LISTING PROGRAM

Database.java

```
package skripsi;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JOptionPane;

public class Database {
   private Connection koneksi;
   private String database = "skripsi";
   private String dbUser = "root";
   private String dbPass = "root";
   private String username;

 public String getUsername() {
       return username;
   }

public Connection konek(){
       try{
          Class.forName("com.mysql.jdbc.Driver");
       }catch(ClassNotFoundException cnfe){
          System.out.println("Tidak ada Driver "+cnfe);
       }

       try{
          koneksi=DriverManager.getConnection("jdbc:mysql://localhost/"+database, dbUser,
dbPass);
       }catch(SQLException se){
          System.err.println("Gagal koneksi: "+se);
          JOptionPane.showMessageDialog(null,"Gagal Koneksi
Database","Peringatan",JOptionPane.WARNING_MESSAGE);
       }
       return koneksi;
   }

   public Object[][] validasikrit(){
   koneksi = new Database().konek();
   Object data[][] = null;
   try{
```

```
        String sql = "SELECT sum(bobot_kriteria)as bobot FROM kriteria";
        Statement stmt = koneksi.createStatement();
        ResultSet rslt = stmt.executeQuery(sql);
        ResultSetMetaData metadata = rslt.getMetaData();
        rslt.last();
        data = new Object[rslt.getRow()][metadata.getColumnCount()];

        int j = 0;
        rslt.beforeFirst();
        while(rslt.next()){
           data[j][0] = rslt.getInt("bobot");

        }
        koneksi.close();
    }catch(Exception ex){
        System.err.println("Gagal Mengambil Data User: "+ex.getMessage());
    }
    return data;
   }

    public String login(String username, String password){
    koneksi = new Database().konek();
    String posisi = "";
    int hasil = 0;
        try{
           String sql = "SELECT * FROM User WHERE Username = ? AND Password = ?";
           PreparedStatement stmt = koneksi.prepareStatement(sql);
           stmt.setString(1, username);
           stmt.setString(2, password);
           ResultSet rslt = stmt.executeQuery();
           while(rslt.next()){
              this.username = rslt.getString("username");
              posisi = rslt.getString("posisi");
              hasil = rslt.getRow();
           }
        koneksi.close();
        }catch(Exception ex){
           System.err.println("Gagal Login: "+ex.getMessage());
        }
    return posisi;
}

public Object[][] getUser(){
    koneksi = new Database().konek();
    Object data[][] = null;
    try{
        String sql = "SELECT * FROM User ORDER BY id_user";
```

```
        Statement stmt = koneksi.createStatement();
        ResultSet rslt = stmt.executeQuery(sql);
        ResultSetMetaData metadata = rslt.getMetaData();
        rslt.last();
        data = new Object[rslt.getRow()][metadata.getColumnCount()];

        int j = 0;
        rslt.beforeFirst();
        while(rslt.next()){
           data[j][0] = rslt.getInt("id_user");
           data[j][1] = rslt.getString("username");
           data[j][2] = rslt.getString("password");
           data[j][3] = rslt.getString("posisi");

           j++;
        }
        koneksi.close();
    }catch(Exception ex){
        System.err.println("Gagal Mengambil Data User: "+ex.getMessage());
    }
    return data;
}
public boolean insertUser(Object data[]){
    koneksi = new Database().konek();
    try {
        String sql = "INSERT INTO User (username, password,posisi) "
              + "VALUES (?, ?, 'perangkat desa')";
        PreparedStatement prep = koneksi.prepareStatement(sql);
        prep.setString(1, (String)data[1]);
        prep.setString(2, (String)data[2]);
        prep.setString(3, (String)data[0]);
        prep.execute();
        koneksi.close();
        return true;
    } catch(Exception ex){
        System.err.println("Gagal Menambah Data User: "+ex.getMessage());
        return false;
    }
}

public boolean updateUser(Object data[]){
    koneksi = new Database().konek();
    try{
        String sql = "UPDATE User SET username = ?, password = ?"
              + "WHERE id_user = ?";
        PreparedStatement prep = koneksi.prepareStatement(sql);
        prep.setString(1, (String) data[1]);
```

```
            prep.setString(2, (String) data[2]);
            prep.setInt(3, (Integer) data[0]);
            prep.execute();
            koneksi.close();
            return true;
        }catch(Exception ex){
            System.err.println("Gagal Mengubah Data User: "+ex.getMessage());
            return false;
        }
    }

    public boolean deleteUser(int id){
        koneksi = new Database().konek();
        try{
            String sql = "DELETE FROM User WHERE id_user = ?";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            prep.setInt(1, id);
            prep.execute();
            koneksi.close();
            return true;
        }catch(Exception ex){
            System.err.println("Gagal Menghapus Data User: "+ex.getMessage());
            return false;
        }
    }

    public Object[][] getPenduduk(){
        koneksi = new Database().konek();
        Object data[][] = null;
        try{
            String sql = "SELECT * FROM penduduk ORDER BY id";
            Statement stmt = koneksi.createStatement();
            ResultSet rslt = stmt.executeQuery(sql);
            ResultSetMetaData metadata = rslt.getMetaData();
            rslt.last();
            data = new Object[rslt.getRow()][metadata.getColumnCount()];
            int j = 0;
            rslt.beforeFirst();
            while(rslt.next()){
                data[j][0] = rslt.getInt("id");
                data[j][1] = rslt.getString("no_kk");
                data[j][2] = rslt.getString("Nama_KK");
                data[j][3] = rslt.getInt("RT");
                data[j][4] = rslt.getInt("RW");
                j++;
            }
            koneksi.close();
```

```
    }catch(Exception ex){
      System.err.println("Gagal Mengambil Data Penduduk: "+ex.getMessage());
    }
    return data;
}

 public Object[][] getPenduduka(){
    koneksi = new Database().konek();
    Object data[][] = null;
    try{
      String sql = "SELECT * FROM penduduk where id not in (select distinct  id from nilai)";
      Statement stmt = koneksi.createStatement();
      ResultSet rslt = stmt.executeQuery(sql);
      ResultSetMetaData metadata = rslt.getMetaData();
      rslt.last();
      data = new Object[rslt.getRow()][metadata.getColumnCount()];
      int j = 0;
      rslt.beforeFirst();
      while(rslt.next()){
        data[j][0] = rslt.getInt("id");
        data[j][1] = rslt.getString("no_kk");
        data[j][2] = rslt.getString("Nama_KK");
        data[j][3] = rslt.getInt("RT");
        data[j][4] = rslt.getInt("RW");
        j++;
      }
      koneksi.close();
    }catch(Exception ex){
      System.err.println("Gagal Mengambil Data Penduduk: "+ex.getMessage());
    }
    return data;
}

  public Object[][] getPendudukb(){
    koneksi = new Database().konek();
    Object data[][] = null;
    try{
      String sql = "SELECT * FROM penduduk where id in (select distinct  id from nilai)";
      Statement stmt = koneksi.createStatement();
      ResultSet rslt = stmt.executeQuery(sql);
      ResultSetMetaData metadata = rslt.getMetaData();
      rslt.last();
      data = new Object[rslt.getRow()][metadata.getColumnCount()];
      int j = 0;
      rslt.beforeFirst();
      while(rslt.next()){
        data[j][0] = rslt.getInt("id");
```

```
            data[j][1] = rslt.getString("no_kk");
            data[j][2] = rslt.getString("Nama_KK");
            data[j][3] = rslt.getInt("RT");
            data[j][4] = rslt.getInt("RW");
            j++;
          }
          koneksi.close();
        }catch(Exception ex){
          System.err.println("Gagal Mengambil Data Penduduk: "+ex.getMessage());
        }
        return data;
    }

     public Object[][] getPendudukc(){
        koneksi = new Database().konek();
        Object data[][] = null;
        try{
          String sql = "SELECT p.id,p.no_kk,p.Nama_KK,p.RT,p.RW,date_format(now(),'%M,%Y')
as tanggal FROM penduduk p where p.id in (select distinct  id from nilai)";
          Statement stmt = koneksi.createStatement();
          ResultSet rslt = stmt.executeQuery(sql);
          ResultSetMetaData metadata = rslt.getMetaData();
          rslt.last();
          data = new Object[rslt.getRow()][metadata.getColumnCount()];
          int j = 0;
          rslt.beforeFirst();
          while(rslt.next()){
            data[j][0] = rslt.getInt("id");
            data[j][1] = rslt.getString("no_kk");
            data[j][2] = rslt.getString("Nama_KK");
            data[j][3] = rslt.getInt("RT");
            data[j][4] = rslt.getInt("RW");
            data[j][5] = rslt.getString("tanggal");
            j++;
          }
          koneksi.close();
        }catch(Exception ex){
          System.err.println("Gagal Mengambil Data Penduduk: "+ex.getMessage());
        }
        return data;
    }


    public Object[][] getPenduduk(int id){
        koneksi = new Database().konek();
        Object data[][] = null;
        try{
```

```
        String sql = "SELECT * FROM penduduk WHERE id = ? ORDER BY id";
        PreparedStatement prep = koneksi.prepareStatement(sql);
        prep.setInt(1, id);

        ResultSet rslt = prep.executeQuery();
        ResultSetMetaData metadata = rslt.getMetaData();
        rslt.last();
        data = new Object[rslt.getRow()][metadata.getColumnCount()];

        int j = 0;
        rslt.beforeFirst();
        while(rslt.next()){
           data[j][0] = rslt.getInt("id");
           data[j][1] = rslt.getString("no_kk");
           data[j][2] = rslt.getString("Nama_KK");
           data[j][3] = rslt.getInt("RT");
           data[j][4] = rslt.getInt("RW");
           j++;
        }
        koneksi.close();
    }catch(Exception ex){
        System.err.println("Gagal Mengambil Data Penduduk: "+ex.getMessage());
    }
    return data;
}

public boolean insertPenduduk(Object data[]){
    koneksi = new Database().konek();
    try {
        String sql = "INSERT INTO penduduk (no_kk,Nama_KK,RT,RW) "
              + "VALUES (?, ?, ?,?)";
        PreparedStatement prep = koneksi.prepareStatement(sql);

        prep.setString(1, (String)data[1]);
        prep.setString(2, (String)data[2]);
        prep.setInt(3, (Integer)data[3]);
        prep.setInt(4, (Integer)data[4]);

        prep.execute();
        koneksi.close();
        return true;
    } catch(Exception ex){
        System.err.println("Gagal Menambah Data Penduduk: "+ex.getMessage());
        return false;
    }
}
```

```
public boolean updatePenduduk(Object data[]){
   koneksi = new Database().konek();
   try{
      String sql = "UPDATE penduduk SET no_kk= ?, nama_KK = ?, RT = ?, RW = ? "
            + "WHERE id = ?";
      PreparedStatement prep = koneksi.prepareStatement(sql);
      prep.setString(1, (String) data[1]);
      prep.setString(2, (String) data[2]);
      prep.setInt(3, (Integer) data[3]);
      prep.setInt(4, (Integer) data[4]);
      prep.setInt(5, (Integer) data[0]);
      prep.execute();
      koneksi.close();
      return true;
   }catch(Exception ex){
      System.err.println("Gagal Mengubah Data Penduduk: "+ex.getMessage());
      return false;
   }
}

public boolean deletePenduduk(int id){
   koneksi = new Database().konek();
   try{
      String sql = "DELETE FROM penduduk WHERE id = ?";
      PreparedStatement prep = koneksi.prepareStatement(sql);
      prep.setInt(1, id);
      prep.execute();
      koneksi.close();
      return true;
   }catch(Exception ex){
      System.err.println("Gagal Menghapus Data penduduk: "+ex.getMessage());
      return false;
   }
}

public Object[][] getKriteria(){
   koneksi = new Database().konek();
   Object data[][] = null;
   try{
      String sql = "SELECT * FROM kriteria ORDER BY kode_kriteria";
      Statement stmt = koneksi.createStatement();
      ResultSet rslt = stmt.executeQuery(sql);
      ResultSetMetaData metadata = rslt.getMetaData();
      rslt.last();
      data = new Object[rslt.getRow()][metadata.getColumnCount()];
```

```
        int j = 0;
        rslt.beforeFirst();
        while(rslt.next()){
            data[j][0] = rslt.getInt("kode_kriteria");
            data[j][1] = rslt.getString("nama_kriteria");
            data[j][2] = rslt.getInt("bobot_kriteria");

            j++;
        }
        koneksi.close();
    }catch(Exception ex){
        System.err.println("Gagal Mengambil Data Kriteria: "+ex.getMessage());
    }
    return data;
}

public boolean insertKriteria(Object data[]){
    koneksi = new Database().konek();
    try {
        String sql = "INSERT INTO kriteria (nama_kriteria, bobot_kriteria) "
                + "VALUES (?, ?)";
        PreparedStatement prep = koneksi.prepareStatement(sql);
        prep.setString(1, (String)data[1]);
        prep.setInt(2, (Integer)data[2]);

        prep.execute();
        koneksi.close();
        return true;
    } catch(Exception ex){
        System.err.println("Gagal Menambah Data Kriteria: "+ex.getMessage());
        return false;
    }
}

public boolean updateKriteria(Object data[]){
    koneksi = new Database().konek();
    try{
        String sql = "UPDATE kriteria SET nama_kriteria = ?, bobot_kriteria = ? "
                + "WHERE kode_kriteria = ?";
        PreparedStatement prep = koneksi.prepareStatement(sql);
        prep.setString(1, (String) data[1]);
        prep.setInt(2, (Integer) data[2]);
        prep.setInt(3, (Integer) data[0]);
        prep.execute();

        koneksi.close();
        return true;
```

```
        }catch(Exception ex){
            System.err.println("Gagal Mengubah Data Kriteria: "+ex.getMessage());
            return false;
        }
    }

    public boolean deleteKriteria(int id){
        koneksi = new Database().konek();
        try{
            String sql = "DELETE FROM kriteria WHERE kode_kriteria = ?";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            prep.setInt(1, id);
            prep.execute();
            koneksi.close();
            return true;
        }catch(Exception ex){
            System.err.println("Gagal Menghapus Data Kriteria: "+ex.getMessage());
            return false;
        }
    }

    public Object[][] getSubKriteria(){
        koneksi = new Database().konek();
        Object data[][] = null;
        try{
            String sql = "SELECT s.kode_sub_kriteria, s.nama_sub_kriteria, s.nilai_profil,
f.kode_faktor, f. nama_faktor, f.bobot_faktor, "
                    + "k.kode_kriteria, k.nama_kriteria, k.bobot_kriteria "
                    + "FROM sub_kriteria s LEFT JOIN faktor AS f ON s.kode_faktor = f.kode_faktor "
                    + "LEFT JOIN kriteria AS k ON s.kode_kriteria = k.kode_kriteria ORDER BY
s.kode_sub_kriteria";
            Statement stmt = koneksi.createStatement();
            ResultSet rslt = stmt.executeQuery(sql);
            ResultSetMetaData metadata = rslt.getMetaData();
            rslt.last();
            data = new Object[rslt.getRow()][metadata.getColumnCount()];

            int j = 0;
            rslt.beforeFirst();
            while(rslt.next()){
                data[j][0] = rslt.getInt("kode_sub_kriteria");
                data[j][1] = rslt.getString("nama_sub_kriteria");
                data[j][2] = rslt.getInt("nilai_profil");
                data[j][3] = rslt.getInt("kode_faktor");
                data[j][4] = rslt.getString("nama_faktor");
                data[j][5] = rslt.getInt("bobot_faktor");
                data[j][6] = rslt.getInt("kode_kriteria");
```

```
            data[j][7] = rslt.getString("nama_kriteria");
            data[j][8] = rslt.getInt("bobot_kriteria");

            j++;
          }
          koneksi.close();
        }catch(Exception ex){
          System.err.println("Gagal Mengambil Data Sub Kriteria: "+ex.getMessage());
        }
        return data;
    }

    public Object[][] getSubKriteria(int kodeKriteria){
        koneksi = new Database().konek();
        Object data[][] = null;
        try{
          String sql = "SELECT s.kode_sub_kriteria, s.nama_sub_kriteria, s.nilai_profil,
f.kode_faktor, f.nama_faktor, f.bobot_faktor, "
                + "k.kode_kriteria, k.nama_kriteria, k.bobot_kriteria "
                + "FROM sub_kriteria s LEFT JOIN faktor AS f ON s.kode_faktor = f.kode_faktor "
                + "LEFT JOIN kriteria AS k ON s.kode_kriteria = k.kode_kriteria "
                + "WHERE k.kode_kriteria = ? ORDER BY s.kode_sub_kriteria";
          PreparedStatement prep = koneksi.prepareStatement(sql);
          prep.setInt(1, kodeKriteria);
          ResultSet rslt = prep.executeQuery();
          ResultSetMetaData metadata = rslt.getMetaData();
          rslt.last();
          data = new Object[rslt.getRow()][metadata.getColumnCount()];

          int j = 0;
          rslt.beforeFirst();
          while(rslt.next()){
            data[j][0] = rslt.getInt("kode_sub_kriteria");
            data[j][1] = rslt.getString("nama_sub_kriteria");
            data[j][2] = rslt.getInt("nilai_profil");
            data[j][3] = rslt.getInt("kode_faktor");
            data[j][4] = rslt.getString("nama_faktor");
            data[j][5] = rslt.getString("bobot_faktor");
            data[j][6] = rslt.getInt("kode_kriteria");
            data[j][7] = rslt.getString("nama_kriteria");
            data[j][8] = rslt.getString("bobot_kriteria");

            j++;
          }
          koneksi.close();
        }catch(Exception ex){
          System.err.println("Gagal Mengambil Data Sub Kriteria: "+ex.getMessage());
```

```
        }
        return data;
    }

    public Object[][] getLaporanKrit(String kodeKriteria){
        koneksi = new Database().konek();
        Object data[][] = null;
        try{
          String sql = "select k.nama_kriteria,s.nama_sub_kriteria,f.nama_faktor,f.bobot_faktor,
k.bobot_kriteria, s.nilai_profil from kriteria k, sub_kriteria s, faktor f where k.kode_kriteria =
s.kode_kriteria and f.kode_faktor = s.kode_faktor and k.nama_kriteria = ?";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            prep.setString(1, kodeKriteria);
            ResultSet rslt = prep.executeQuery();
            ResultSetMetaData metadata = rslt.getMetaData();
            rslt.last();
            data = new Object[rslt.getRow()][metadata.getColumnCount()];
            int j = 0;
            rslt.beforeFirst();
            while(rslt.next()){
               data[j][0] = rslt.getString("nama_kriteria");
               data[j][1] = rslt.getString("nama_sub_kriteria");
               data[j][2] = rslt.getString("nama_faktor");
               data[j][3] = rslt.getInt("bobot_faktor");
               data[j][4] = rslt.getInt("bobot_kriteria");
               data[j][5] = rslt.getInt("nilai_profil");
               j++;
            }
            koneksi.close();
        }catch(Exception ex){
            System.err.println("Gagal Mengambil Data Sub Kriteria: "+ex.getMessage());
        }
        return data;
    }

    public boolean insertSubKriteria(Object data[]){
        koneksi = new Database().konek();
        try {
            String sql = "INSERT INTO sub_kriteria (nama_sub_kriteria,
nilai_profil,kode_kriteria,kode_faktor) "
                    + "VALUES (?, ?, ?, ?)";
            PreparedStatement prep = koneksi.prepareStatement(sql);

            prep.setString(1, (String)data[1]);
            prep.setInt(2, (Integer)data[2]);
            prep.setInt(3, (Integer)data[3]);
            prep.setInt(4, (Integer)data[4]);
```

```
            prep.execute();
            koneksi.close();
            return true;
        } catch(Exception ex){
            System.err.println("Gagal Menambah Data Sub Kriteria: "+ex.getMessage());
            return false;
        }
    }

    public boolean updateSubKriteria(Object data[]){
        koneksi = new Database().konek();
        try{
            String sql = "UPDATE sub_kriteria SET nama_sub_kriteria = ?, nilai_profil = ?,
kode_faktor = ?, kode_kriteria = ? "
                    + "WHERE kode_sub_kriteria = ?";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            prep.setInt(1, (Integer) data[1]);
            prep.setString(2, (String) data[2]);
            prep.setInt(3, (Integer) data[3]);
            prep.setInt(4, (Integer) data[4]);
            prep.setInt(5, (Integer) data[0]);
            prep.execute();

            koneksi.close();
            return true;
        }catch(Exception ex){
            System.err.println("Gagal Mengubah Data Sub Kriteria: "+ex.getMessage());
            return false;
        }
    }

    public boolean deleteSubKriteria(int id){
        koneksi = new Database().konek();
        try{
