```
public Kategori getById(int id)
{
    Kategori kat = new Kategori();
    ResultSet rs = DBHelper.selectQuery("SELECT * FROM kategori "
                                        + " WHERE idkategori = '" + id + "'");

    try
    {
        while(rs.next())
        {
            kat = new Kategori();
            kat.setIdkategori(rs.getInt("idkategori"));
            kat.setNama(rs.getString("nama"));
            kat.setKeterangan(rs.getString("keterangan"));
        }
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }

    return kat;
}
```

7. Tambahkan method **getAll()** untuk mendapatkan semua data Kategori yang ada di database, dan ditampung ke ArrayList<Kategori>.

```
public ArrayList<Kategori> getAll()
{
    ArrayList<Kategori> ListKategori = new ArrayList();

    ResultSet rs = DBHelper.selectQuery("SELECT * FROM kategori");

    try
    {
        while(rs.next())
        {
            Kategori kat = new Kategori();
            kat.setIdkategori(rs.getInt("idkategori"));
            kat.setNama(rs.getString("nama"));
            kat.setKeterangan(rs.getString("keterangan"));

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

    return ListKategori;
}
```

8. Tambahkan method **search()** agar bisa melakukan pencarian data. Method ini mirip dengan method **getAll()** namun querynya berbeda.

```
public ArrayList<Kategori> search(String keyword)
{
    ArrayList<Kategori> ListKategori = new ArrayList();

    String sql = "SELECT * FROM kategori WHERE "
               + "     nama LIKE '%" + keyword + "%' "
               + "     OR keterangan LIKE '%" + keyword + "%' ";

    ResultSet rs = DBHelper.selectQuery(sql);

    try
    {
        while(rs.next())
        {
            Kategori kat = new Kategori();
            kat.setIdkategori(rs.getInt("idkategori"));
            kat.setNama(rs.getString("nama"));
            kat.setKeterangan(rs.getString("keterangan"));

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

    return ListKategori;
}
```

9. Tambahkan method **save()**. Method ini memiliki dua fungsi, yaitu insert dan update. Jika data yang diinputkan belum ada (idkategori = 0) maka akan otomatis insert. Jika data yang diinputkan sudah ada, maka otomatis update.

```
public void save()
{
    if(getById(idkategori).getIdkategori() == 0)
    {
        String SQL = "INSERT INTO kategori (nama, keterangan) VALUES("
                   + "     '" + this.nama + "', "
                   + "     '" + this.keterangan + "' "
                   + "     )";
        this.idkategori = DBHelper.insertQueryGetId(SQL);
    }
    else
    {
        String SQL = "UPDATE kategori SET "
                   + "     nama = '" + this.nama + "', "
                   + "     keterangan = '" + this.keterangan + "' "
                   + "     WHERE idkategori = '" + this.idkategori + "'";
        DBHelper.executeQuery(SQL);
    }
}
```

10. Tambahkan method **delete()** untuk melakukan operasi penghapusan pada tabel kategori pada database.

```
    public void delete()
    {
        String SQL = "DELETE FROM kategori WHERE idkategori = '" + this.idkategori + "'";
        DBHelper.executeQuery(SQL);
    }
```

3.5 Percobaan 5

Mencoba backed yang sudah dibuat dengan mengoperasikannya lewat frontend berbasis teks (console). Percobaan ini dapat anda skip jika anda telah yakin bahwa backend yang anda buat sudah berfungsi dengan baik.