            String sql = "DELETE FROM sub_kriteria WHERE kode_sub_kriteria = ?";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            prep.setInt(1, id);
            prep.execute();
            koneksi.close();
            return true;
        }catch(Exception ex){
            System.err.println("Gagal Menghapus Data Sub Kriteria: "+ex.getMessage());
            return false;
        }
    }
    public Object[][] getNilai(){
```

```
    koneksi = new Database().konek();
    Object data[][] = null;
    try{
      String sql = "SELECT * FROM nilai n, sub_kriteria s "
            + "WHERE n.kode_sub_kriteria = s.kode_sub_kriteria "
            + "ORDER BY n.kode_sub_kriteria AND n.id ASC";
      Statement stmt = koneksi.createStatement();
      ResultSet rslt = stmt.executeQuery(sql);
      rslt.last();
      data = new Object[rslt.getRow()][11];

      int j = 0;
      rslt.beforeFirst();
      while(rslt.next()){
        data[j][0] = rslt.getInt("kode_nilai");
        data[j][1] = rslt.getInt("nilai");
        data[j][2] = rslt.getInt("kode_sub_kriteria");
        data[j][3] = rslt.getInt("id");
        data[j][4] = rslt.getInt("nilai_profil");
        data[j][5] = rslt.getInt("kode_faktor");
        data[j][6] = rslt.getInt("kode_kriteria");

        j++;
      }
      koneksi.close();
    }catch(Exception ex){
      System.err.println("Gagal Mengambil Data Nilai: "+ex.getMessage());
    }
    return data;
}

public Object[][] getNilaiLaporan(){
    koneksi = new Database().konek();
    Object data[][] = null;
    try{
      String sql = "SELECT * FROM sub_kriteria s "
            + "INNER JOIN kriteria k ON s.kode_kriteria = k.kode_kriteria "
            + "INNER JOIN faktor f ON s.id_faktor = f.kode_faktor "
            + "ORDER BY k.kode_kriteria";
      Statement stmt = koneksi.createStatement();
      ResultSet rslt = stmt.executeQuery(sql);
      rslt.last();
      data = new Object[rslt.getRow()][11];

      int j = 0;
      rslt.beforeFirst();
      while(rslt.next()){
```

```
            data[j][0] = rslt.getInt("kode_kriteria");
            data[j][1] = rslt.getInt("kode_sub_kriteria");
            data[j][2] = rslt.getInt("kode_faktor");
            data[j][3] = rslt.getString("nama_kriteria");
            data[j][4] = rslt.getString("nama_sub_kriteria");
            data[j][5] = rslt.getString("nama_faktor");
            data[j][6] = rslt.getInt("nilai_profil");

            j++;
        }
        koneksi.close();
    }catch(Exception ex){
        System.err.println("Gagal Mengambil Data Nilai: "+ex.getMessage());
    }
    return data;
}

public Object[][] getSFaktor(){
    koneksi =new Database().konek();
    Object data[][] = null;
    try{
        String sql= "select sum(bobot_faktor) as jumlah from faktor";
       PreparedStatement stmt = koneksi.prepareStatement(sql);
        stmt.execute();
        ResultSet rslt = stmt.executeQuery();
        rslt.last();
        data = new Object[rslt.getRow()][1];
        int j = 0;
        rslt.beforeFirst();
        while(rslt.next()){
            data[j][0] = rslt.getInt("jumlah");
            j++;
    }
    koneksi.close();
    }catch(Exception ex){
        System.err.println("Gagal Mengambil Data Nilai: "+ex.getMessage());
    }
    return data;
}

public Object[][] getSKriteria(){
    koneksi =new Database().konek();
    Object data[][] = null;
    try{
        String sql= "select sum(bobot_kriteria) as jumlah from kriteria";
       PreparedStatement stmt = koneksi.prepareStatement(sql);
        stmt.execute();
```

```
        ResultSet rslt = stmt.executeQuery();
        rslt.last();
        data = new Object[rslt.getRow()][1];
        int j = 0;
        rslt.beforeFirst();
        while(rslt.next()){
            data[j][0] = rslt.getInt("jumlah");
            j++;
    }
    koneksi.close();
    }catch(Exception ex){
        System.err.println("Gagal Mengambil Data Nilai: "+ex.getMessage());
    }
    return data;
}

public Object[][] getNilaiPenduduk(int kodePenduduk){
    koneksi = new Database().konek();
    Object data[][] = null;
    try{
        String sql = "SELECT * FROM nilai n, sub_kriteria s "
                + "WHERE n.kode_sub_kriteria = s.kode_sub_kriteria "
                + "AND n.id = ? "
                + "ORDER BY n.kode_sub_kriteria";
        PreparedStatement stmt = koneksi.prepareStatement(sql);
        stmt.setInt(1, kodePenduduk);
        stmt.execute();
        ResultSet rslt = stmt.executeQuery();
        rslt.last();
        data = new Object[rslt.getRow()][8];

        int j = 0;
        rslt.beforeFirst();
        while(rslt.next()){
            data[j][0] = rslt.getInt("kode_nilai");
            data[j][1] = rslt.getInt("nilai");
            data[j][2] = rslt.getInt("kode_sub_kriteria");
            data[j][3] = rslt.getInt("id");
            data[j][4] = rslt.getInt("nilai_profil");
            data[j][5] = rslt.getInt("kode_faktor");
            data[j][6] = rslt.getInt("kode_kriteria");
            data[j][7]= rslt.getString("nama_sub_kriteria");

            j++;
        }
        koneksi.close();
    }catch(Exception ex){
```

```
            System.err.println("Gagal Mengambil Data Nilai: "+ex.getMessage());
        }
        return data;
    }

    public int getNilai(int kodeSubKriteria, int kodePenduduk, int kodeOutput){
        koneksi = new Database().konek();
        int hasil = 0;
        try{
            String sql = "SELECT * FROM nilai n, sub_kriteria s "
                    + "WHERE n.id_sub_kriteria = s.id_sub_kriteria "
                    + "AND s.id_sub_kriteria = ? AND n.id = ? "
                    + "ORDER BY n.id_sub_kriteria";
            PreparedStatement stmt = koneksi.prepareStatement(sql);
            stmt.setInt(1, kodeSubKriteria);
            stmt.setInt(2, kodePenduduk);
            ResultSet rslt = stmt.executeQuery();

            while(rslt.next()){
                if (kodeOutput == 0)
                    hasil = rslt.getInt("nilai");
                else if (kodeOutput == 1)
                    hasil = rslt.getInt("nilai_profil");
            }
            koneksi.close();
        }catch(Exception ex){
            System.err.println("Gagal Mengambil Data Nilai: "+ex.getMessage());
        }
        return hasil;
    }

    public int[] getNilai(String kriteria, String subKriteria, String penduduk){
        koneksi = new Database().konek();
        int hasil[] = {0, 0, 0, 0};
        try{
            String sql = "SELECT n.kode_nilai, n.nilai, s.kode_sub_kriteria, s.nama_sub_kriteria,
s.nilai_profil, "
                    + "f.kode_faktor, f.nama_faktor, f.bobot_faktor,  k.kode_kriteria,
k.nama_kriteria, k.bobot_kriteria, "
                    + "p.id, p.no_kk "
                    + "FROM nilai n, sub_kriteria s, penduduk p, faktor f, kriteria k "
                    + "WHERE n.id = p.id "
                    + "AND n.kode_sub_kriteria = s.kode_sub_kriteria "
                    + "AND s.kode_faktor = f.kode_faktor AND k.kode_kriteria = s.kode_kriteria "
                    + "AND k.nama_kriteria = ? AND s.nama_sub_kriteria = ? AND p.no_kk = ? "
                    + "ORDER BY n.kode_nilai";
            PreparedStatement stmt = koneksi.prepareStatement(sql);
```

```
            stmt.setString(1, kriteria);
            stmt.setString(2, subKriteria);
            stmt.setString(3, penduduk);
            ResultSet rslt = stmt.executeQuery();
            while(rslt.next()){
               hasil[0] = rslt.getInt("kode_nilai");
               hasil[1] = rslt.getInt("id");
               hasil[2] = rslt.getInt("kode_sub_kriteria");
               hasil[3] = rslt.getInt("nilai");
            }
         }catch(Exception ex){
            System.err.println("Gagal Mengambil Data Nilai: "+ex.getMessage());
         }
         return hasil;
      }

      public boolean insertNilai(Object data[]){
         koneksi = new Database().konek();
         try {
            String sql = "INSERT INTO nilai (id, kode_sub_kriteria, nilai) "
                  + "VALUES (?, ?, ?)";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            prep.setInt(1, (Integer)data[1]);
            prep.setInt(2, (Integer)data[2]);
            prep.setInt(3, (Integer)data[3]);
            prep.execute();
            koneksi.close();
            return true;
         } catch(Exception ex){
            System.err.println("Gagal Menambah Data Nilai: "+ex.getMessage());
            return false;
         }   }
      public boolean updateNilai(Object data[]){
         koneksi = new Database().konek();
         try{
            String sql = "UPDATE nilai SET id = ?, kode_sub_kriteria = ?, nilai = ? "
                  + "WHERE kode_nilai = ?";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            prep.setInt(1, (Integer) data[1]);
            prep.setInt(2, (Integer) data[2]);
            prep.setInt(3, (Integer) data[3]);
            prep.setInt(4, (Integer) data[0]);
            prep.execute();

            System.out.println("UPDATE nilai SET id = "+data[1]+", kode_sub_kriteria =
"+data[2]+", nilai = "+data[3]+" "
                  + "WHERE kode_nilai = "+data[0]);
```

```
            koneksi.close();
            return true;
        }catch(Exception ex){
            System.err.println("Gagal Mengubah Data Nilai: "+ex.getMessage());
            return false;
        }
    }
    public boolean deleteNilai(int id){
        koneksi = new Database().konek();
        try{
            String sql = "DELETE FROM nilai WHERE kode_nilai = ?";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            prep.setInt(1, id);
            prep.execute();
            koneksi.close();
            return true;
        }catch(Exception ex){
            System.err.println("Gagal Menghapus Data Nilai: "+ex.getMessage());
            return false;
        }
    }
    public boolean deleteNilai(int kodeKriteria, int kodePelamar) {
        koneksi = new Database().konek();
        try{
            String sql = "DELETE FROM nilai WHERE id = ? "
                    + "AND kode_sub_kriteria IN "
                    + "(SELECT s.kode_sub_kriteria FROM sub_kriteria s "
                    + "INNER JOIN kriteria k ON k.kode_kriteria = s.kode_kriteria "
                    + "AND k.kode_kriteria = ?)";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            prep.setInt(1, kodePelamar);
            prep.setInt(2, kodeKriteria);
            prep.execute();
            koneksi.close();
            return true;
        }catch(Exception ex){
            System.err.println("Gagal Menghapus Data Nilai: "+ex.getMessage());
            return false;
        }
    }

    public Object[][] getFaktor(){
        koneksi = new Database().konek();
        Object data[][] = null;
        try{
            String sql = "SELECT * FROM faktor ORDER BY kode_faktor";
```

```
        Statement stmt = koneksi.createStatement();
        ResultSet rslt = stmt.executeQuery(sql);
        ResultSetMetaData metadata = rslt.getMetaData();
        rslt.last();
        data = new Object[rslt.getRow()][metadata.getColumnCount()];

        int j = 0;
        rslt.beforeFirst();
        while(rslt.next()){
            data[j][0] = rslt.getInt("kode_faktor");
            data[j][1] = rslt.getString("nama_faktor");
            data[j][2] = rslt.getInt("bobot_faktor");
            j++;
        }
        koneksi.close();
    }catch(Exception ex){
        System.err.println("Gagal Mengambil Data Faktor: "+ex.getMessage());
    }
    return data;
}
public boolean insertFaktor(Object data[]){
    koneksi = new Database().konek();
    try {
        String sql = "INSERT INTO faktor (nama_faktor, bobot_faktor) "
                + "VALUES (?, ?)";
        PreparedStatement prep = koneksi.prepareStatement(sql);
        prep.setString(1, (String)data[1]);
        prep.setInt(2, (Integer)data[2]);
        prep.execute();
        koneksi.close();
        return true;
    } catch(Exception ex){
        System.err.println("Gagal Menambah Data Faktor: "+ex.getMessage());
        return false;
    }
}
public boolean updateFaktor(Object data[]){
    koneksi = new Database().konek();
    try{
        String sql = "UPDATE faktor SET nama_faktor = ?,bobot_faktor = ? "
                + "WHERE kode_faktor = ?";
        PreparedStatement prep = koneksi.prepareStatement(sql);
        prep.setString(1, (String) data[1]);
        prep.setInt(2, (Integer) data[2]);
        prep.setInt(3, (Integer) data[0]);
        prep.execute();
```

```
            koneksi.close();
            return true;
        }catch(Exception ex){
            System.err.println("Gagal Mengubah Data Faktor: "+ex.getMessage());
            return false;
        }
    }
    public boolean deleteFaktor(int id){
        koneksi = new Database().konek();
        try{
            String sql = "DELETE FROM faktor WHERE kode_faktor = ?";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            prep.setInt(1, id);
            prep.execute();
            koneksi.close();
            return true;
        }catch(Exception ex){
            System.err.println("Gagal Menghapus Data Faktor: "+ex.getMessage());
            return false;
        }
    }
    public Object[][] getNilaiByVar(int kodeKriteria, int kodePenduduk, int kodeFaktor){
        koneksi = new Database().konek();
        Object data[][] = null;
        try{
            String sql = "SELECT * FROM sub_kriteria s "
                    + "INNER JOIN nilai n ON s.kode_sub_kriteria = n.kode_sub_kriteria "
                    + "INNER JOIN penduduk p ON p.id= n.id "
                    + "INNER JOIN kriteria k ON k.kode_kriteria = s.kode_kriteria "
                    + "INNER JOIN faktor f ON f.kode_faktor = s.kode_faktor "
                    + "WHERE s.kode_kriteria = ? AND p.id = ? AND f.kode_faktor = ?";
            PreparedStatement stmt = koneksi.prepareCall(sql);
            stmt.setInt(1, kodeKriteria);
            stmt.setInt(2, kodePenduduk);
            stmt.setInt(3, kodeFaktor);
            ResultSet rslt = stmt.executeQuery();
            ResultSetMetaData metadata = rslt.getMetaData();
            rslt.last();
            data = new Object[rslt.getRow()][13];
            int j = 0;
            rslt.beforeFirst();
            while(rslt.next()){
                data[j][0] = rslt.getInt("kode_sub_kriteria");
                data[j][1] = rslt.getInt("kode_nilai");
                data[j][2] = rslt.getInt("kode_kriteria");
                data[j][3] = rslt.getInt("id");
                data[j][4] = rslt.getInt("kode_faktor");
```

```
            data[j][5] = rslt.getString("Nama_KK");
            data[j][6] = rslt.getInt("nilai");
            data[j][7] = rslt.getString("nama_sub_kriteria");
            data[j][8] = rslt.getInt("nilai_profil");
            data[j][9] = rslt.getString("nama_kriteria");
            data[j][10] = rslt.getInt("bobot_kriteria");
            data[j][11] = rslt.getString("nama_faktor");
            data[j][12] = rslt.getInt("bobot_faktor");
            j++;
          }
          koneksi.close();
        }catch(Exception ex){
          System.err.println("Gagal Mengambil Data Nilai Hitung: "+ex.getMessage());
        }
        return data;
    }
        public boolean insertNilaiPenduduk(Object data[]){
        koneksi = new Database().konek();
        try {
          String sql = "INSERT INTO nilai_penduduk (id,no_kk, luas_lantai, "
                + "jenis_dinding, jenis_lantai, Penerangan, sarana_memasak, "
                + "Air_Minum, jumlah_penghasilan,
pendidikan,Sanggup_pakaian,Sanggup_kesehatan) "
                + "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
          PreparedStatement prep = koneksi.prepareStatement(sql);
          for (int i = 2; i < data.length; i++) {
            prep.setInt(1,(Integer)data[0]);
            prep.setString(2,data[1].toString());
            prep.setInt(i+1, (Integer)data[i]);
                    }
          prep.execute();
          koneksi.close();
          return true;
        } catch(Exception ex){
          System.err.println("Gagal Menambah Data Nilai Penduduk: "+ex.getMessage());
          return false;
        }
    }
    public boolean deleteNilaiAkhir(){
        koneksi = new Database().konek();
        try {
          String sql = "DELETE FROM nilai_akhir";
          Statement prep = koneksi.createStatement();
          boolean hasil = prep.execute(sql);
          koneksi.close();
          return hasil;
        } catch(Exception ex){
```

```
            System.err.println("Gagal Menambah Data Nilai Akhir: "+ex.getMessage());
            return false;
        }
    }
    public boolean deleteNilaiPenduduk(){
        koneksi = new Database().konek();
        try {
            String sql = "DELETE FROM nilai_penduduk";
            Statement prep = koneksi.createStatement();
            boolean hasil = prep.execute(sql);
            koneksi.close();
            return hasil;
        } catch(Exception ex){
            System.err.println("Gagal Hapus Data Nilai Penduduk: "+ex.getMessage());
            return false;
        }
    }
    public boolean deleteKP(){
        koneksi = new Database().konek();
        try {
            String sql = "DELETE FROM KP";
            Statement prep = koneksi.createStatement();
            boolean hasil = prep.execute(sql);
            koneksi.close();
            return hasil;
        } catch(Exception ex){
            System.err.println("Gagal Hapus Data Nilai Penduduk: "+ex.getMessage());
            return false;
        }
    }
  public boolean insertKP(Object data[]){
        koneksi = new Database().konek();
        try {
            String sql = "INSERT INTO KP (nama_kriteria,
nama_sub_kriteria,nama_faktor,bobot_faktor,bobot_kriteria,nilai_profil) "
                    + "VALUES (?, ?, ?, ?, ?, ?)";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            for (int i = 0; i < data.length;i++) {
                prep.setString(1,data[0].toString());
                prep.setString(2,data[1].toString());
                prep.setString(3,data[2].toString());
                prep.setInt(4,(Integer)data[3]);
                prep.setInt(5,(Integer)data[4]);
                prep.setInt(6,(Integer)data[5]);
             }
            prep.execute();
            koneksi.close();
```

```
            return true;
        } catch(Exception ex){
            System.err.println("Gagal Menambah Data Nilai Penduduk: "+ex.getMessage());
            return false;
        }
    }
  public boolean insertNA(Object data[]){
        koneksi = new Database().konek();
        try {
            String sql = "INSERT INTO nilai_akhir (no_kk, nilai_akhir, status, Periode) "
                    + "VALUES (?, ?, ?, ?)";
            PreparedStatement prep = koneksi.prepareStatement(sql);
            for (int i = 0; i < data.length;i++) {
               prep.setString(1,data[0].toString());
               prep.setString(2,data[1].toString());
               prep.setString(3,data[3].toString());
               prep.setString(4,data[2].toString());
              }
            prep.execute();
            koneksi.close();
            return true;
        } catch(Exception ex){
            System.err.println("Gagal Menambah Data Nilai Penduduk: "+ex.getMessage());
            return false;
        }
    }

public int[] getNilaiN(String kriteria, String subKriteria, int penduduk){
        koneksi = new Database().konek();
        int hasil[] = {0, 0, 0, 0};
        try{
            String sql = "SELECT n.kode_nilai, n.nilai, s.kode_sub_kriteria, s.nama_sub_kriteria,
s.nilai_profil, "
                    + "f.kode_faktor, f.nama_faktor, f.bobot_faktor,  k.kode_kriteria,
k.nama_kriteria, k.bobot_kriteria, "
                    + "p.id, p.no_kk "
                    + "FROM nilai n, sub_kriteria s, penduduk p, faktor f, kriteria k "
                    + "WHERE n.id = p.id "
                    + "AND n.kode_sub_kriteria = s.kode_sub_kriteria "
                    + "AND s.kode_faktor = f.kode_faktor AND k.kode_kriteria = s.kode_kriteria "
                    + "AND k.nama_kriteria = ? AND s.nama_sub_kriteria = ? AND p.id = ? "
                    + "ORDER BY n.kode_nilai";
            PreparedStatement stmt = koneksi.prepareStatement(sql);
            stmt.setString(1, kriteria);
            stmt.setString(2, subKriteria);
            stmt.setInt(3, penduduk);
            ResultSet rslt = stmt.executeQuery();
```

```
        while(rslt.next()){
            hasil[0] = rslt.getInt("kode_nilai");
            hasil[1] = rslt.getInt("id");
            hasil[2] = rslt.getInt("kode_sub_kriteria");
            hasil[3] = rslt.getInt("nilai");
        }
    }catch(Exception ex){
        System.err.println("Gagal Mengambil Data Nilai: "+ex.getMessage());
    }
    return hasil;
  } }
```

Table.java

```
package skripsi;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableColumn;
public class Table {
 public void createTable(JTable tabel, int lebar[], int tinggi){
      tabel.setAutoResizeMode(tabel.AUTO_RESIZE_OFF);
      int kolom = tabel.getColumnCount();
      for (int i = 0; i < kolom; i++) {
         TableColumn tabelKolom = tabel.getColumnModel().getColumn(i);
         tabelKolom.setPreferredWidth(lebar[i]);
         tabel.setRowHeight(tinggi);
      }   }
   public DefaultTableModel getDefaultTabelModel(String judul[]) {
      final boolean[] editable = new boolean[judul.length];
      for (int i = 0; i < editable.length; i++) {
         editable[i] = false;
      }
      return new DefaultTableModel(
            new Object[][]{},
            judul ){
         boolean canEdit[] = editable;

         @Override
         public boolean isCellEditable(int rowIndex, int columnIndex){
            return canEdit[columnIndex];
         }
      };
   }
   public DefaultTableModel getDefaultTabelModel(String judul[], boolean[] edit) {
      final boolean[] editable = edit;
      return new DefaultTableModel(
            new Object[][]{},
```

```
            judul ){
        boolean canEdit[] = editable;

        @Override
        public boolean isCellEditable(int rowIndex, int columnIndex){
            return canEdit[columnIndex];
        }
      };
   }   }
```

Skripsi.java

```
package skripsi;

public class Skripsi {
   public static void main(String[] args) {
   new LoginUser().setVisible(true);
   }
}
```

LoginUser.java

```
package skripsi;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
public class LoginUser extends javax.swing.JFrame {
 private String username;
 private String password;

   public LoginUser() {
      initComponents();
      setLocationRelativeTo(null);
   }
   private boolean validasi(JTextField txtuser, JPasswordField txtpass) {
      if (txtuser.getText().isEmpty()){
         JOptionPane.showMessageDialog(null, "Username Harus Diisi!", "Peringatan",
JOptionPane.WARNING_MESSAGE);
         txtuser.requestFocus();
         return false;
      } else if (txtpass.getText().isEmpty()){
         JOptionPane.showMessageDialog(null, "Password Harus Diisi!", "Peringatan",
JOptionPane.WARNING_MESSAGE);
         txtpass.requestFocus();
         return false;
      } else {
         return true;
```

```
    }   }
 private String login(String username, String password){
      Database database = new Database();
      return database.login(username, password);
   }
private void loginActionPerformed(java.awt.event.ActionEvent evt) {
   if (validasi(txtuser, txtpass)){
         username = txtuser.getText();
         password = txtpass.getText();
         String jabatan = login(username, password);
         if (jabatan.isEmpty()){
            JOptionPane.showMessageDialog(null, "Login Gagal!", "PEringatan",
JOptionPane.WARNING_MESSAGE);
            txtuser.requestFocus();
         } else if (jabatan.equals("lurah")){
         new MenuUtama("lurah").setVisible(true);
            dispose();
         } else if (jabatan.equals("perangkat desa")){
         new MenuUtama("perangkat desa").setVisible(true);
            dispose();
         }
      }}
     public static void main(String args[]) {
      try {
         for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
               javax.swing.UIManager.setLookAndFeel(info.getClassName());
               break;
            }          }
      } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(LoginUser.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
      } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(LoginUser.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
      } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(LoginUser.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
      } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(LoginUser.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
      }
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new LoginUser().setVisible(true);
         }
```

```
        });
    }
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JButton login;
    private javax.swing.JPasswordField txtpass;
    private javax.swing.JTextField txtuser;
}
```

MenuUtama.java

```
package skripsi;
import java.util.HashMap;
import java.util.Map;
import java.sql.Connection;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.view.JasperViewer;
public class MenuUtama extends javax.swing.JFrame {
private static String posisi;
    public MenuUtama(String posisi) {
        initComponents();
        setLocationRelativeTo(null);
       MenuUtama.posisi = posisi;
        if (posisi == "lurah"){
            menuuser.setEnabled(true);
            menufaktor.setEnabled(false);
            menukriteria.setEnabled(false);
            menusubkriteria.setEnabled(false);
            menupenduduk.setEnabled(false);
            menunilai.setEnabled(false);
            menuperhitungan.setEnabled(false);
            menupenerima.setEnabled(true);
            lappenduduk.setEnabled(false);
            lapnilai.setEnabled(false);
            menukritNsubkrit.setEnabled(false);
        } else if (posisi == "perangkat desa"){
            menuuser.setEnabled(false);
            menufaktor.setEnabled(true);
            menukriteria.setEnabled(true);
            menusubkriteria.setEnabled(true);
            menupenduduk.setEnabled(true);
            menunilai.setEnabled(true);
```

```
            menuperhitungan.setEnabled(true);
            menupenerima.setEnabled(true);
            lappenduduk.setEnabled(true);
            lapnilai.setEnabled(true);
            menukritNsubkrit.setEnabled(true);
        }     }
private void menuuserMenuKeyPressed(javax.swing.event.MenuKeyEvent evt) {
        new User().setVisible(true);
}

private void menupendudukActionPerformed(java.awt.event.ActionEvent evt) {
       new Penduduk().setVisible(true);
}
private void menufaktorActionPerformed(java.awt.event.ActionEvent evt) {
        new Faktor().setVisible(true);
}
private void menukriteriaActionPerformed(java.awt.event.ActionEvent evt) {
        new Kriteria().setVisible(true);
}
private void menusubkriteriaActionPerformed(java.awt.event.ActionEvent evt) {
      new SubKriteria().setVisible(true);
}
private void menusubkriteriaMenuKeyPressed(javax.swing.event.MenuKeyEvent evt) {
      new Kriteria().setVisible(true);
}
private void menunilaiActionPerformed(java.awt.event.ActionEvent evt) {
        new Nilai().setVisible(true);
}
private void menuperhitunganActionPerformed(java.awt.event.ActionEvent evt) {
        new Perhitungan().setVisible(true);
}
private void menupenerimaActionPerformed(java.awt.event.ActionEvent evt) {
       new LaporanHasil().setVisible(true);
}
private void lappendudukActionPerformed(java.awt.event.ActionEvent evt) {
       try{
         Connection database = new Database().konek();
        Map prs = new HashMap();
        JasperReport JRpt =
JasperCompileManager.compileReport("src/skripsi/laporan/laporanpenduduk.jrxml");
        JasperPrint JPrint = JasperFillManager.fillReport(JRpt, prs , database);
        JasperViewer.viewReport(JPrint, false);
        }
    catch(Exception rpt){
        System.out.println("tidak dapat menampilkan karena : "+rpt);
           }
}
```

```
private void lapnilaiActionPerformed(java.awt.event.ActionEvent evt) {
      new LaporanNilai().setVisible(true);
}
private void menukritNsubkritActionPerformed(java.awt.event.ActionEvent evt) {
      new LaporanSubKriteria().setVisible(true);
}
   public static void main(String args[]) {
      try {
         for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
               javax.swing.UIManager.setLookAndFeel(info.getClassName());
               break;
            }
         }
      } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(MenuUtama.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
      } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(MenuUtama.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
      } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(MenuUtama.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
      } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(MenuUtama.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
      }
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new MenuUtama(posisi).setVisible(true);
         }
      });
   }
   private javax.swing.JMenu jMenu1;
   private javax.swing.JMenu jMenu2;
   private javax.swing.JMenuBar jMenuBar1;
   private javax.swing.JMenuItem lapnilai;
   private javax.swing.JMenuItem lappenduduk;
   private javax.swing.JMenuItem menufaktor;
   private javax.swing.JMenuItem menukritNsubkrit;
   private javax.swing.JMenuItem menukriteria;
   private javax.swing.JMenuItem menunilai;
   private javax.swing.JMenuItem menupenduduk;
   private javax.swing.JMenuItem menupenerima;
   private javax.swing.JMenuItem menuperhitungan;
   private javax.swing.JMenuItem menusubkriteria;
```

```
   private javax.swing.JMenuItem menuuser;
}
```