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

```
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
        kat3.delete();

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

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

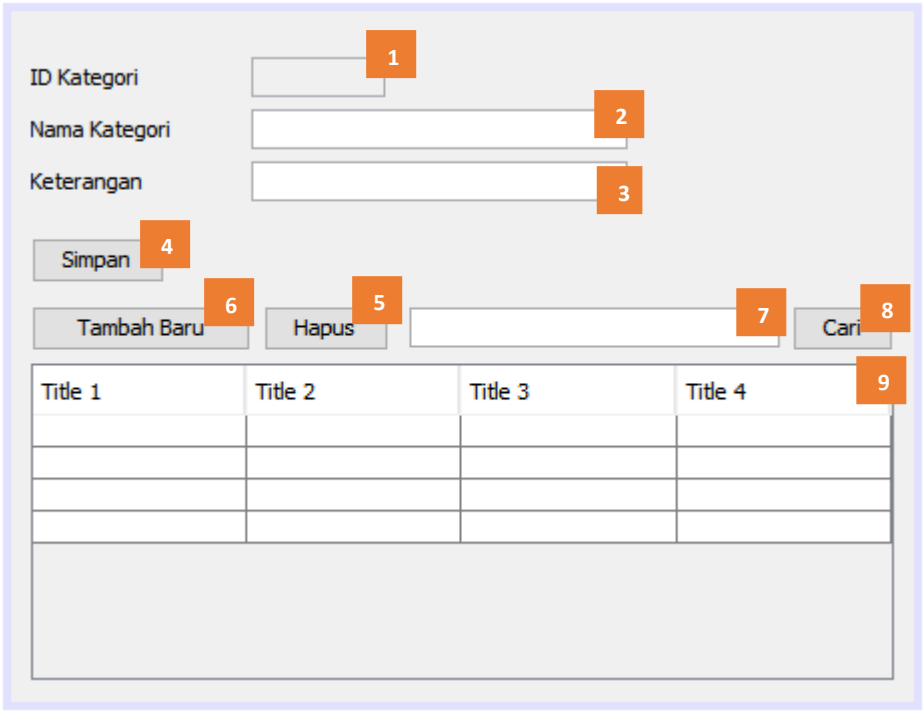
3. Jalankan TestBackend dengan klik kanan, Run File. Cocokkan outputnya:

```
run:
Nama: Novel, Ket: Koleksi buku novel
Nama: Referensi, Ket: Koleksi buku referensi ilmiah
Nama: Referensi, Ket: Koleksi buku referensi ilmiah
BUILD SUCCESSFUL (total time: 0 seconds)
```

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 berikut ini, atur propertinya sesuai nomor:



Pengaturan properti:

Nomor	Tipe Komponen	Nama Komponen	Properties
1	JTextField	txtIdKategori	text: kosong, enabled: uncheck
2	JTextField	txtNama	text: kosong
3	JTextField	txtKeterangan	text: kosong
4	JButton	btnSimpan	text: Simpan
5	JButton	btnHapus	text: Hapus
6	JButton	btnTambahBaru	text: Tambah Baru
7	JTextField	txtCari	text: kosong
8	JButton	btnCari	text: Cari
9	JTable	tblKategori	

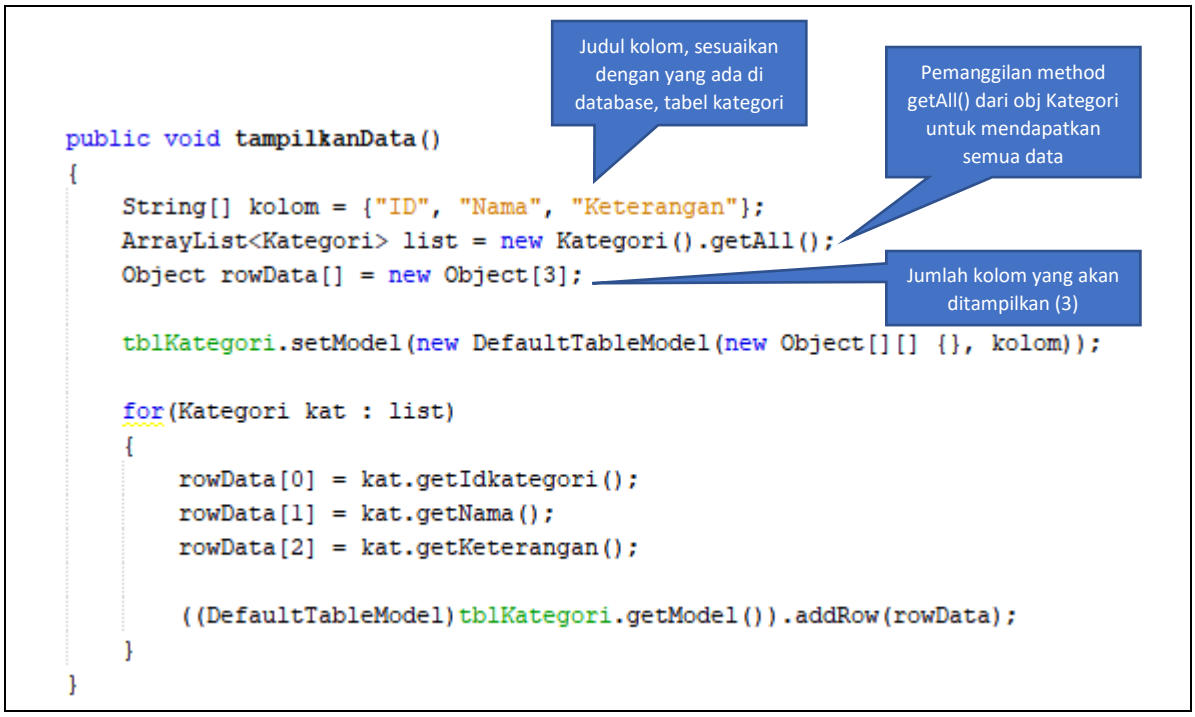
3. Edit kodenya, tambahkan import backend.*, java.util.ArrayList, javax.swing.table.DefaultTableModel;

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

4. Tambahkan method **kosongkanForm()** untuk mengosongkan isian textbox pada form.

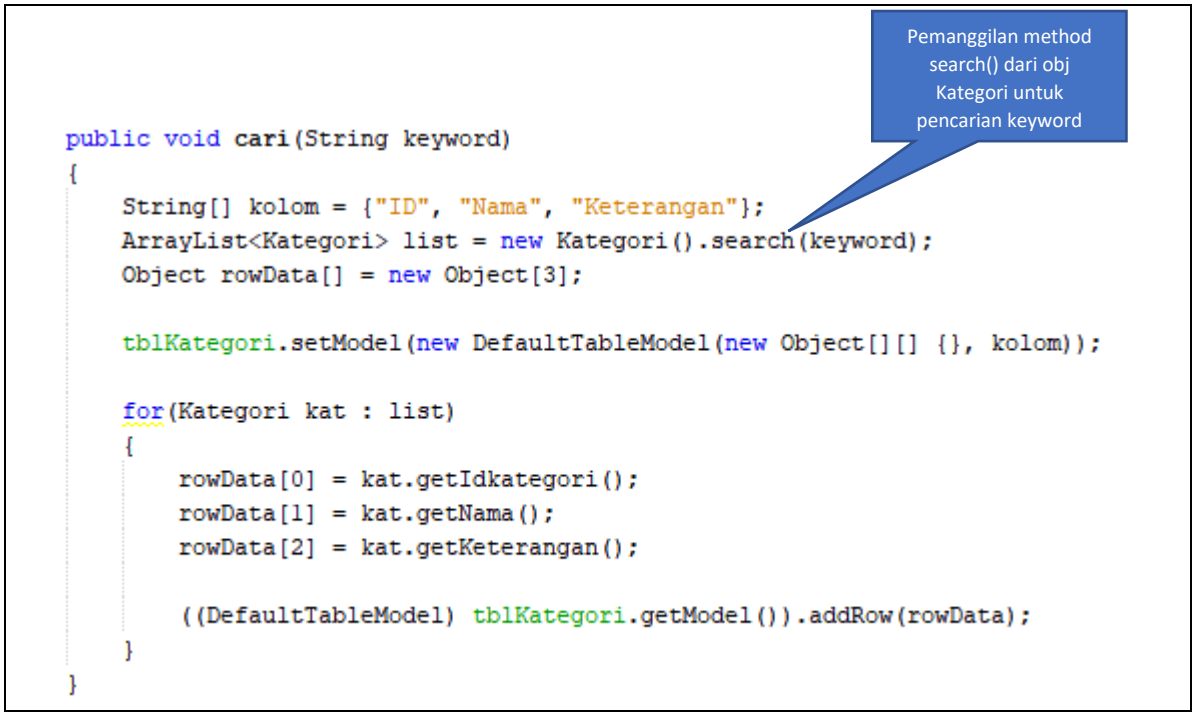
```
    public void kosongkanForm()
    {
        txtIdKategori.setText("0");
        txtNama.setText("");
        txtKeterangan.setText("");
    }
```

5. Tambahkan method **tampilkanData()** untuk mengambil semua data kategori dari database dan menampilkannya ke JTable tblKategori.