Faktor.java

```
package skripsi;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;
public class Faktor extends javax.swing.JFrame {
   private int kodeFaktor;
   private String namaFaktor;
   private int bobotFaktor;
   private boolean ubah;
   private DefaultTableModel tabelModel;
      public Faktor() {
      initComponents();
      showTable(tabel, tabelModel);
   }
private void batal(boolean b){
      txtnamafaktor.setEnabled(b);
      txtbobotfaktor.setEnabled(b);
   }
      private void kosong(){
         txtkodefaktor.setText("");
         txtnamafaktor.setText("");
         txtbobotfaktor.setText("");
      }
      private boolean showTable(JTable tabel, DefaultTableModel tabelModel){
      Database database = new Database();
      try {
         Object[][] data = database.getFaktor();
         Object[] dataTable = new Object[data[0].length+1];
         Table createTable = new Table();
         tabelModel = createTable.getDefaultTabelModel(
               new String[]{"Kode Faktor", "Nama Faktor", "Bobot Faktor"});
         tabel.setModel(tabelModel);
         tabel.getTableHeader().setResizingAllowed(false);
         tabel.getTableHeader().setReorderingAllowed(false);
         for (int i = 0; i < tabel.getColumnCount(); i++) {
            tabel.getColumnModel().getColumn(i).setResizable(false);
         }

         tabelModel.setRowCount(0);
         for (int i = 0; i < data.length; i++) {
            dataTable[0] = data[i][0];
            dataTable[1] = data[i][1];
```

```
            dataTable[2] = data[i][2];
            tabelModel.addRow(dataTable);
          }
          return true;
        } catch (Exception e) {
          return false;
        }   }
    private int getInt(JTextField txtkodefaktor){
        int hasil = 0;
        try {
          hasil = Integer.parseInt(txtkodefaktor.getText());
        } catch (NumberFormatException numberFormatException) {
        }
        return hasil;
    }
    private void ubah(){
      if(tabel.getSelectedRow() >= 0){
        try {
          int baris = tabel.getSelectedRow();
          kodeFaktor = (Integer) tabel.getValueAt(baris, 0);
          namaFaktor = (String) tabel.getValueAt(baris, 1);
          bobotFaktor = (Integer) tabel.getValueAt(baris, 2);
          txtkodefaktor.setText("" + kodeFaktor);
          txtnamafaktor.setText(namaFaktor);
          txtbobotfaktor.setText("" + bobotFaktor);
        } catch (Exception e) {
        }   }
    else{
     JOptionPane.showMessageDialog(null, "Pilih Data Yang akan diedit", "Pemberitahuan",
JOptionPane.INFORMATION_MESSAGE);
}}
    private void hapus(int idUser){
        Database database = new Database();
        if (database.deleteFaktor(idUser)){
          JOptionPane.showMessageDialog(null, "Berhasil Menghapus Data Faktor",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
        } else {
          JOptionPane.showMessageDialog(null, "Gagal Menghapus Data Faktor",
"Peringatan", JOptionPane.WARNING_MESSAGE);
        }   }
    private void simpan(int kodeFaktor, String namaFaktor, int bobotFaktor, boolean ubah){
        Database database = new Database();
        Object[] data = {kodeFaktor, namaFaktor, bobotFaktor};
        if (!ubah){
          if(validasiv()){
          if (database.insertFaktor(data)){
```

```
            JOptionPane.showMessageDialog(null, "Berhasil Menginput Data Faktor",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
          } else {
            JOptionPane.showMessageDialog(null, "Gagal Menginput Data Faktor",
"Peringatan", JOptionPane.WARNING_MESSAGE);
          }          }
       } else {
          if (validasif()){

          if (database.updateFaktor(data)){
            JOptionPane.showMessageDialog(null, "Berhasil Mengupdate Data Faktor",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
          } else {
            JOptionPane.showMessageDialog(null, "Gagal Mengupdate Data Faktor",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
          }
          }       }    }
    private boolean validasi(JTextField txtnamafaktor, JTextField txtbobotfaktor){
       if (txtnamafaktor.getText().equals("")){
          JOptionPane.showMessageDialog(null, "Nama Kriteria Harus Diisi!", "Peringatan!",
JOptionPane.WARNING_MESSAGE);
          return false;
       } else if (txtbobotfaktor.getText().equals("")){
          JOptionPane.showMessageDialog(null, "Persentasi Harus Diisi!", "Peringatan!",
JOptionPane.WARNING_MESSAGE);
          return false;
       } else {
          return true;
       }    }
       private boolean validasif(){
      Database database = new Database();
      int row = tabel.getSelectedRow();
      int databaru = Integer.parseInt(txtbobotfaktor.getText());
      int datalama = (Integer) tabel.getValueAt(row, 2);
      try {
      int hasil = 0;
      Object[][] data = database.getSFaktor();
      for (int i = 0; i < data.length; i++) {
            hasil = (Integer) data[i][0];
            int result = (hasil-datalama)+databaru;
            if (result > 100){
          JOptionPane.showMessageDialog(null, "Bobot faktor sudah mencapai maksimal",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
          return false;
      }      }
      return true;
      }
```

```
      catch (Exception e) {
          return false;
       }         }
   private boolean validasiv(){
      Database database = new Database();
      try {
      int hasil = 0;
      Object[][] data = database.getSFaktor();
      for (int i = 0; i < data.length; i++) {
            hasil = (Integer) data[i][0];
            if (hasil >= 100){
         JOptionPane.showMessageDialog(null, "Bobot faktor sudah mencapai maksimal",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
          return false;
      }
      }
      return true;
      }
      catch (Exception e) {
          return false;
       }  }
private void btnhapusActionPerformed(java.awt.event.ActionEvent evt) {
 try {
          int baris = tabel.getSelectedRow();
          kodeFaktor = (Integer) tabel.getValueAt(baris, 0);
          int jawab = JOptionPane.showOptionDialog(null, "Anda Yakin Akan Menghapus data
ini?", "Pertanyaan", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
null,new String[]{"Ya", "Tidak"},null);
          if (jawab == 0) {
             hapus(kodeFaktor);
             showTable(tabel, tabelModel);
          } else {
          }
       } catch (Exception e) {
          JOptionPane.showMessageDialog(null, "Pilih Data Yang Akan Dihapus Dulu",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
       }
}
private void btnbatalActionPerformed(java.awt.event.ActionEvent evt) {
kosong();
}
private void btntambahActionPerformed(java.awt.event.ActionEvent evt) {
       ubah = false;
       batal(true);
       btnsimpan.setEnabled(true);
       btnbatal.setEnabled(true);
}
```

```
private void btnubahActionPerformed(java.awt.event.ActionEvent evt) {
      ubah = true;
      ubah();
      batal(false);
      batal(true);
      btnsimpan.setEnabled(true);
      btnbatal.setEnabled(true);
}
private void btnsimpanActionPerformed(java.awt.event.ActionEvent evt) {
kodeFaktor = getInt(txtkodefaktor);
       namaFaktor = txtnamafaktor.getText();
       bobotFaktor = getInt(txtbobotfaktor);
       if (validasi(txtnamafaktor, txtbobotfaktor)) {
          simpan(kodeFaktor, namaFaktor, bobotFaktor, ubah);
          btnsimpan.setEnabled(false);
          btnbatal.setEnabled(false);
          batal(false);
          showTable(tabel, tabelModel);
         }kodeFaktor = getInt(txtkodefaktor);
       namaFaktor = txtnamafaktor.getText();
       bobotFaktor = getInt(txtbobotfaktor);
       if (validasi(txtnamafaktor, txtbobotfaktor)) {
          simpan(kodeFaktor, namaFaktor, bobotFaktor, ubah);
          btnsimpan.setEnabled(false);
          btnbatal.setEnabled(false);
          batal(false);
          showTable(tabel, tabelModel);
               }}
   public static void main(String args[]) {
       try {
          for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
             if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
             }          }
       } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(Faktor.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);
       } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Faktor.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);
       } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(Faktor.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);
```

```
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Faktor.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Faktor().setVisible(true);
            }
        });
    }
    private javax.swing.JButton btnbatal;
    private javax.swing.JButton btnhapus;
    private javax.swing.JButton btnsimpan;
    private javax.swing.JButton btntambah;
    private javax.swing.JButton btnubah;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tabel;
    private javax.swing.JTextField txtbobotfaktor;
    private javax.swing.JTextField txtkodefaktor;
    private javax.swing.JTextField txtnamafaktor;
}
```

Kriteria.java

```
package skripsi;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;
public class Kriteria extends javax.swing.JFrame {

    private int kodeKriteria;
    private String namaKriteria;
    private int bobotKriteria;
    private boolean ubah;
    private DefaultTableModel tabelModel;
      public Kriteria() {
       initComponents();
```

```
        showTable(tabel, tabelModel);
    }
    private void batal(boolean b){
        txtnamakrit.setEnabled(b);
        txtbobotkrit.setEnabled(b);
    }
private void kosong(){
            txtkodekrit.setText("");
            txtnamakrit.setText("");
            txtbobotkrit.setText("");
           }
    private boolean showTable(JTable tabel, DefaultTableModel tabelModel){
        Database database = new Database();
        try {
            Object[][] data = database.getKriteria();
            Object[] dataTable = new Object[data[0].length+1];
            Table createTable = new Table();
            tabelModel = createTable.getDefaultTabelModel(
            new String[]{"Kode Kriteria", "Nama Kriteria", "Bobot Kriteria"});
            tabel.setModel(tabelModel);
            tabel.getTableHeader().setResizingAllowed(false);
            tabel.getTableHeader().setReorderingAllowed(false);
            for (int i = 0; i < tabel.getColumnCount(); i++) {
                tabel.getColumnModel().getColumn(i).setResizable(false);
            }
            tabelModel.setRowCount(0);
            for (int i = 0; i < data.length; i++) {
                dataTable[0] = data[i][0];
                dataTable[1] = data[i][1];
                dataTable[2] = data[i][2];
                tabelModel.addRow(dataTable);
            }
            return true;
        } catch (Exception e) {
            return false;
        }   }
    private int getInt(JTextField txtkodekriteria){
        int hasil = 0;
        try {
            hasil = Integer.parseInt(txtkodekriteria.getText());
        } catch (NumberFormatException numberFormatException) {
        }
        return hasil;
    }
    private void ubah(){
         if(tabel.getSelectedRow() >= 0){
        try {
```

```
            int baris = tabel.getSelectedRow();
            kodeKriteria = (Integer) tabel.getValueAt(baris, 0);
            namaKriteria = (String) tabel.getValueAt(baris, 1);
            bobotKriteria = (Integer) tabel.getValueAt(baris, 2);
            txtkodekrit.setText("" + kodeKriteria);
            txtnamakrit.setText(namaKriteria);
            txtbobotkrit.setText("" + bobotKriteria);
        } catch (Exception e) {
        }       }
         else{
     JOptionPane.showMessageDialog(null, "Pilih Data Yang akan diedit", "Pemberitahuan",
JOptionPane.INFORMATION_MESSAGE);
}    }
    private void hapus(int id){
        Database database = new Database();
        if (database.deleteKriteria(id)){
            JOptionPane.showMessageDialog(null, "Berhasil Menghapus Data Kriteria",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null, "Gagal Menghapus Data Kriteria",
"Peringatan", JOptionPane.WARNING_MESSAGE);
        }    }
    private void simpan(int kodeKriteria, String namaKriteria, int bobotKriteria, boolean
ubah){
        Database database = new Database();
        Object[] data = {kodeKriteria, namaKriteria, bobotKriteria};
        if (!ubah){
            if (validasiv()){
            if (database.insertKriteria(data)){
            JOptionPane.showMessageDialog(null, "Berhasil Menginput Data Kriteria",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
            }else {
                JOptionPane.showMessageDialog(null, "Gagal Menginput DataKriteria",
"Peringatan", JOptionPane.WARNING_MESSAGE);
        }           }
        }
        else {
            if (validasif()){
            if (database.updateKriteria(data)){
                JOptionPane.showMessageDialog(null, "Berhasil Mengupdate Data Kriteria",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Gagal Mengupdate Data Kriteria",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
            }
        }    }    }
    private boolean validasi(JTextField txtnamakriteria, JTextField txtbobotkriteria){
```

```
       if (txtnamakriteria.getText().equals("")){
          JOptionPane.showMessageDialog(null, "Nama Kriteria Harus Diisi!", "Peringatan!",
JOptionPane.WARNING_MESSAGE);
          return false;
       } else if (txtbobotkriteria.getText().equals("")){
          JOptionPane.showMessageDialog(null, "Persentasi Harus Diisi!", "Peringatan!",
JOptionPane.WARNING_MESSAGE);
          return false;
       } else {
          return true;
       }   }
 private boolean validasif(){
      Database database = new Database();
      int row = tabel.getSelectedRow();
      int databaru = Integer.parseInt(txtbobotkrit.getText());
      int datalama = (Integer) tabel.getValueAt(row, 2);
      try {
      int hasil = 0;
      Object[][] data = database.getSKriteria();
      for (int i = 0; i < data.length; i++) {
            hasil = (Integer) data[i][0];
             int result = (hasil-datalama)+databaru;
            if  (result > 100){
         JOptionPane.showMessageDialog(null, "Bobot kriteria sudah mencapai maksimal",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
          return false;
      }
                }
      return true;
      }
      catch (Exception e) {
          return false;
       }         }
private boolean validasiv(){
      Database database = new Database();
       try {
      int hasil = 0;
      Object[][] data = database.getSKriteria();
      for (int i = 0; i < data.length; i++) {
            hasil = (Integer) data[i][0];
            if  (hasil >= 100){
         JOptionPane.showMessageDialog(null, "Bobot kriteria sudah mencapai maksimal",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
          return false;
      }
                }
      return true;
```

```
      }
      catch (Exception e) {
          return false;
       }
         }
   private void btnhapusActionPerformed(java.awt.event.ActionEvent evt) {
try {
          int baris = tabel.getSelectedRow();
          kodeKriteria = (Integer) tabel.getValueAt(baris, 0);
          int jawab = JOptionPane.showOptionDialog(null, "Anda Yakin Akan Menghapus data
ini?", "Pertanyaan", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
null,new String[]{"Ya", "Tidak"},null);
          if (jawab == 0) {
          hapus(kodeKriteria);
          showTable(tabel, tabelModel);
          } else {
          }
      } catch (Exception e) {
          JOptionPane.showMessageDialog(null, "Pilih Data Yang Akan Dihapus Dulu",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
      }
}
private void btnbatalActionPerformed(java.awt.event.ActionEvent evt) {
kosong();
}
private void btntambahActionPerformed(java.awt.event.ActionEvent evt) {
       kosong();
       ubah = false;
       batal(true);
       btnsimpan.setEnabled(true);
       btnbatal.setEnabled(true);
}
private void btnubahActionPerformed(java.awt.event.ActionEvent evt) {
      ubah = true;
      ubah();
      batal(false);
      batal(true);
      btnsimpan.setEnabled(true);
      btnbatal.setEnabled(true);
}

private void btnsimpanActionPerformed(java.awt.event.ActionEvent evt) {
   kodeKriteria = getInt(txtkodekrit);
       namaKriteria = txtnamakrit.getText();
       bobotKriteria = getInt(txtbobotkrit);
       if (validasi(txtnamakrit, txtbobotkrit)) {
          simpan(kodeKriteria, namaKriteria, bobotKriteria, ubah);
```

```
            btnsimpan.setEnabled(false);
            btnbatal.setEnabled(false);
            batal(false);
            showTable(tabel, tabelModel);
        } }
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(Kriteria.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(Kriteria.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(Kriteria.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Kriteria.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new Kriteria().setVisible(true);
            }
        });
    }
    private javax.swing.JButton btnbatal;
    private javax.swing.JButton btnhapus;
    private javax.swing.JButton btnsimpan;
    private javax.swing.JButton btntambah;
    private javax.swing.JButton btnubah;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
```

```
    private javax.swing.JPanel jPanel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tabel;
    private javax.swing.JTextField txtbobotkrit;
    private javax.swing.JTextField txtkodekrit;
    private javax.swing.JTextField txtnamakrit;
}
```

Subkriteria.java

```
package skripsi;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;

public class SubKriteria extends javax.swing.JFrame {

    public static int kodeKriteria[];
    private String namaKriteria;
    private int kodeSubKriteria;
    private String namaSubKriteria;
    private int NilaiProfil;
    public static int kodeFaktor[];
    private String namaFaktor;
    private boolean ubah;
    private DefaultTableModel tabelModel;
        public SubKriteria() {
        initComponents();
        kodeKriteria = createComboKriteria(cmbkriteria);
        kodeFaktor = createComboFaktor(cmbfaktor);
        showTable(tabel, tabelModel);
    }
 public static int[] createComboKriteria(JComboBox cmbKriteria){
        int[] hasil = null;
        Database database = new Database();
        try {
            Object[][] data = database.getKriteria();
            hasil = new int[data.length];
            cmbKriteria.removeAllItems();

            for (int i = 0; i < data.length; i++) {
                hasil[i] = (Integer) data[i][0];
                cmbKriteria.addItem(data[i][1]);
            }
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        return hasil;
    }
    public static int[] createComboFaktor(JComboBox cmbFaktor){
        int[] hasil = null;
        Database database = new Database();
        try {
            Object[][] data = database.getFaktor();
            hasil = new int[data.length];
            cmbFaktor.removeAllItems();

            for (int i = 0; i < data.length; i++) {
                hasil[i] = (Integer) data[i][0];
                cmbFaktor.addItem(data[i][1]);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return hasil;
    }
    private boolean showTable(JTable tabel, DefaultTableModel tabelModel){
        Database database = new Database();
        try {
            Object[][] data = database.getSubKriteria();
            Object[] dataTable = new Object[data[0].length+1];
            Table createTable = new Table();
            tabelModel = createTable.getDefaultTabelModel(
            new String[]{"Kode Sub Kriteria", "Nama Sub Kriteria", "Nilai Profil", "Nama Kriteria",
"Faktor",});
            tabel.setModel(tabelModel);
            tabel.getTableHeader().setResizingAllowed(false);
            tabel.getTableHeader().setReorderingAllowed(false);

            for (int i = 0; i < tabel.getColumnCount(); i++) {
                tabel.getColumnModel().getColumn(i).setResizable(false);
            }

            tabelModel.setRowCount(0);
            for (int i = 0; i < data.length; i++) {
                dataTable[0] = data[i][0];
                dataTable[1] = data[i][1];
                dataTable[2] = data[i][2];
                dataTable[3] = data[i][7];
                dataTable[4] = data[i][4];
                tabelModel.addRow(dataTable);
            }
```

```
            return true;
        } catch (Exception e) {
            return false;
        }    }
    private int getInt(JTextField a){
        int hasil = 0;
        try {
            hasil = Integer.parseInt(a.getText());
        } catch (NumberFormatException numberFormatException) {
        }
        return hasil;
    }
   private void ubah(){
       if(tabel.getSelectedRow() >=0 ){
        int baris = tabel.getSelectedRow();
        namaKriteria = (String) tabel.getValueAt(baris, 3);
        kodeSubKriteria = (Integer) tabel.getValueAt(baris,0);
        namaSubKriteria = (String) tabel.getValueAt(baris,1);
        NilaiProfil = (Integer) tabel.getValueAt(baris, 2);
        namaFaktor= (String) tabel.getValueAt(baris,4);

        txtkodesubkrit.setText(""+kodeSubKriteria);
        txtnamasubkrit.setText(namaSubKriteria);
        txtnilaiprofil.setText(""+NilaiProfil);
        cmbkriteria.setSelectedItem(namaKriteria);
        cmbfaktor.setSelectedItem(namaFaktor);
    }
       else{
     JOptionPane.showMessageDialog(null, "Pilih Data Yang akan diedit", "Pemberitahuan",
JOptionPane.INFORMATION_MESSAGE);
        }
}

    private void hapus(int id){
        Database database = new Database();
        if (database.deleteSubKriteria(id)){
            JOptionPane.showMessageDialog(null, "Berhasil Menghapus Data Sub Kriteria",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null, "Gagal Menghapus Data Sub Kriteria",
"Peringatan", JOptionPane.WARNING_MESSAGE);
        }    }
 private void simpan(int kodeSubKriteria, String namaSubKriteria, int NilaiProfil, int
kodeKriteria, int kodeFaktor, boolean ubah){
        Database database = new Database();
        Object[] data = {kodeSubKriteria, namaSubKriteria, NilaiProfil, kodeKriteria,
kodeFaktor};
```

```
        if (!ubah){
            if (database.insertSubKriteria(data)){
                JOptionPane.showMessageDialog(null, "Berhasil Menginput Data Sub Kriteria",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Gagal Menginput Data Sub Kriteria",
"Peringatan", JOptionPane.WARNING_MESSAGE);
            }
        } else {
            if (database.updateSubKriteria(data)){
                JOptionPane.showMessageDialog(null, "Berhasil Mengupdate Data Sub Kriteria",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Gagal Mengupdate Data Sub Kriteria",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
            }
        }   }
    private void batal(boolean b){

        txtkodesubkrit.setEnabled(b);
        txtnamasubkrit.setEnabled(b);
        cmbfaktor.setEnabled(b);
        txtnilaiprofil.setEnabled(b);
        cmbkriteria.setEnabled(b);
    }
        private void kosong(){
       txtkodesubkrit.setText("");
       txtnamasubkrit.setText("");
       txtnilaiprofil.setText("");
       cmbfaktor.setSelectedIndex(0);
       cmbkriteria.setSelectedIndex(0);
          }

    private boolean validasi(JTextField namasubkriteria, JTextField nilaiprofil){
        if (namasubkriteria.getText().equals("")){
            JOptionPane.showMessageDialog(null, "Nama Sub Kriteria Harus Diisi!",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
            return false;
        } else if (nilaiprofil.getText().equals("")){
            JOptionPane.showMessageDialog(null, "Bobot Harus Diisi!", "Peringatan!",
JOptionPane.WARNING_MESSAGE);
            return false;
        } else {
            return true;
        }   }
private void btnhapusActionPerformed(java.awt.event.ActionEvent evt) {
 try {
```

```
            int baris = tabel.getSelectedRow();
            kodeSubKriteria = (Integer) tabel.getValueAt(baris, 0);
            int jawab = JOptionPane.showOptionDialog(null, "Anda Yakin Akan Menghapus data
ini?", "Pertanyaan", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
null,new String[]{"Ya", "Tidak"},null);
            if (jawab == 0) {
               hapus(kodeSubKriteria);
               showTable(tabel, tabelModel);
            } else {
            }
         } catch (Exception e) {
            JOptionPane.showMessageDialog(null, "Pilih Data Yang Akan Dihapus Dulu",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
         }
}
private void btnbatalActionPerformed(java.awt.event.ActionEvent evt) {
kosong();
}
private void btntambahActionPerformed(java.awt.event.ActionEvent evt) {
 ubah = false;
         batal(true);
         btnsimpan.setEnabled(true);
         btnbatal.setEnabled(true);
}
private void btnubahActionPerformed(java.awt.event.ActionEvent evt) {
         ubah = true;
         ubah();
         batal(true);
         btnsimpan.setEnabled(true);
         btnbatal.setEnabled(true);
}
private void btnsimpanActionPerformed(java.awt.event.ActionEvent evt) {
         kodeSubKriteria = getInt(txtkodesubkrit);
         namaSubKriteria = txtnamasubkrit.getText();
         NilaiProfil = getInt(txtnilaiprofil);
         int kodeKriteria = this.kodeKriteria[cmbkriteria.getSelectedIndex()];
         int kodeFactor = this.kodeFaktor[cmbfaktor.getSelectedIndex()];
         if (validasi(txtnamasubkrit, txtnilaiprofil)) {
            simpan(kodeSubKriteria, namaSubKriteria, NilaiProfil, kodeKriteria, kodeFactor,
ubah);
            btnsimpan.setEnabled(false);
            btnbatal.setEnabled(false);
            batal(false);
            showTable(tabel, tabelModel);
         }}
 public static void main(String args[]) {
         try {
```

```
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(SubKriteria.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(SubKriteria.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(SubKriteria.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(SubKriteria.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
    }
    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {
            new SubKriteria().setVisible(true);
        }
    });
  }
  private javax.swing.JButton btnbatal;
  private javax.swing.JButton btnhapus;
  private javax.swing.JButton btnsimpan;
  private javax.swing.JButton btntambah;
  private javax.swing.JButton btnubah;
  private javax.swing.JComboBox cmbfaktor;
  private javax.swing.JComboBox cmbkriteria;
  private javax.swing.JLabel jLabel1;
  private javax.swing.JLabel jLabel2;
  private javax.swing.JLabel jLabel3;
  private javax.swing.JLabel jLabel4;
  private javax.swing.JLabel jLabel5;
  private javax.swing.JLabel jLabel6;
  private javax.swing.JPanel jPanel1;
  private javax.swing.JPanel jPanel2;
  private javax.swing.JPanel jPanel3;
```

```
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tabel;
    private javax.swing.JTextField txtkodesubkrit;
    private javax.swing.JTextField txtnamasubkrit;
    private javax.swing.JTextField txtnilaiprofil;
}
```