```
public void tampilkanData()
{
    String[] kolom = {"ID", "Nama", "Keterangan"};
    ArrayList<Kategori> list = new Kategori().getAll();
    Object rowData[] = new Object[3];

    tblKategori.setModel(new DefaultTableModel(new Object[][] {}, kolom));

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

        ((DefaultTableModel)tblKategori.getModel()).addRow(rowData);
    }
}
```

6. Tambahkan method **cari()** untuk melakukan pencarian berdasarkan keyword tertentu.




```
public void cari(String keyword)
{
    String[] kolom = {"ID", "Nama", "Keterangan"};
    ArrayList<Kategori> list = new Kategori().search(keyword);
    Object rowData[] = new Object[3];

    tblKategori.setModel(new DefaultTableModel(new Object[][] {}, kolom));

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

        ((DefaultTableModel) tblKategori.getModel()).addRow(rowData);
    }
}
```

7. Pada konstruktor, tambahkan pemanggilan method kosongkanForm() dan tampilkanData(), agar ketika form ditampilkan pertama kali, maka form isian akan kosong dan data kategori langsung ditampilkan.

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

8. Double klik pada **btnSimpan** untuk menambahkan kode untuk menyimpan data. Aksi menyimpan ini secara otomatis menentukan apakah insert atau update, karena pada method **save()** dari objek **kat**, sudah dicek apakah data baru atau sudah ada (baca Percobaan 4, no. 9)

```
private void btnSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Kategori kat = new Kategori();
    kat.setIdkategori(Integer.parseInt(txtIdKategori.getText()));
    kat.setNama(txtNama.getText());
    kat.setKeterangan(txtKeterangan.getText());
    kat.save();
    txtIdKategori.setText(Integer.toString(kat.getIdkategori()));
    tampilkanData();
}
```

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

```
private void btnHapusActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel model = (DefaultTableModel)tblKategori.getModel();
    int row = tblKategori.getSelectedRow();

    Kategori kat = new Kategori().getById(Integer.parseInt(model.getValueAt(row, 0).toString()));
    kat.delete();
    kosongkanForm();
    tampilkanData();
}
```

10. Double klik pada **btnTambahBaru** untuk mengosongkan form sehingga dapat digunakan untuk menginputkan data baru.

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

11. Double klik pada **btnCari** untuk melakukan pencarian terhadap keyword yang dimasukkan pada **txtCari**.

```
private void btnCariActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    cari(txtCari.getText());
}
```

12. Agar kita dapat memilih data yang ada pada **tblKategori**, agar dapat diedit atau dihapus, maka kita tambahkan event mouse click pada **tblKategori**. Ketika pengguna mengklik pada tblKategori, maka data tersebut akan ditampilkan di tex field. Caranya, klik kanan pada **tblKategori**, pilih Events → Mouse → MouseClicked. Tambahkan kode berikut ini:

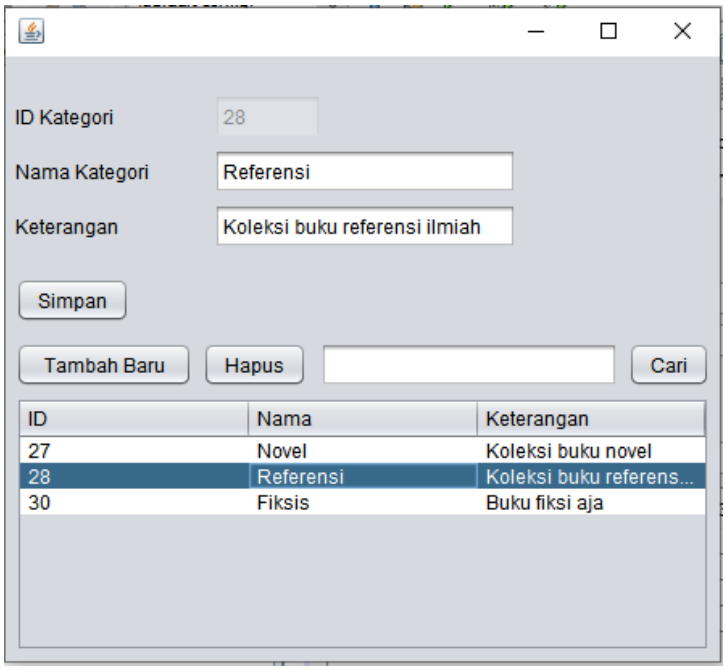
```
private void tblKategoriMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel model = (DefaultTableModel)tblKategori.getModel();
    int row = tblKategori.getSelectedRow();

    txtIdKategori.setText(model.getValueAt(row, 0).toString());
    txtNama.setText(model.getValueAt(row, 1).toString());
    txtKeterangan.setText(model.getValueAt(row, 2).toString());
}
```

Kolom ke-0, utk mengeset txtIdKategori,

kolom ke-1 utk mengeset txtNama,

kolom ke-2 untuk mengeset txtKeterangan

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



3.6 Percobaan 6

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**.

3.7 Percobaan 7

Buat form untuk data **Anggota**.

1. Buat **FrmAnggota** pada package **frontend** dan lengkapi komponen, method serta event-nya.

3.7 Percobaan 7

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**

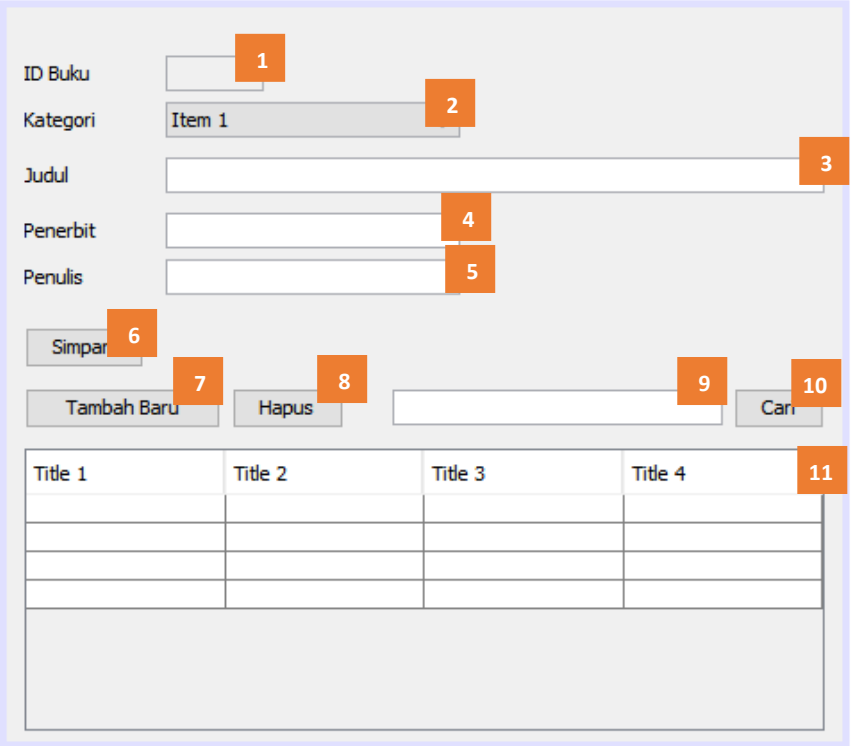
b. Query select yang melibatkan join table pada method getById(), getAll() dan search().

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

3.8 Percobaan 8

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	Nama Komponen	Properties
1	JTextField	txtIdBuku	text: kosong, enabled: uncheck
2	JComboBox	cmbKategori	
3	JTextField	txtJudul	text: kosong
4	JTextField	txtPenerbit	text: kosong
5	JTextField	txtPenulis	text: kosong
6	JButton	btnSimpan	text: Simpan
7	JButton	btnHapus	text: Hapus
8	JButton	btnTambahBaru	text: Tambah Baru
9	JTextField	txtCari	text: kosong
10	JButton	btnCari	text: Cari
11	JTable	tblBuku	

2. Edit kodenya, tambahkan import backend.*, java.util.ArrayList, javax.swing.table.DefaultTableModel, javax.swing.DefaultComboBoxModel;

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

3. Tambahkan method **kosongkanForm()** untuk mengosongkan isian textbox pada form.

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

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

```
public void tampilkanData()
{
    String[] kolom = {"ID", "Kategori", "Judul", "Penulis", "Penerbit"};
    ArrayList<Buku> list = new Buku().getAll();
    Object rowData[] = new Object[5];

    tblBuku.setModel(new DefaultTableModel(new Object[][] {}, kolom));

    for(int i = 0; i < list.size(); i++)
    {
        rowData[0] = list.get(i).getIdbuku();
        rowData[1] = list.get(i).getKategori().getNama();
        rowData[2] = list.get(i).getJudul();
        rowData[3] = list.get(i).getPenulis();
        rowData[4] = list.get(i).getPenerbit();

        ((DefaultTableModel)tblBuku.getModel()).addRow(rowData);
    }
}
```

5. Tambahkan method **cari()** untuk melakukan pencarian berdasarkan keyword tertentu.