Penduduk.java

```
package skripsi;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;

public class Penduduk extends javax.swing.JFrame {

private int Id;
private String noKk;
private String namaKk;
private int RT;
private int RW;
private boolean ubah;
private DefaultTableModel tabelModel;
    public Penduduk() {
        initComponents();
         showTable(tabel, tabelModel);
    }
private boolean showTable(JTable tabel, DefaultTableModel tabelModel){
        Database database = new Database();
        try {
            Object[][] data = database.getPenduduk();
            Object[] dataTable = new Object[data[0].length+1];
            Table createTable = new Table();
            tabelModel = createTable.getDefaultTabelModel(
                  new String[]{"ID Penduduk", "Nomor KK", "Nama KK", "RT", "RW"});
            tabel.setModel(tabelModel);
            tabel.getTableHeader().setResizingAllowed(false);
            tabel.getTableHeader().setReorderingAllowed(false);

            for (int i = 0; i < tabel.getColumnCount(); i++) {
               tabel.getColumnModel().getColumn(i).setResizable(false);
            }

            tabelModel.setRowCount(0);
```

```
            for (int i = 0; i < data.length; i++) {
                dataTable[0] = data[i][0];
                dataTable[1] = data[i][1];
                dataTable[2] = data[i][2];
                dataTable[3] = data[i][3];
                dataTable[4] = data[i][4];
                tabelModel.addRow(dataTable);
            }
            return true;
        } catch (Exception e) {
            return false;
        }
    }
    private int getID(JTextField textPenduduk){
        int hasil = 0;
        try {
            hasil = Integer.parseInt(textPenduduk.getText());
        } catch (NumberFormatException numberFormatException) {
        }
        return hasil;
    }

    private void ubah(){
        if(tabel.getSelectedRow() < 0){
            JOptionPane.showMessageDialog(null, "Pilih Data Yang akan diedit",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
        }else{
        int baris = tabel.getSelectedRow();
        Id = (Integer) tabel.getValueAt(baris, 0);
        noKk = (String) tabel.getValueAt(baris, 1);
        namaKk = (String) tabel.getValueAt(baris, 2);
        RT= (Integer) tabel.getValueAt(baris, 3);
        RW = (Integer) tabel.getValueAt(baris, 4);
        txtid.setText(""+Id);
        txtnokk.setText(noKk);
        txtnamakk.setText(namaKk);
        txtrt.setText(""+RT);
        txtrw.setText(""+RW);
        }       }
    private void batal(boolean b){
        txtnokk.setEnabled(b);
        txtnamakk.setEnabled(b);
        txtrt.setEnabled(b);
        txtrw.setEnabled(b);
    }
    private void kosong(){
        txtid.setText("");
```

```
        txtnokk.setText("");
        txtnamakk.setText("");
        txtrt.setText("");
        txtrw.setText("");
    }
   private void hapus(int IdPenduduk){
        Database database = new Database();
        if (database.deletePenduduk(IdPenduduk)){
            JOptionPane.showMessageDialog(null, "Berhasil Menghapus Data Penduduk",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null, "Gagal Menghapus Data Penduduk",
"Peringatan", JOptionPane.WARNING_MESSAGE);
        }
    }

    private void simpan(int Id, String noKk, String namaKk, int RT, int RW, boolean ubah){
        Database database = new Database();
        Object[] data = {Id,noKk,namaKk,RT,RW};
        if (!ubah){
            if (database.insertPenduduk(data)){
                JOptionPane.showMessageDialog(null, "Berhasil Menginput Data Penduduk",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Gagal Menginput Data Penduduk",
"Peringatan", JOptionPane.WARNING_MESSAGE);
            }
        } else {
            if (database.updatePenduduk(data)){
                JOptionPane.showMessageDialog(null, "Berhasil Mengupdate Data Penduduk",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Gagal Mengupdate Data Penduduk",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
            }       }    }
 private boolean validasi(JTextField txtnokk, JTextField txtnamakk, JTextField txtrt,JTextField
txtrw){
        if (txtnokk.getText().equals("")){
            JOptionPane.showMessageDialog(null, "Nama Harus Diisi!", "Peringatan!",
JOptionPane.WARNING_MESSAGE);
            return false;
        } else if (txtnamakk.getText().equals("")){
            JOptionPane.showMessageDialog(null, "Alamat Harus Diisi!", "Peringatan!",
JOptionPane.WARNING_MESSAGE);
            return false;
        } else {
            try {
```

```
            Float.parseFloat(txtrt.getText());
            Float.parseFloat(txtrw.getText());
            return true;
        } catch (NumberFormatException numberFormatException) {
            numberFormatException.printStackTrace();
            JOptionPane.showMessageDialog(null, "Nomer RT dan RW Harus Berupa Angka!",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
            return false;
        }       }    }
private void btnhapusActionPerformed(java.awt.event.ActionEvent evt) {
try {
        int baris = tabel.getSelectedRow();
        Id = (Integer) tabel.getValueAt(baris, 0);
        int jawab = JOptionPane.showOptionDialog(null, "Anda Yakin Akan Menghapus data
ini?", "Pertanyaan", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
null,new String[]{"Ya", "Tidak"},null);
        if (jawab == 0) {
            hapus(Id);
            showTable(tabel, tabelModel);
        } else {
        }
      } catch (Exception e) {
          JOptionPane.showMessageDialog(null, "Pilih Data Yang Akan Dihapus Dulu",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
      }
}

private void btnbatalActionPerformed(java.awt.event.ActionEvent evt) {
kosong();
}
private void btntambahActionPerformed(java.awt.event.ActionEvent evt) {
      ubah = false;
      batal(false);
      batal(true);
      kosong();
      btnsimpan.setEnabled(true);
      btnbatal.setEnabled(true);
}
private void btnubahActionPerformed(java.awt.event.ActionEvent evt) {
      ubah = true;
      ubah();
      batal(true);
      btnsimpan.setEnabled(true);
      btnbatal.setEnabled(true);
}
private void btnsimpanActionPerformed(java.awt.event.ActionEvent evt) {
      Id = getID(txtid);
```

```
        noKk = txtnokk.getText();
        namaKk = txtnamakk.getText();
        RT = getID(txtrt);
        RW = getID(txtrw);
         if (validasi(txtnokk, txtnamakk, txtrt,txtrw)) {
        simpan(Id, noKk, namaKk, RT, RW,ubah);
        btnsimpan.setEnabled(false);
        kosong();
        btnbatal.setEnabled(false);
        batal(false);
        showTable(tabel, tabelModel);
        }
}
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }          }
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(Penduduk.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(Penduduk.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(Penduduk.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Penduduk.class.getName()).log(java.util.logging.Level.SEV
ERE, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new Penduduk().setVisible(true);
            }
        });
    }
    private javax.swing.JButton btnbatal;
    private javax.swing.JButton btnhapus;
    private javax.swing.JButton btnsimpan;
    private javax.swing.JButton btntambah;
    private javax.swing.JButton btnubah;
    private javax.swing.JLabel jLabel1;
```

```
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tabel;
    private javax.swing.JTextField txtid;
    private javax.swing.JTextField txtnamakk;
    private javax.swing.JTextField txtnokk;
    private javax.swing.JTextField txtrt;
    private javax.swing.JTextField txtrw;
    }
```

Nilai.java

```
package skripsi;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;

public class Nilai extends javax.swing.JFrame {

   private int kodeNilai[];
    public static int idPenduduk[];
    public static int kodeKriteria[];
    private int kodeSubKriteria[];
    private String noKk;
    private String namaKk;
    private int RT;
    private int RW;
    private boolean ubah;
    private DefaultTableModel tabelModel;

    public Nilai() {
        initComponents();
        idPenduduk = createComboPenduduk(cmbid);
        kodeKriteria = createComboKriteria(cmbkriteria);
        try {
            int id = idPenduduk[cmbid.getSelectedIndex()];
```

```
            showData(id,txtnokk,txtnamakk,txtrt,txtrw);
        } catch (Exception e) {
        }
        try {
            int kodeKriteria = this.kodeKriteria[cmbkriteria.getSelectedIndex()];
            kodeSubKriteria = showTable(tabel, tabelModel, kodeKriteria);
        } catch (Exception e) {
        }
        try {
            boolean setTabelNilai = setTabelNilai(cmbid, cmbkriteria, tabel);
            ubah = !setTabelNilai;
        } catch (Exception e) {
        }
    }
public static int[] createComboPenduduk(JComboBox cmbid){
        int[] hasil = null;
        Database database = new Database();
        try {
            Object[][] data = database.getPenduduka();
            hasil = new int[data.length];
            cmbid.removeAllItems();
            for (int i = 0; i < data.length; i++) {
                hasil[i] = (Integer) data[i][0];
                cmbid.addItem(data[i][1] +"  ||   "+ data[i][2]);
            }
        } catch (Exception e) {
        }
        return hasil;
    }
    public static int[] createComboKriteria(JComboBox cmbkriteria){
        int[] hasil = null;
        Database database = new Database();
        try {
            Object[][] data = database.getKriteria();
            hasil = new int[data.length];
            cmbkriteria.removeAllItems();
            for (int i = 0; i < data.length; i++) {
                hasil[i] = (Integer) data[i][0];
                cmbkriteria.addItem(data[i][1]);
            }
        } catch (Exception e) {
        }
        return hasil;
    }
    private int[] showTable(JTable tabel, DefaultTableModel tabelModel, int kodeKriteria){
        Database database = new Database();
        int[] hasil = null;
```

```
        try {
            Object[][] data = database.getSubKriteria(kodeKriteria);
            Object[] dataTable = new String[data[0].length+1];
            hasil = new int[data.length];
            Table createTable = new Table();
            tabelModel = createTable.getDefaultTabelModel(
                  new String[]{" Nama Sub Kriteria", "Nilai"}, new boolean[]{false, true});
            tabel.setModel(tabelModel);
            tabel.getTableHeader().setResizingAllowed(false);
            tabel.getTableHeader().setReorderingAllowed(false);
            for (int i = 0; i < tabel.getColumnCount(); i++) {
               tabel.getColumnModel().getColumn(i).setResizable(false);
            }

            tabelModel.setRowCount(0);
            for (int i = 0; i < data.length; i++) {
               hasil[i] = (Integer) data[i][0];
               dataTable[0] = data[i][1];
               dataTable[1] = "";
               tabelModel.addRow(dataTable);
            }
        } catch (Exception e) {
        }
        return hasil;
    }

    private void showData(int id, JTextField txtnokk,JTextField txtnamakk,JTextField
txtrt,JTextField txtrw){
        Database database = new Database();
        try {
            Object[][] data = database.getPenduduk(id);
            txtnokk.setText(""+data[0][1]);
            txtnamakk.setText(""+data[0][2]);
            txtrt.setText(""+data[0][3]);
            txtrw.setText(""+data[0][4]);

             } catch (Exception e) {
        }
    }
    private boolean setTabelNilai(JComboBox cmbid, JComboBox cmbkriteria, JTable tabel){
        boolean hasil = false;
        Database database = new Database();
        String nokk = cmbid.getSelectedItem().toString();
        String namaKriteria = cmbkriteria.getSelectedItem().toString();
        kodeNilai = new int[tabel.getRowCount()];
        for (int i = 0; i < tabel.getRowCount(); i++) {
            String namaSubKriteria = (String) tabel.getValueAt(i, 0);
```

```
            int nilai[] = database.getNilai(namaKriteria, namaSubKriteria, nokk);
            tabel.setValueAt(nilai[3], i, 1);
            kodeNilai[i] = nilai[0];
            if (nilai[0] > 0) {
               hasil = true;
            } else {
            }
        }
        return hasil;
    }
    private boolean simpan(int kodeNilai, int nilai, int kodeSubKriteria, int idPenduduk,
boolean ubah){
        Database database = new Database();
        Object[] data = {kodeNilai,idPenduduk, kodeSubKriteria, nilai };
        if (database.insertNilai(data)) {
              //JOptionPane.showMessageDialog(null, "Berhasil Menambah Data Nilai!",
"Informasi!", JOptionPane.INFORMATION_MESSAGE);
            return true;
        } else {
            return false;
        }
    }
private void cmbidItemStateChanged(java.awt.event.ItemEvent evt) {
    try {
            int id = idPenduduk[cmbid.getSelectedIndex()];
            showData(id, txtnokk, txtnamakk, txtrt, txtrw);
        } catch (Exception e) {
        }
        try {
            int kodeKriteria = this.kodeKriteria[cmbkriteria.getSelectedIndex()];
            kodeSubKriteria = showTable(tabel, tabelModel, kodeKriteria);
        } catch (Exception e) {
        }
        try {
            boolean setTabelNilai = setTabelNilai(cmbid, cmbkriteria, tabel);
            ubah = !setTabelNilai;
        } catch (Exception e) {
        }
}
private void cmbkriteriaItemStateChanged(java.awt.event.ItemEvent evt) {
 try {
            int kodeKriteria = this.kodeKriteria[cmbkriteria.getSelectedIndex()];
            kodeSubKriteria = showTable(tabel, tabelModel, kodeKriteria);
        } catch (Exception e) {
        }
        try {
            boolean setTabelNilai = setTabelNilai(cmbid, cmbkriteria, tabel);
```

```
            ubah = !setTabelNilai;
        } catch (Exception e) {
        }
}
private void btnsimpanActionPerformed(java.awt.event.ActionEvent evt) {
 try {
            boolean stopCellEditing = tabel.getCellEditor().stopCellEditing();
        } catch (Exception e) {
        }
        int nilai = 0;
        int kodeSubKriteria = 0;
        int idKriteria = kodeKriteria[cmbkriteria.getSelectedIndex()];
        int idPenduduk = this.idPenduduk[cmbid.getSelectedIndex()];
        Database database = new Database();
        database.deleteNilai(idKriteria, idPenduduk);
        int[][] data = new int[tabel.getRowCount()][4];
        int benar = 0;
        for (int i = 0; i < tabel.getRowCount(); i++) {
            System.out.println(""+i);
            nilai = Integer.parseInt(tabel.getValueAt(i, 1).toString());
            kodeSubKriteria = this.kodeSubKriteria[i];
            if (simpan(kodeNilai[i], nilai, kodeSubKriteria, idPenduduk, ubah)) {
                benar++;
            }
        }
        if (tabel.getRowCount() == benar){
            JOptionPane.showMessageDialog(null, "Nilai Berhasil Dimasukkan!", "Informasi",
JOptionPane.INFORMATION_MESSAGE);
            }
}
private void btnubahActionPerformed(java.awt.event.ActionEvent evt) {
new NilaiEdit().setVisible(true);
}
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }           }
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(Nilai.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(Nilai.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
```

```
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(Nilai.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Nilai.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new Nilai().setVisible(true);
            }
        });
    }
    private javax.swing.JButton btnbatal;
    private javax.swing.JButton btnsimpan;
    private javax.swing.JButton btnubah;
    private javax.swing.JComboBox cmbid;
    private javax.swing.JComboBox cmbkriteria;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tabel;
    private javax.swing.JTextField txtnamakk;
    private javax.swing.JTextField txtnokk;
    private javax.swing.JTextField txtrt;
    private javax.swing.JTextField txtrw;
}
```