```
public void cari(String keyword)
{
    String[] kolom = {"ID", "Kategori", "Judul", "Penulis", "Penerbit"};
    ArrayList<Buku> list = new Buku().search(keyword);
    Object rowData[] = new Object[5];

    tblBuku.setModel(new DefaultTableModel(new Object[][] {}, kolom));

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

        ((DefaultTableModel)tblBuku.getModel()).addRow(rowData);
    }
}
```

6. Tambahkan method **tampilkanCmbKategori()** untuk mengambil data Kategori dari database dan menampilkannya ke **cmbKategori**.

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

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

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

7. Pada konstruktor, tambahkan pemanggilan method kosongkanForm(), tampilkanCmbKategori() dan tampilkanData(), agar ketika form ditampilkan pertama kali, maka form isian akan kosong dan data buku langsung ditampilkan.

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

8. Double klik pada **btnSimpan** untuk menambahkan kode untuk menyimpan data. Aksi menyimpan ini secara otomatis menentukan apakah insert atau update, karena pada method **save()** dari objek **kat**, sudah dicek apakah data baru atau sudah ada (baca Percobaan 4, no. 9)

```
private void btnSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Buku buku = new Buku();
    buku.setIdbuku(Integer.parseInt(txtIdBuku.getText()));
    buku.setKategori((Kategori)cmbKategori.getSelectedItem());
    buku.setJudul(txtJudul.getText());
    buku.setPenulis(txtPenulis.getText());
    buku.setPenerbit(txtPenerbit.getText());
    buku.save();

    txtIdBuku.setText(Integer.toString(buku.getIdbuku()));

    tampilkanData();
}
```

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

```
private void btnHapusActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel model = (DefaultTableModel)tblBuku.getModel();
    int row = tblBuku.getSelectedRow();

    Buku buku = new Buku().getById(Integer.parseInt(model.getValueAt(row, 0).toString()));
    buku.delete();
    kosongkanForm();
    tampilkanData();
}
```

10. Double klik pada **btnTambahBaru** untuk mengosongkan form sehingga dapat digunakan untuk menginputkan data baru.

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

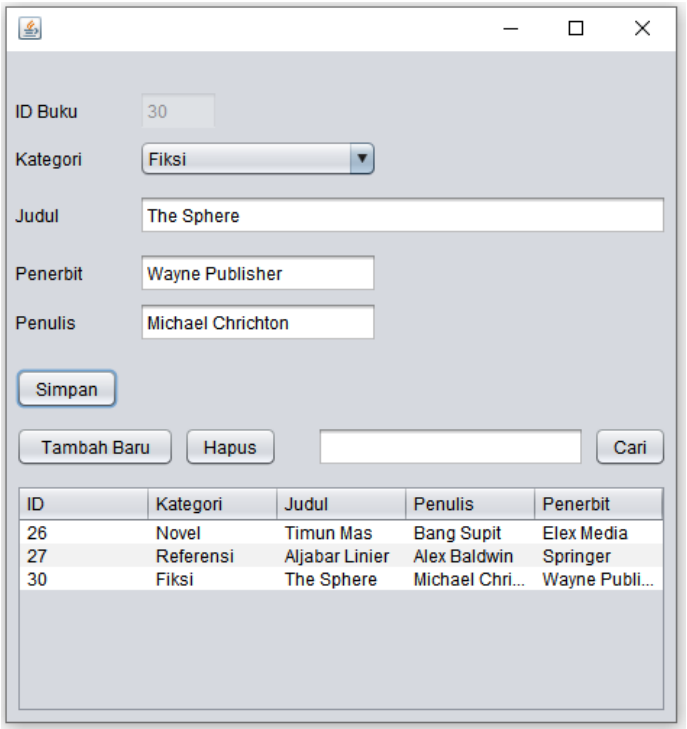
11. Double klik pada **btnCari** untuk melakukan pencarian terhadap keyword yang dimasukkan pada **txtCari**.

```
private void btnCariActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    cari(txtCari.getText());
}
```

12. Agar kita dapat memilih data yang ada pada **tblBuku**, agar dapat ditampilkan di textfield untuk diedit atau dihapus, maka kita tambahkan event mouse click pada **tblKategori**. Caranya, klik kanan pada **tblBuku**, pilih Events → Mouse → MouseClicked. Tambahkan kode berikut ini:

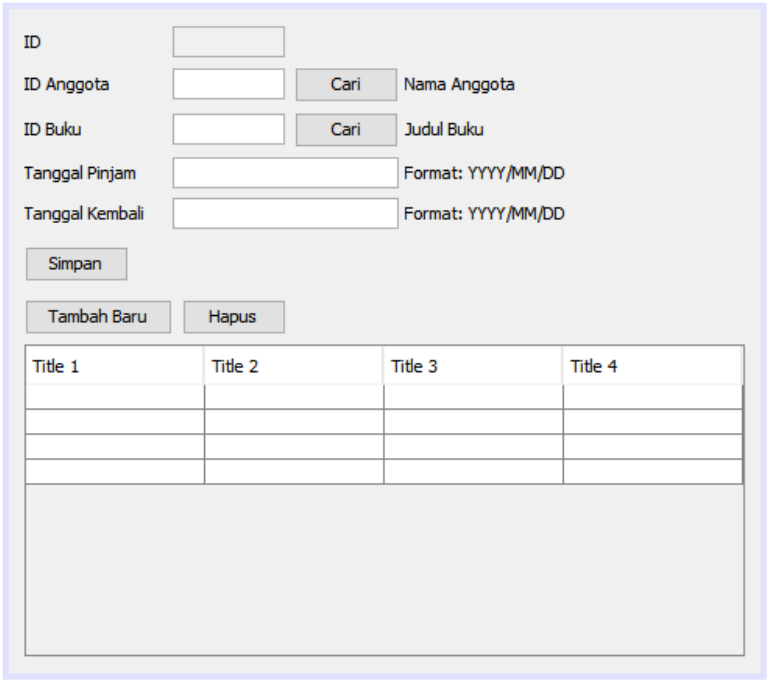
```
private void tblBukuMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel model = (DefaultTableModel)tblBuku.getModel();
    int row = tblBuku.getSelectedRow();
    Buku buku = new Buku();

    buku = buku.getById(Integer.parseInt(model.getValueAt(row, 0).toString()));

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

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

```java
package backend;

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

public class Buku
{
    private int idbuku;
    private Kategori kategori = new Kategori();
    private String judul;
    private String penerbit;
    private String penulis;

    public int getIdbuku() {
        return idbuku;
    }

    public void setIdbuku(int idbuku) {
        this.idbuku = idbuku;
    }

    public Kategori getKategori() {
        return kategori;
    }

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

    public String getJudul() {
        return judul;
    }

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

    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 Buku()
    {

    }

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