nilaiEdit.java

```
package skripsi;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;
```

```
public class NilaiEdit extends javax.swing.JFrame {
   private int kodeNilai[];
   public static int idPenduduk[];
   public static int kodeKriteria[];
   private int kodeSubKriteria[];
   private String noKk;
   private String namaKk;
   private int RT;
   private int RW;
   private boolean ubah;
   private DefaultTableModel tabelModel;

   public NilaiEdit() {
      initComponents();
       idPenduduk = createComboPenduduk(cmbid,ubah);
      kodeKriteria = createComboKriteria(cmbkriteria);
      try {
         int id = idPenduduk[cmbid.getSelectedIndex()];
         showData(id,txtnokk,txtnamakk,txtrt,txtrw);
      } catch (Exception e) {
      }
      try {
         int kodeKriteria = this.kodeKriteria[cmbkriteria.getSelectedIndex()];
         kodeSubKriteria = showTable(tabel, tabelModel, kodeKriteria);
      } catch (Exception e) {
      }
      try {
         boolean setTabelNilai = setTabelNilai(cmbid, cmbkriteria, tabel);
         ubah = !setTabelNilai;
      } catch (Exception e) {
      }
   }
 public static int[] createComboPenduduk(JComboBox cmbid,boolean ubah){
      int[] hasil = null;
      Database database = new Database();
      try {
         Object[][] data = database.getPendudukb();
         hasil = new int[data.length];
         cmbid.removeAllItems();
         for (int i = 0; i < data.length; i++) {
            hasil[i] = (Integer) data[i][0];
            cmbid.addItem(data[i][1] +"-" +data[i][2]) ;
       }

      }catch (Exception e) {
      }
```

```
            return hasil;
        }
    public static int[] createComboKriteria(JComboBox cmbkriteria){
        int[] hasil = null;
        Database database = new Database();
        try {
            Object[][] data = database.getKriteria();
            hasil = new int[data.length];
            cmbkriteria.removeAllItems();
            for (int i = 0; i < data.length; i++) {
                hasil[i] = (Integer) data[i][0];
                cmbkriteria.addItem(data[i][1]);
            }
        } catch (Exception e) {
        }
        return hasil;
    }
        private int[] showTable(JTable tabel, DefaultTableModel tabelModel, int kodeKriteria){
        Database database = new Database();
        int[] hasil = null;
        try {
            Object[][] data = database.getSubKriteria(kodeKriteria);
            Object[] dataTable = new String[data[0].length+1];
            hasil = new int[data.length];
            Table createTable = new Table();
            tabelModel = createTable.getDefaultTabelModel(
                  new String[]{" Nama Sub Kriteria", "Nilai"}, new boolean[]{false, true});
            tabel.setModel(tabelModel);
            tabel.getTableHeader().setResizingAllowed(false);
            tabel.getTableHeader().setReorderingAllowed(false);
            for (int i = 0; i < tabel.getColumnCount(); i++) {
                tabel.getColumnModel().getColumn(i).setResizable(false);
            }
            tabelModel.setRowCount(0);
            for (int i = 0; i < data.length; i++) {
                hasil[i] = (Integer) data[i][0];
                dataTable[0] = data[i][1];
                dataTable[1] = "";
                tabelModel.addRow(dataTable);
            }
        } catch (Exception e) {
        }
        return hasil;
    }
private void showData(int id, JTextField txtnokk,JTextField txtnamakk,JTextField
txtrt,JTextField txtrw){
        Database database = new Database();
```

```
        try {
            Object[][] data = database.getPenduduk(id);
            txtnokk.setText(""+data[0][1]);
            txtnamakk.setText(""+data[0][2]);
            txtrt.setText(""+data[0][3]);
            txtrw.setText(""+data[0][4]);

             } catch (Exception e) {
        }   }

    private boolean setTabelNilai(JComboBox cmbid, JComboBox cmbkriteria, JTable tabel){
        boolean hasil = false;
        Database database = new Database();
        int nokk = this.idPenduduk[cmbid.getSelectedIndex()];
        String namaKriteria = cmbkriteria.getSelectedItem().toString();
        kodeNilai = new int[tabel.getRowCount()];
        for (int i = 0; i < tabel.getRowCount(); i++) {
            String namaSubKriteria = (String) tabel.getValueAt(i, 0);
            int nilai[] = database.getNilaiN(namaKriteria, namaSubKriteria, nokk);
            tabel.setValueAt(nilai[3], i, 1);
            kodeNilai[i] = nilai[0];
            if (nilai[0] > 0) {
                hasil = true;
            } else {
            }       }

        return hasil;
    }

    private boolean simpan(int kodeNilai, int nilai, int kodeSubKriteria, int idPenduduk,
boolean ubah){
        Database database = new Database();
        Object[] data = {kodeNilai,idPenduduk, kodeSubKriteria, nilai };
        if (database.insertNilai(data)) {
          JOptionPane.showMessageDialog(null, "Berhasil Update Data Nilai!", "Informasi!",
JOptionPane.INFORMATION_MESSAGE);
            return true;
        } else {
            return false;
        }
    }
    private void batal(JTable tabel){
    }
private void btnsimpanActionPerformed(java.awt.event.ActionEvent evt) {
try {
            boolean stopCellEditing = tabel.getCellEditor().stopCellEditing();
        } catch (Exception e) {
```

```
        }
        int nilai = 0;
        int kodeSubKriteria = 0;
        int idKriteria = kodeKriteria[cmbkriteria.getSelectedIndex()];
        int idPenduduk = this.idPenduduk[cmbid.getSelectedIndex()];
        Database database = new Database();
        database.deleteNilai(idKriteria, idPenduduk);
        int[][] data = new int[tabel.getRowCount()][4];
        int benar = 0;
        for (int i = 0; i < tabel.getRowCount(); i++) {
            System.out.println(""+i);
            nilai = Integer.parseInt(tabel.getValueAt(i, 1).toString());
            kodeSubKriteria = this.kodeSubKriteria[i];
            if (simpan(kodeNilai[i], nilai, kodeSubKriteria, idPenduduk, ubah)) {
                benar++;
            }
        }
        if (tabel.getRowCount() == benar){
            JOptionPane.showMessageDialog(null, "Nilai Berhasil Dimasukkan!", "Informasi",
JOptionPane.INFORMATION_MESSAGE);
            }
}
private void cmbidItemStateChanged(java.awt.event.ItemEvent evt) {
 try {
            int id = idPenduduk[cmbid.getSelectedIndex()];
            showData(id, txtnokk, txtnamakk, txtrt, txtrw);
        } catch (Exception e) {
        }
        try {
            int kodeKriteria = this.kodeKriteria[cmbkriteria.getSelectedIndex()];
            kodeSubKriteria = showTable(tabel, tabelModel, kodeKriteria);
        } catch (Exception e) {
        }
        try {
            boolean setTabelNilai = setTabelNilai(cmbid, cmbkriteria, tabel);
            ubah = !setTabelNilai;
        } catch (Exception e) {
        }
}
private void cmbkriteriaItemStateChanged(java.awt.event.ItemEvent evt) {
try {
            int kodeKriteria = this.kodeKriteria[cmbkriteria.getSelectedIndex()];
            kodeSubKriteria = showTable(tabel, tabelModel, kodeKriteria);
        } catch (Exception e) {
        }
        try {
            boolean setTabelNilai = setTabelNilai(cmbid, cmbkriteria, tabel);
```

```
            ubah = !setTabelNilai;
        } catch (Exception e) {
        }
}
 public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(NilaiEdit.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(NilaiEdit.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(NilaiEdit.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(NilaiEdit.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        }
            java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new NilaiEdit().setVisible(true);
            }
        });
    }
    private javax.swing.JButton btnbatal;
    private javax.swing.JButton btnsimpan;
    private javax.swing.JComboBox cmbid;
    private javax.swing.JComboBox cmbkriteria;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
```

```
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tabel;
    private javax.swing.JTextField txtnamakk;
    private javax.swing.JTextField txtnokk;
    private javax.swing.JTextField txtrt;
    private javax.swing.JTextField txtrw;
    }
```

Perhitungan.java

```
package skripsi;
import java.text.DecimalFormat;
import javax.swing.JComboBox;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
public class Perhitungan extends javax.swing.JFrame {
    private static int kodePenduduk;
    private static double[] kriteria;
    private static double nilaiTotal;
    private static String header[];
    private static Object data[];
    private DefaultTableModel tabelModel;
    protected static int[] idPenduduk;
    DecimalFormat decimalFormat;

    public Perhitungan() {
        initComponents();
        decimalFormat = new DecimalFormat("#.###");
        idPenduduk = createComboPenduduk(cmbnamakk);
    }
protected static int[] createComboPenduduk(JComboBox cmbnamakk){
        int[] hasil = null;
        Database database = new Database();
        try {
            Object[][] data = database.getPendudukb();
            hasil = new int[data.length];
            cmbnamakk.removeAllItems();

            for (int i = 0; i < data.length; i++) {
                hasil[i] = (Integer) data[i][0];
                cmbnamakk.addItem(data[i][1]+"-"+data[i][2] );
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
        return hasil;
    }
   private int[][] getNilai(int kodePenduduk){
        Database database = new Database();
        Object[][] nilai = database.getNilaiPenduduk(kodePenduduk);
        int[][] hasil = new int[nilai.length][3];
        int indeks = 0;
        System.out.println("kode_nilai | nilai | kode_sub | id | nilai_profil | kode_faktor |
kode_kriteria ");
        for (int i = 0; i < nilai.length; i++) {
            for (int j = 0; j < nilai[i].length; j++) {
                System.out.print(nilai[i][j]+" \t");
                hasil[indeks][0] = (Integer) nilai[i][1];
                hasil[indeks][1] = (Integer) nilai[i][4];
                hasil[indeks][2] = hasil[indeks][0] - hasil[indeks][1];
            }
            indeks++;
            System.out.println("");
        }
        System.out.println("---------------------------------------");
        return hasil;
    }
    private double[] getFaktor(int kodePenduduk){
        Database database = new Database();
        Object[][] kriteria = database.getKriteria();
        Object[][] faktor = database.getFaktor();
        double totalFaktor[][] = new double[kriteria.length+1][faktor.length];
        double totalKriteria[] = new double[kriteria.length+1];
        int bil = 0;
        System.out.println("---------------------------------------");
        for (int i = 0; i < kriteria.length; i++) {
              for (int j = 0; j < faktor.length; j++) {
                 int kodeKriteria = (Integer) kriteria[i][0];
                 int kodeFaktor = (Integer) faktor[j][0];
                 Object[][] hasil = database.getNilaiByVar(kodeKriteria, kodePenduduk,
kodeFaktor);
                 double b = 0;
                 int pjg = 0;
                 for (int k = 0; k < hasil.length; k++) {
                    double a = ((Integer)hasil[k][6] - (Integer) hasil[k][8]);
                    double bobot = 5;
                    System.out.println("GAP : "+a);
                      if ( a == -1) {
                       a= bobot + a + 0.5;
                      }
                       else if (a == -2){
                       a= bobot + a + 1;
```

```
                    }
                    else if ( a == 1){
                    a= bobot -a - 0.5;
                    }
                    else if ( a == 2){
                    a= bobot - a;
                    }
                    else if (a > 2){
                    a = bobot - a ;
                    }
                    else if (a < -2){
                    a = bobot - a +0.5;
                    }
                    else {
                    a = bobot;
                    }

                System.out.println("Bobot Nilai "+a);
                if ((Integer)hasil[k][4] == 1){ // jika (hasil[k][4]) kode_faktor = 1 maka bobot (b)
= bobot + a
                   b = (b + a);
                } else {
                   b = (b+a);
                }
                System.out.println(+k+"| penduduk "+hasil[k][5]+": "+a+" "+b); // hasil[k][5] -->
nama penduduk
                pjg++;
                System.out.println("");
             }
             b = b/pjg;
             System.out.println("Nilai rata-rata faktor : "+b);
             totalFaktor[i][j] = b;
             bil++;
             System.out.println("");
          }

      }
      System.out.println("---------------------------------------");
         System.out.println("Penghitungan nilai total");
         System.out.println("(40% untuk sub kriteria secondary faktor) & (60% untuk sub
kriteria core faktor)");
         System.out.println("");
         for (int i = 0; i < kriteria.length; i++) {
            System.out.println("== Nilai Total Tiap Kriteria==");
            for (int j = 0; j < faktor.length; j++) {
               totalKriteria[i] = totalKriteria[i] + (totalFaktor[i][j] * ((Integer)faktor[j][2]/100.0));
```

```
                System.out.println(totalKriteria[i]+" = (rata-rata factor) "+totalFaktor[i][j]+" *
"+(Integer)faktor[j][2]+"/"+100);
            }
            System.out.println("");
            totalKriteria[kriteria.length] = totalKriteria[kriteria.length] + (totalKriteria[i] *
(Integer)kriteria[i][2]/100.0);
        }
            return totalKriteria;
   }
      private boolean showTable(JTable tabel, DefaultTableModel tabelModel){
      Database database = new Database();
      int kodePenduduk = idPenduduk[cmbnamakk.getSelectedIndex()];
      try {
         Object[][] data = database.getNilaiPenduduk(kodePenduduk);
         Object[] dataTable = new Object[data[0].length+1];
         Table createTable = new Table();
         tabelModel = createTable.getDefaultTabelModel(
         new String[]{" Nama Sub Kriteria ","Nilai Penduduk", "Nilai Profil", "Gap"});
         tabel.setModel(tabelModel);
         tabel.getTableHeader().setResizingAllowed(false);
         tabel.getTableHeader().setReorderingAllowed(false);
         int[][] hasil = new int[data.length][3];
         int indeks = 0;
         for (int i = 0; i < tabel.getColumnCount(); i++) {
            tabel.getColumnModel().getColumn(i).setResizable(false);
         }

         tabelModel.setRowCount(0);
         for (int i = 0; i < data.length; i++) {
            hasil[indeks][0] = (Integer) data[i][1];
            hasil[indeks][1] = (Integer) data[i][4];
            hasil[indeks][2] = hasil[indeks][0] - hasil[indeks][1];
            dataTable[0] = data[i][7];
            dataTable[1] = hasil[indeks][0];
            dataTable[2] = hasil[indeks][1];
            dataTable[3] = hasil[indeks][2];
            tabelModel.addRow(dataTable);
         }
         return true;
      } catch (Exception e) {
         return false;
      }
      }
private void btnhitungActionPerformed(java.awt.event.ActionEvent evt) {
 showTable(tabel,tabelModel);
      int kodePenduduk = idPenduduk[cmbnamakk.getSelectedIndex()];
      double[] coba = getFaktor(kodePenduduk);
```

```
      for (int i = 0; i < coba.length; i++) {
         System.out.println(i+": "+coba[i]);
      tabelNilai.setValueAt(decimalFormat.format(coba[i]), 0, i+1);
      }
      tabelNilai.setValueAt(cmbnamakk.getSelectedItem().toString(), 0, 0);
}

   public static void main(String args[]) {
      try {
         for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
               javax.swing.UIManager.setLookAndFeel(info.getClassName());
               break;
            }          }
      } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(Perhitungan.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
      } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(Perhitungan.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
      } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Perhitungan.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
      } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Perhitungan.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
      }
      java.awt.EventQueue.invokeLater(new Runnable() {

         public void run() {
            new Perhitungan().setVisible(true);
         }
      });
   }
   private javax.swing.JButton btnhitung;
   private javax.swing.JComboBox cmbnamakk;
   private javax.swing.JLabel jLabel1;
   private javax.swing.JLabel jLabel2;
   private javax.swing.JPanel jPanel1;
   private javax.swing.JPanel jPanel2;
   private javax.swing.JPanel jPanel3;
   private javax.swing.JPanel jPanel4;
   private javax.swing.JPanel jPanel5;
   private javax.swing.JScrollPane jScrollPane1;
   private javax.swing.JScrollPane jScrollPane2;
```

```
   private javax.swing.JTable tabel;
   private javax.swing.JTable tabelNilai;
   }
```