```
67      public Buku getById(int id)
68      {
69          Buku buku = new Buku();
70          ResultSet rs = DBHelper.selectQuery("SELECT "
71                                              + "     b.idbuku AS idbuku, "
72                                              + "     b.judul AS judul, "
73                                              + "     b.penerbit AS penerbit, "
74                                              + "     b.penulis AS penulis, "
75                                              + "     k.idkategori AS idkategori, "
76                                              + "     k.nama AS nama, "
77                                              + "     k.keterangan AS keterangan "
78                                              + "     FROM buku b "
79                                              + "     LEFT JOIN kategori k ON b.idkategori = k.idkategori "
80                                              + "     WHERE b.idbuku = '" + id + "'");
81
82          try
83          {
84              while(rs.next())
85              {
86                  buku = new Buku();
87                  buku.setIdbuku(rs.getInt("idbuku"));
88                  buku.getKategori().setIdkategori(rs.getInt("idkategori"));
89                  buku.getKategori().setNama(rs.getString("nama"));
90                  buku.getKategori().setKeterangan(rs.getString("keterangan"));
91                  buku.setJudul(rs.getString("judul"));
92                  buku.setPenerbit(rs.getString("penerbit"));
93                  buku.setPenulis(rs.getString("penulis"));
94              }
95          }
96          catch (Exception e)
97          {
98              e.printStackTrace();
99          }
100
101         return buku;
102     }
103
104     public ArrayList<Buku> getAll()
105     {
106         ArrayList<Buku> ListBuku = new ArrayList();
107
108         ResultSet rs = DBHelper.selectQuery("SELECT "
109                                             + "     b.idbuku AS idbuku, "
110                                             + "     b.judul AS judul, "
111                                             + "     b.penerbit AS penerbit, "
112                                             + "     b.penulis AS penulis, "
113                                             + "     k.idkategori AS idkategori, "
114                                             + "     k.nama AS nama, "
115                                             + "     k.keterangan AS keterangan "
116                                             + "     FROM buku b "
117                                             + "     LEFT JOIN kategori k ON b.idkategori = k.idkategori ");
118
119         try
120         {
121             while(rs.next())
122             {
123                 Buku buku = new Buku();
124                 buku.setIdbuku(rs.getInt("idbuku"));
125                 buku.getKategori().setIdkategori(rs.getInt("idkategori"));
126                 buku.getKategori().setNama(rs.getString("nama"));
127                 buku.getKategori().setKeterangan(rs.getString("keterangan"));
128                 buku.setJudul(rs.getString("judul"));
129                 buku.setPenerbit(rs.getString("penerbit"));
130                 buku.setPenulis(rs.getString("penulis"));
131
132                 ListBuku.add(buku);
133             }
134         }
135         catch (Exception e)
136         {
137             e.printStackTrace();
138         }
139
140         return ListBuku;
141     }
```

```
    public ArrayList<Buku> search(String keyword)
    {
        ArrayList<Buku> ListBuku = new ArrayList();

        ResultSet rs = DBHelper.selectQuery("SELECT "
                                          + "     b.idbuku AS idbuku, "
                                          + "     b.judul AS judul, "
                                          + "     b.penerbit AS penerbit, "
                                          + "     b.penulis AS penulis, "
                                          + "     k.idkategori AS idkategori, "
                                          + "     k.nama AS nama, "
                                          + "     k.keterangan AS keterangan "
                                          + "     FROM buku b "
                                          + "     LEFT JOIN kategori k ON b.idkategori = k.idkategori "
                                          + "     WHERE b.judul LIKE '%" + keyword + "%' "
                                          + "         OR b.penerbit LIKE '%" + keyword + "%' "
                                          + "         OR b.penulis LIKE '%" + keyword + "%' ");

        try
        {
            while(rs.next())
            {
                Buku buku = new Buku();
                buku.setIdbuku(rs.getInt("idbuku"));
                buku.getKategori().setIdkategori(rs.getInt("idkategori"));
                buku.getKategori().setNama(rs.getString("nama"));
                buku.getKategori().setKeterangan(rs.getString("keterangan"));
                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 void save()
    {
        if(getById(idbuku).getIdbuku() == 0)
        {
            String SQL = "INSERT INTO buku (judul, idkategori, penulis, penerbit) VALUES("
                       + "     '" + this.judul + "', "
                       + "     '" + this.getKategori().getIdkategori() + "', "
                       + "     '" + this.penulis + "', "
                       + "     '" + this.penerbit + "' "
                       + "     )";
            this.idbuku = DBHelper.insertQueryGetId(SQL);
        }
        else
        {
            String SQL = "UPDATE buku SET "
                       + "     judul = '" + this.judul + "', "
                       + "     idkategori = '" + this.getKategori().getIdkategori() + "', "
                       + "     penulis = '" + this.penulis + "', "
                       + "     penerbit = '" + this.penerbit + "' "
                       + "     WHERE idbuku = '" + this.idbuku + "'";
            DBHelper.executeQuery(SQL);
        }
    }

    public void delete()
    {
        String SQL = "DELETE FROM buku WHERE idbuku = '" + this.idbuku + "'";
        DBHelper.executeQuery(SQL);
    }
}
```

Lampiran 2. Kode untuk test class Buku. Silahkan di COPAS.

```
import backend.*;

public class TestBackend
{
    public static void main(String[] args)
    {
        Kategori novel = new Kategori().getById(27);
        Kategori referensi = new Kategori().getById(28);

        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();

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

        // test delete
        buku3.delete();

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

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