User.java

```
package skripsi;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;
public class User extends javax.swing.JFrame {
   private boolean ubah;
   private int idUser;
   private String username;
   private String password;
   private String posisi;
   private DefaultTableModel tabelModel;

  public User() {
      initComponents();
      showTable(tabel, tabelModel);
   }
private void batal(boolean b){
      txtuser.setEnabled(b);
      txtpass.setEnabled(b);
       }
    private boolean showTable(JTable tabel, DefaultTableModel tabelModel){
      Database database = new Database();
      try {
         Object[][] data = database.getUser();
         Object[] dataTable = new Object[data[0].length+1];
         Table createTable = new Table();
         tabelModel = createTable.getDefaultTabelModel(
         new String[]{"Kode User", "Username", "Password","Posisi"});
         tabel.setModel(tabelModel);
         tabel.getTableHeader().setResizingAllowed(false);
         tabel.getTableHeader().setReorderingAllowed(false);
         for (int i = 0; i < tabel.getColumnCount(); i++) {
            tabel.getColumnModel().getColumn(i).setResizable(false);
         }
         tabelModel.setRowCount(0);
         for (int i = 0; i < data.length; i++) {
            dataTable[0] = data[i][0];
            dataTable[1] = data[i][1];
            dataTable[2] = data[i][2];
            dataTable[3] = data[i][3];
```

```
            tabelModel.addRow(dataTable);
          }
          return true;
       } catch (Exception e) {
          return false;
       }   }
    private int getIDUser(JTextField textID){
       int hasil = 0;
       try {
          hasil = Integer.parseInt(textID.getText());
       } catch (NumberFormatException numberFormatException) {
       }
       return hasil;
   }
private boolean validasi(JTextField txtuser, JTextField txtpass){
       if (txtuser.getText().equals("")){
          JOptionPane.showMessageDialog(null, "Username Harus Diisi!", "Peringatan!",
JOptionPane.WARNING_MESSAGE);
          return false;
       } else if (txtpass.getText().equals("")){
          JOptionPane.showMessageDialog(null, "Password Harus Diisi!", "Peringatan!",
JOptionPane.WARNING_MESSAGE);
          return false;
       } else {
          return true;
       }   }
   private void ubah(){
       if(tabel.getSelectedRow() >= 0){
       int baris = tabel.getSelectedRow();
          idUser = (Integer) tabel.getValueAt(baris, 0);
          username = (String) tabel.getValueAt(baris, 1);
          password = (String) tabel.getValueAt(baris, 2);
          posisi = (String) tabel.getValueAt(baris, 3);
          txtid.setText(""+idUser);
          txtuser.setText(username);
          txtpass.setText(password);
          txtposisi.setText(posisi);

   }
       else{
        JOptionPane.showMessageDialog(null, "Pilih Data Yang akan diedit", "Pemberitahuan",
JOptionPane.INFORMATION_MESSAGE);
       }       }
   private void hapus(int idUser){
       Database database = new Database();
       if (database.deleteUser(idUser)){
```

```
            JOptionPane.showMessageDialog(null, "Berhasil Menghapus Data User",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(null, "Gagal Menghapus Data User", "Peringatan",
JOptionPane.WARNING_MESSAGE);
        }
    }
    public void kosong()
    {
        txtuser.setText("");
        txtpass.setText("");
        txtposisi.setText("");
        txtid.setText("");
        }
    private void simpan(int idUser, String username, String password, boolean ubah){
        Database database = new Database();
        Object[] data = {idUser, username, password};
        if (!ubah){
            if (database.insertUser(data)){
                JOptionPane.showMessageDialog(null, "Berhasil Menginput Data User",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Gagal Menginput Data User", "Peringatan",
JOptionPane.WARNING_MESSAGE);
            }
        } else {
            if (database.updateUser(data)){
                JOptionPane.showMessageDialog(null, "Berhasil Mengupdate Data User",
"Pemberitahuan", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Gagal Mengupdate Data User",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
            }
        }   }
private void btnhapusActionPerformed(java.awt.event.ActionEvent evt) {
try {
            int baris = tabel.getSelectedRow();
            idUser = (Integer) tabel.getValueAt(baris, 0);
            int jawab = JOptionPane.showOptionDialog(null, "Anda Yakin Akan Menghapus data
ini?", "Pertanyaan", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
null,new String[]{"Ya", "Tidak"},null);
            if (jawab == 0) {
                hapus(idUser);
                showTable(tabel, tabelModel);
            } else {
            }
        } catch (Exception e) {
```

```
            JOptionPane.showMessageDialog(null, "Pilih Data Yang Akan Dihapus Dulu",
"Peringatan!", JOptionPane.WARNING_MESSAGE);
        }
}
private void btnresetActionPerformed(java.awt.event.ActionEvent evt) {
 kosong();
        batal(false);
        btnsimpan.setEnabled(false);
        btnreset.setEnabled(false);
}
private void btntambahActionPerformed(java.awt.event.ActionEvent evt) {
        ubah = false;
        batal(true);
        btnsimpan.setEnabled(true);
        btnreset.setEnabled(true);
}
private void btnubahActionPerformed(java.awt.event.ActionEvent evt) {
       ubah = true;
       ubah();
       batal(false);
       batal(true);
       btnsimpan.setEnabled(true);
       btnreset.setEnabled(true);
}
private void btnsimpanActionPerformed(java.awt.event.ActionEvent evt) {
            idUser = getIDUser(txtid);
            username = txtuser.getText();
            password = txtpass.getText();
            if (validasi(txtuser, txtpass)) {
            simpan(idUser, username, password,ubah);
            btnsimpan.setEnabled(false);
            btnreset.setEnabled(false);
            batal(false);
            kosong();
            showTable(tabel, tabelModel);
        } }

    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }           }
```

```
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(User.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(User.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(User.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(User.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }

        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new User().setVisible(true);
            }
        });
    }
    private javax.swing.JButton btnhapus;
    private javax.swing.JButton btnreset;
    private javax.swing.JButton btnsimpan;
    private javax.swing.JButton btntambah;
    private javax.swing.JButton btnubah;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JPanel jPanel5;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tabel;
    private javax.swing.JTextField txtid;
    private javax.swing.JTextField txtpass;
    private javax.swing.JTextField txtposisi;
    private javax.swing.JTextField txtuser;
}
```

LaporanHasil.java

```
package skripsi;
import java.sql.Connection;
import java.sql.SQLException;
import java.text.DecimalFormat;
import java.util.Calendar;
import java.util.HashMap;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.view.JasperViewer;
public class LaporanHasil extends javax.swing.JFrame {
   private DefaultTableModel tabelModel;
   private Object[][] data = null;
      public LaporanHasil() {
      initComponents();
   }
   private boolean showTable(JTable tabel, DefaultTableModel tabelModel, Object[][] data){
      if (data == null){
         JOptionPane.showMessageDialog(null, "Data Tidak Ditemukan!", "Informasi",
JOptionPane.WARNING_MESSAGE);
      }
      DecimalFormat decimalFormat = new DecimalFormat("#.###");
      try {
         Object[] dataTable = new Object[8];
         Table createTable = new Table();
         tabelModel = createTable.getDefaultTabelModel(
               new String[]{"nomor","Nomor KK", "Nama KK", "RT", "RW", "Nilai",
"Periode","Status"});
         tabel.setModel(tabelModel);
         tabel.getTableHeader().setResizingAllowed(false);
         tabel.getTableHeader().setReorderingAllowed(false);

         for (int i = 0; i < tabel.getColumnCount(); i++) {
            tabel.getColumnModel().getColumn(i).setResizable(true);
         }
         tabelModel.setRowCount(0);
```

```
        for (int i = 0; i < data.length; i++) {
            dataTable[0] = i+1;
            dataTable[1] = data[i][1];
            dataTable[2] = data[i][2];
            dataTable[3] = data[i][3];
            dataTable[4] = data[i][4];
            dataTable[5] = decimalFormat.format(data[i][5]);
            dataTable[6] = data[i][6];
            dataTable[7] = data[i][7];
            tabelModel.addRow(dataTable);
        }
        return true;
    } catch (Exception e) {
        return false;
    }    }
private int[][] getNilai(int kodePenduduk){
    Database database = new Database();
    Object[][] nilai = database.getNilaiPenduduk(kodePenduduk);
    int[][] hasil = new int[nilai.length][3];
    int indeks = 0;
    for (int i = 0; i < nilai.length; i++) {
        for (int j = 0; j < nilai[i].length; j++) {
            hasil[indeks][0] = (Integer) nilai[i][1];
            hasil[indeks][1] = (Integer) nilai[i][4];
            hasil[indeks][2] = hasil[indeks][0] - hasil[indeks][1];
        }
        indeks++;
    }
    return hasil;
}

private double[] getFaktor(int kodePenduduk){
    Database database = new Database();
    Object[][] kriteria = database.getKriteria();
    Object[][] faktor = database.getFaktor();
    double totalFaktor[][] = new double[kriteria.length+1][faktor.length];
    double totalKriteria[] = new double[kriteria.length+1];
    int bil = 0;
    for (int i = 0; i < kriteria.length; i++) {
            for (int j = 0; j < faktor.length; j++) {
                int kodeKriteria = (Integer) kriteria[i][0];
                int kodeFaktor = (Integer) faktor[j][0];
       Object[][] hasil = database.getNilaiByVar(kodeKriteria, kodePenduduk, kodeFaktor);
                double b = 0;
                int pjg = 0;
                for (int k = 0; k < hasil.length; k++) {
                    double a = ((Integer)hasil[k][6] - (Integer) hasil[k][8]);
```

```
                double bobot = 5;
                if ( a == -1) {
                   a= bobot + a + 0.5;
                  }
                   else if (a == -2){
                   a= bobot + a + 1;
                   }
                   else if ( a == 1){
                   a= bobot -a - 0.5;
                   }
                   else if ( a == 2){
                   a= bobot - a;
                   }
                   else if (a > 2){
                   a = bobot - a ;
                   }
                   else if (a < -2){
                   a = bobot - a +0.5;
                   }
                   else {
                   a = bobot;
                   }
                if ((Integer)hasil[k][4] == 1){
                   b = (b + a);
                } else {
                   b = (b+a);
                }
                pjg++;
             }
             b = b/pjg;
             totalFaktor[i][j] = b;
             bil++;
          }
       }
       for (int i = 0; i < kriteria.length; i++) {
          for (int j = 0; j < faktor.length; j++) {
             totalKriteria[i] = totalKriteria[i] + (totalFaktor[i][j] * ((Integer)faktor[j][2]/100.0));
          }
          totalKriteria[kriteria.length] = totalKriteria[kriteria.length] + (totalKriteria[i] *
(Integer)kriteria[i][2]/100.0);
       }
     return totalKriteria;
  }
     private Object[][] sortData(Object[][] penduduk) {
    for (int i = 0; i < penduduk.length; i++) {
       for (int j = 0; j < penduduk.length-1; j++) {
          double nilaiPembanding = (Double) penduduk[j+1][5];
```

```
            double nilaiAwal = (Double) penduduk[j][5];
            if (nilaiAwal < nilaiPembanding){
               penduduk[j][5] = nilaiPembanding;
               penduduk[j+1][5] = nilaiAwal;
               String kode = (String) penduduk[j][1];
               penduduk[j][1] = penduduk[j+1][1];
               penduduk[j+1][1] = kode;
               String nama = (String) penduduk[j][2];
               penduduk[j][2] = penduduk[j+1][2];
               penduduk[j+1][2] = nama;
               int rt = (Integer)penduduk[j][3];
               penduduk[j][3] = penduduk[j+1][3];
               penduduk[j+1][3] = rt;
               int rw = (Integer) penduduk[j][4];
               penduduk[j][4] = penduduk[j+1][4];
               penduduk[j+1][4] = rw;

         }          }
      }
      return penduduk;

}
   private Object[][] getDataPenduduk() {
      Database database = new Database();
      Object[][] penduduk = database.getPendudukc();
      Object[][] hasil = new Object[penduduk.length][8];
      int jum = Integer.parseInt(txtjum.getText());
      if (penduduk.length <= jum){
      for (int i = 0; i < penduduk.length; i++) {
         int kodePenduduk = (Integer) penduduk[i][0];
         int[][] nilai = getNilai(kodePenduduk);
         double[] coba = getFaktor(kodePenduduk);
         String status = "Diterima";
         hasil[i][0] = i+1;
         hasil[i][1] = penduduk[i][1];
         hasil[i][2] = penduduk[i][2];
         hasil[i][3] = penduduk[i][3];
         hasil[i][4] = penduduk[i][4];
         hasil[i][5] = coba[coba.length-1]*1.0;
         hasil[i][6] = penduduk[i][5];
         hasil[i][7] = status;
      }}
      else if (penduduk.length > jum){
         for (int i = 0; i < jum; i++) {
         int kodePenduduk = (Integer) penduduk[i][0];
         int[][] nilai = getNilai(kodePenduduk);
         double[] coba = getFaktor(kodePenduduk);
```

```
            String status = "Diterima";
            hasil[i][0] = i+1;
            hasil[i][1] = penduduk[i][1];
            hasil[i][2] = penduduk[i][2];
            hasil[i][3] = penduduk[i][3];
            hasil[i][4] = penduduk[i][4];
            hasil[i][5] = coba[coba.length-1]*1.0;
            hasil[i][6] = penduduk[i][5];
            hasil[i][7] = status;
            }
            for (int i = jum; i < penduduk.length; i++) {
            int kodePenduduk = (Integer) penduduk[i][0];
            int[][] nilai = getNilai(kodePenduduk);
            double[] coba = getFaktor(kodePenduduk);
            String status = "Tidak Diterima";
            hasil[i][0] = i+1;
            hasil[i][1] = penduduk[i][1];
            hasil[i][2] = penduduk[i][2];
            hasil[i][3] = penduduk[i][3];
            hasil[i][4] = penduduk[i][4];
            hasil[i][5] = coba[coba.length-1]*1.0;
            hasil[i][6] = penduduk[i][5];
            hasil[i][7] = status;
            }
        }
        return hasil;
    }
private void btnlihatActionPerformed(java.awt.event.ActionEvent evt) {
        Object[][] penduduk = null;
        penduduk = getDataPenduduk();
        penduduk = sortData(penduduk);
        showTable(tabel, tabelModel, penduduk);
        data = penduduk;
        btncetak.setEnabled(true);
         Database database = new Database();
        Object[] data = new Object[4];
        for (int i = 0; i < tabel.getRowCount(); i++) {
            data[0] = tabel.getValueAt(i, 1);
            data[1] = tabel.getValueAt(i, 5);
            data[2] = tabel.getValueAt(i, 6);
            data[3] = tabel.getValueAt(i, 7);
            database.insertNA(data);
        }}
private void btncetakActionPerformed(java.awt.event.ActionEvent evt) {
            try {
            Connection koneksi = new Database().konek();
            String report = "src/skripsi/laporan/nilai.jrxml";
```

```
            HashMap hashMap = new HashMap();
            JasperReport jasperReport = JasperCompileManager.compileReport(report);
            JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, hashMap,
koneksi);
            JasperViewer.viewReport(jasperPrint,false);
            koneksi.close();
        } catch (JRException ex) {
            Logger.getLogger(Skripsi.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(Skripsi.class.getName()).log(Level.SEVERE, null, ex);
        } catch (Exception ex) {
            Logger.getLogger(Skripsi.class.getName()).log(Level.SEVERE, null, ex);
        }
        btncetak.setEnabled(false);
}
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }           }
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(LaporanHasil.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(LaporanHasil.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(LaporanHasil.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(LaporanHasil.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new LaporanHasil().setVisible(true);
            }
        });
    }
    private javax.swing.JButton btncetak;
    private javax.swing.JButton btnlihat;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
```

```
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tabel;
    private javax.swing.JTextField txtjum;
}
```

LaporanNilai.java

```
package skripsi;
import java.sql.Connection;
import java.sql.SQLException;
import java.text.DecimalFormat;
import java.util.HashMap;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.view.JasperViewer;
public class LaporanNilai extends javax.swing.JFrame {
    private DefaultTableModel tabelModel;
    private DecimalFormat decimalFormat;
        public LaporanNilai() {
        initComponents();
        decimalFormat = new DecimalFormat("#.###");
    }
  private boolean showTable(JTable tabel, DefaultTableModel tabelModel){
        Database database = new Database();
        try {
            Object[][] data = database.getPendudukb();
            Object[] dataTable = new Object[17];
            Table createTable = new Table();
            tabelModel = createTable.getDefaultTabelModel(
            new String[]{"NO","Nomor KK", "Nama KK", "Luas Lantai", "Jenis dinding", "Jenis
lantai", "Penerangan", "Sarana Memasak", "Sumber Air", "Jumlah penghasilan",
"pendidikan","Sanggup Pakaian","Sanggup Kesehatan"});
            tabel.setModel(tabelModel);
            tabel.getTableHeader().setResizingAllowed(true);
            tabel.getTableHeader().setReorderingAllowed(false);
            for (int i = 0; i < tabel.getColumnCount(); i++) {
                tabel.getColumnModel().getColumn(i).setResizable(true);
            }
```

```
            tabelModel.setRowCount(0);
            for (int i = 0; i < data.length; i++) {
                Object[][] dataNilai = database.getNilaiPenduduk((Integer)data[i][0]);
                System.out.println(""+dataNilai.length);
                dataTable[0] = i+1;
                dataTable[1] = data[i][1];
                dataTable[2] = data[i][2];
                for (int j = 0; j < dataNilai.length; j++) {
                    dataTable[3+j] = dataNilai[j][1];
                }
                tabelModel.addRow(dataTable);
                for (int j = 0; j < dataNilai.length; j++) {
                    dataTable[3+j] = "";
                }          }
            return true;
        } catch (Exception e) {
            return false;
        }   }
    private void btnlihatActionPerformed(java.awt.event.ActionEvent evt) {
        showTable(tabel, tabelModel);
        btncetak.setEnabled(true);
    }
    private void btncetakActionPerformed(java.awt.event.ActionEvent evt) {
        Database database = new Database();
        database.deleteNilaiPenduduk();
        Object[] data = new Object[12];
        for (int i = 0; i < tabel.getRowCount(); i++) {
            data[0] = tabel.getValueAt(i, 0);
            data[1] = tabel.getValueAt(i, 1);
            data[2] = tabel.getValueAt(i, 3);
            data[3] = tabel.getValueAt(i, 4);
            data[4] = tabel.getValueAt(i, 5);
            data[5] = tabel.getValueAt(i, 6);
            data[6] = tabel.getValueAt(i, 7);
            data[7] = tabel.getValueAt(i, 8);
            data[8] = tabel.getValueAt(i, 9);
            data[9] = tabel.getValueAt(i, 10);
            data[10]= tabel.getValueAt(i, 11);
            data[11]= tabel.getValueAt(i, 12);
            database.insertNilaiPenduduk(data);
                }
        try {
            Connection koneksi = new Database().konek();
            String report = "src/skripsi/laporan/laporannilaipenduduk.jrxml";
            HashMap hashMap = new HashMap();
            JasperReport jasperReport = JasperCompileManager.compileReport(report);
         JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, hashMap, koneksi);
```

```
            JasperViewer.viewReport(jasperPrint,false);
            koneksi.close();
        } catch (JRException ex) {
            Logger.getLogger(Skripsi.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(Skripsi.class.getName()).log(Level.SEVERE, null, ex);
        } catch (Exception ex) {
            Logger.getLogger(Skripsi.class.getName()).log(Level.SEVERE, null, ex);
        }
        btncetak.setEnabled(false);
}
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }           }
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(LaporanNilai.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(LaporanNilai.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(LaporanNilai.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(LaporanNilai.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new LaporanNilai().setVisible(true);
            }
        });
    }
    private javax.swing.JButton btncetak;
    private javax.swing.JButton btnlihat;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tabel;
}
```

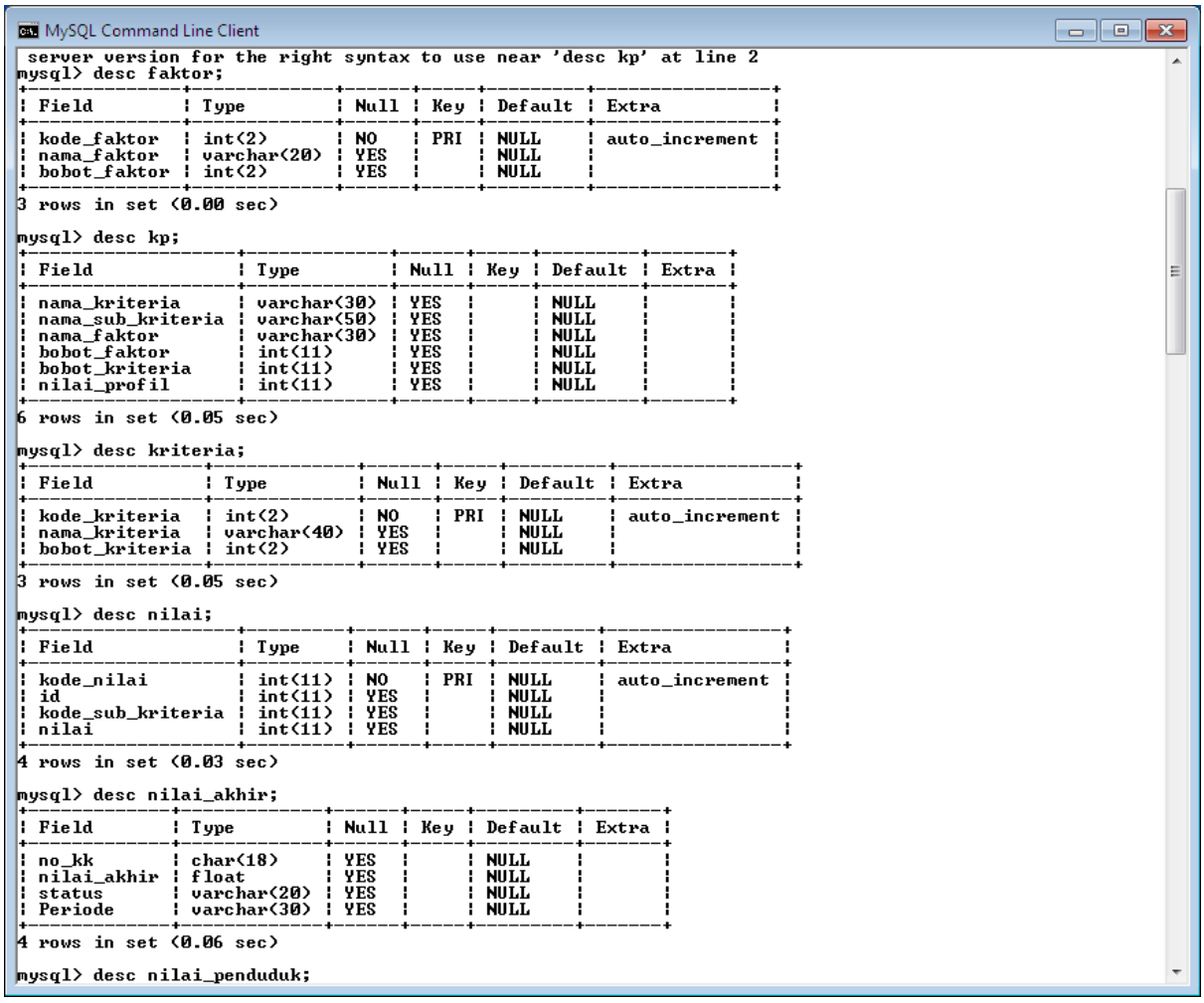
LaporanSubKriteria.java

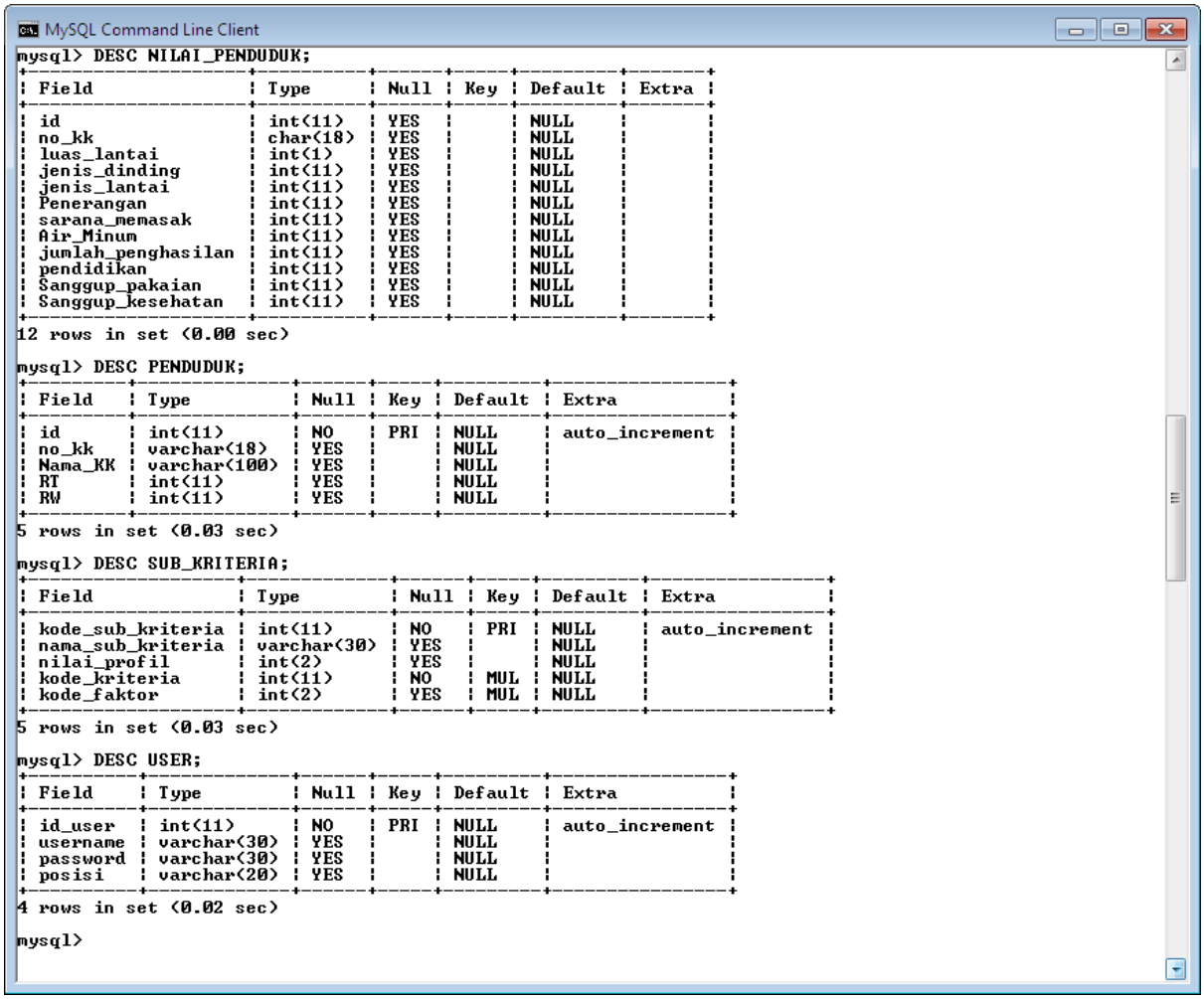
```
package skripsi;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JComboBox;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.view.JasperViewer;
public class LaporanSubKriteria extends javax.swing.JFrame {
 private DefaultTableModel tabelModel;
 public static int kodeKriteria[];
    public LaporanSubKriteria() {
       initComponents();
         kodeKriteria = createComboKriteria(cmbkriteria);
   }
   public static int[] createComboKriteria(JComboBox cmbkriteria){
       int[] hasil = null;
       Database database = new Database();
       try {
          Object[][] data = database.getKriteria();
          hasil = new int[data.length];
          cmbkriteria.removeAllItems();
          for (int i = 0; i < data.length; i++) {
             hasil[i] = (Integer) data[i][0];
             cmbkriteria.addItem(data[i][1]);
          }
       } catch (Exception e) {
       }
       return hasil;
   }
  private boolean showTable(JTable tabel, DefaultTableModel tabelModel,String kriteria){
       Database database = new Database();
       try {
          Object[][] data = database.getLaporanKrit(kriteria);
          Object[] dataTable = new Object[data[0].length+1];
          Table createTable = new Table();
          tabelModel = createTable.getDefaultTabelModel(
```

```
            new String[]{"No","Nama Kriteria", "Nama Sub Kriteria", "Nama Faktor", "Bobot
Faktor", "Bobot Kriteria","Nilai Profil"});
        tabel.setModel(tabelModel);
        tabel.getTableHeader().setResizingAllowed(true);
        tabel.getTableHeader().setReorderingAllowed(false);
        for (int i = 0; i < tabel.getColumnCount(); i++) {
           tabel.getColumnModel().getColumn(i).setResizable(true);
        }
        tabelModel.setRowCount(0);
        for (int i = 0; i < data.length; i++) {
           dataTable[0] = i+1;
           dataTable[1] = data[i][0];
           dataTable[2] = data[i][1];
           dataTable[3] = data[i][2];
           dataTable[4] = data[i][3];
           dataTable[5] = data[i][4];
           dataTable[6] = data[i][5];
           tabelModel.addRow(dataTable);
        }
        return true;
      } catch (Exception e) {
        return false;
      }   }
private void btnlihatActionPerformed(java.awt.event.ActionEvent evt) {
String idKriteria = cmbkriteria.getSelectedItem().toString().trim();
 showTable(tabel, tabelModel, idKriteria);
}
private void btncetakActionPerformed(java.awt.event.ActionEvent evt) {
 Database database = new Database();
      database.deleteKP();
      Object[] data = new Object[6];
      for (int i = 0; i < tabel.getRowCount(); i++) {
        data[0] = tabel.getValueAt(i, 1);
        data[1] = tabel.getValueAt(i, 2);
        data[2] = tabel.getValueAt(i, 3);
        data[3] = tabel.getValueAt(i, 4);
        data[4] = tabel.getValueAt(i, 5);
        data[5] = tabel.getValueAt(i, 6);
       database.insertKP(data);
             }
      try {
        Connection koneksi = new Database().konek();
        String report = "src/skripsi/laporan/laporannilaipenduduk.jrxml";
        HashMap hashMap = new HashMap();
        JasperReport jasperReport = JasperCompileManager.compileReport(report);
       JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, hashMap, koneksi);
        JasperViewer.viewReport(jasperPrint,false);
```

```
            koneksi.close();
        } catch (JRException ex) {
            Logger.getLogger(Skripsi.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SQLException ex) {
            Logger.getLogger(Skripsi.class.getName()).log(Level.SEVERE, null, ex);
        } catch (Exception ex) {
            Logger.getLogger(Skripsi.class.getName()).log(Level.SEVERE, null, ex);
        }
        btncetak.setEnabled(false);
}
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;    }          }
        } catch (ClassNotFoundException ex) {
java.util.logging.Logger.getLogger(LaporanSubKriteria.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
java.util.logging.Logger.getLogger(LaporanSubKriteria.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
java.util.logging.Logger.getLogger(LaporanSubKriteria.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(LaporanSubKriteria.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new LaporanSubKriteria().setVisible(true);
            }
        });    }
    private javax.swing.JButton btncetak;
    private javax.swing.JButton btnlihat;
    private javax.swing.JComboBox cmbkriteria;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tabel;
}
```

DATABASE :


MySQL Command Line Client
server version for the right syntax to use near 'desc kp' at line 2
mysql> desc faktor;
+--------------+-------------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+----------------+
kode_faktor	int(2)	NO	PRI	NULL	auto_increment
nama_faktor	varchar(20)	YES		NULL	
bobot_faktor	int(2)	YES		NULL	
+--------------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
mysql> desc kp;
+-------------------+-------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------------------+-------------+------+-----+---------+-------+
nama_kriteria	varchar(30)	YES		NULL	
nama_sub_kriteria	varchar(50)	YES		NULL	
nama_faktor	varchar(30)	YES		NULL	
bobot_faktor	int(11)	YES		NULL	
bobot_kriteria	int(11)	YES		NULL	
nilai_profil	int(11)	YES		NULL	
+-------------------+-------------+------+-----+---------+-------+
6 rows in set (0.05 sec)
mysql> desc kriteria;
+----------------+-------------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+----------------+
kode_kriteria	int(2)	NO	PRI	NULL	auto_increment
nama_kriteria	varchar(40)	YES		NULL	
bobot_kriteria	int(2)	YES		NULL	
+----------------+-------------+------+-----+---------+----------------+
3 rows in set (0.05 sec)
mysql> desc nilai;
+-------------------+---------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+-------------------+---------+------+-----+---------+----------------+
kode_nilai	int(11)	NO	PRI	NULL	auto_increment
id	int(11)	YES		NULL	
kode_sub_kriteria	int(11)	YES		NULL	
nilai	int(11)	YES		NULL	
+-------------------+---------+------+-----+---------+----------------+
4 rows in set (0.03 sec)
mysql> desc nilai_akhir;
+-------------+-------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
no_kk	char(18)	YES		NULL	
nilai_akhir	float	YES		NULL	
status	varchar(20)	YES		NULL	
Periode	varchar(30)	YES		NULL	
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.06 sec)
mysql> desc nilai_penduduk;


MySQL Command Line Client
mysql> DESC NILAI_PENDUDUK;
+---------------------+----------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+---------------------+----------+------+-----+---------+-------+
id	int(11)	YES		NULL	
no_kk	char(18)	YES		NULL	
luas_lantai	int(1)	YES		NULL	
jenis_dinding	int(11)	YES		NULL	
jenis_lantai	int(11)	YES		NULL	
Penerangan	int(11)	YES		NULL	
sarana_memasak	int(11)	YES		NULL	
Air_Minum	int(11)	YES		NULL	
jumlah_penghasilan	int(11)	YES		NULL	
pendidikan	int(11)	YES		NULL	
Sanggup_pakaian	int(11)	YES		NULL	
Sanggup_kesehatan	int(11)	YES		NULL	
+---------------------+----------+------+-----+---------+-------+
12 rows in set (0.00 sec)
mysql> DESC PENDUDUK;
+---------+-------------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+----------------+
id	int(11)	NO	PRI	NULL	auto_increment
no_kk	varchar(18)	YES		NULL	
Nama_KK	varchar(100)	YES		NULL	
RT	int(11)	YES		NULL	
RW	int(11)	YES		NULL	
+---------+-------------+------+-----+---------+----------------+
5 rows in set (0.03 sec)
mysql> DESC SUB_KRITERIA;
+-------------------+-------------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+-------------------+-------------+------+-----+---------+----------------+
kode_sub_kriteria	int(11)	NO	PRI	NULL	auto_increment
nama_sub_kriteria	varchar(30)	YES		NULL	
nilai_profil	int(2)	YES		NULL	
kode_kriteria	int(11)	NO	MUL	NULL	
kode_faktor	int(2)	YES	MUL	NULL	
+-------------------+-------------+------+-----+---------+----------------+
5 rows in set (0.03 sec)
mysql> DESC USER;
+----------+-------------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+----------------+
id_user	int(11)	NO	PRI	NULL	auto_increment
username	varchar(30)	YES		NULL	
password	varchar(30)	YES		NULL	
posisi	varchar(20)	YES		NULL	
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.02 sec)
mysql>