LAPORAN PEMROGRAMAN JARINGAN

"APLIKASI CRUD DATABASE MAHASISWA MENGGUNAKAN TEKNOLOGI RMI"

"Disusun untuk memenuhi salah satu tugas Pemrograman Jaringan."

Disusun oleh :

Arry Febryan	201610225200
Yusuf Bagas Satria	201610225257

Mata Kuliah : Pemrograman Jaringan

Dosen Pembimbing : Rakhmat Purnomo, S.Pd., S.Kom., M.Kom.

PROGRAM STUDI INFORMATIKA FAKULTAS TEKNIK

UNIVERSITAS BHAYANGKARA JAKARTA RAYA

JANUARI 2019

A. Tujuan

Tujuan dibuatnya laporan ini adalah:

1. Dibuatnya project ini bertujuan untuk menyelesaikan tugas akhir dari mata kuliah Pemrograman Jaringan.
2. Agar membantu memberi pemahaman kepada Mahasiswa bagaimana proses dari aplikasi client-server.
3. Agar Mahasiswa dapat membangun, dan tahu bagaimana cara membuat aplikasi database (CRUD) dengan menggunkan teknologi RMI.

B. Remote Method Invocation (RMI)

RMI (*Remote Method Invocation*) menyediakan sarana dimana client dan server dapat berkomunikasi dan saling bertukar infomasi. RMI memungkinkan pengembang perangkat lunak untuk merancang aplikasi terdistribusi dimana methods dari remote object dapat dipanggil dari JVM (*Java Virtual Mechine*) lain, yang mungkin berjalan pada host yang berbeda. Remote object adalah obyek dalam Java yang dapat direferensikan secara remote. Pemrogram seakan- akan memanggil methods lokal dari file kelas lokal, sedang dalam kenyataannya semua argumen dikirimkan ke remote target dan diinter-pretasikan, kemudian hasilnya dikirimkan kembali ke pemanggil. Dalam RMI, server akan membuat remote objects, membuat referensi, dan menunggu client untuk memanggil methods dari remote object ini. Sedangkan client akan mendapatkan remote reference dari satu atau lebih remote object dan memanggil methods untuk remote object tersebut.

C. Tutorial Membuat Program CRUD dengan Remote Method Invocation

Bismillaahir-rohmaanir-rohiim.

Pada tulisan ini, saya akan membuat tutorial pembuatan aplikasi desktop sederhana berbasis client-server dengan Bahasa Pemrograman Java, dan menggunakan teknologi RMI (Remote Method Invocation). Dalam Tutorial kali ini saya akan membuat aplikasi yang dapat melakukak proses CRUD untuk database Mahasiswa. Semoga tutorial ini dapat membantu dan memberikan pemahaman kepada para pembaca. Aamiin.

Sebelum memulai membuat program ada beberapa hal yang perlu dipersiapkan terlebih dahulu, yaitu sebuah laptop yang di dalamnya terinstal:

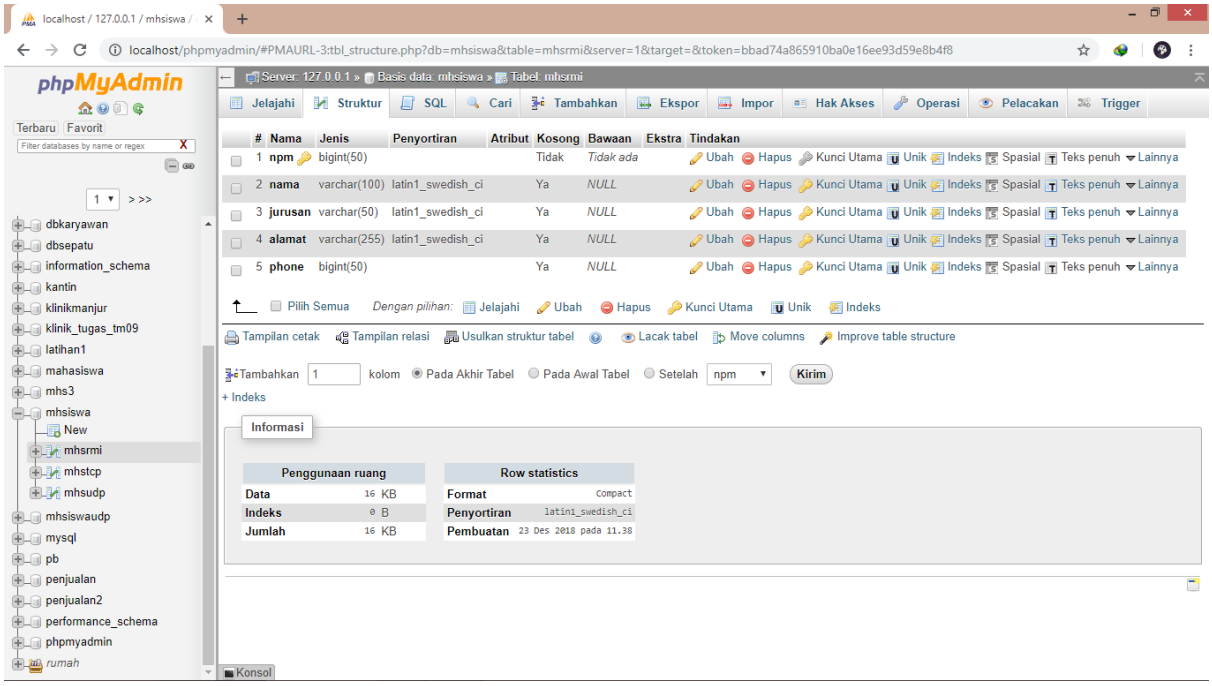
- Java Netbeans
- dan XAMPP

Setelah kebutuhan diatas terpenuhi, baru kita bisa melanjutkan ke langkah-langkah berikutnya.

1. Membuat Database

Langkah pertama yang harus kita lakukan adalah menyiapkan terlebih dahulu sebuah database beserta tabel dan entias-entitasnya. Nantinya database ini akan digunakan untuk menyimpan data-data yang akan di-input.

Disini saya akan membuat sebuah database dengan nama **mhsiswa** beserta struktur tabel sebagai berikut:



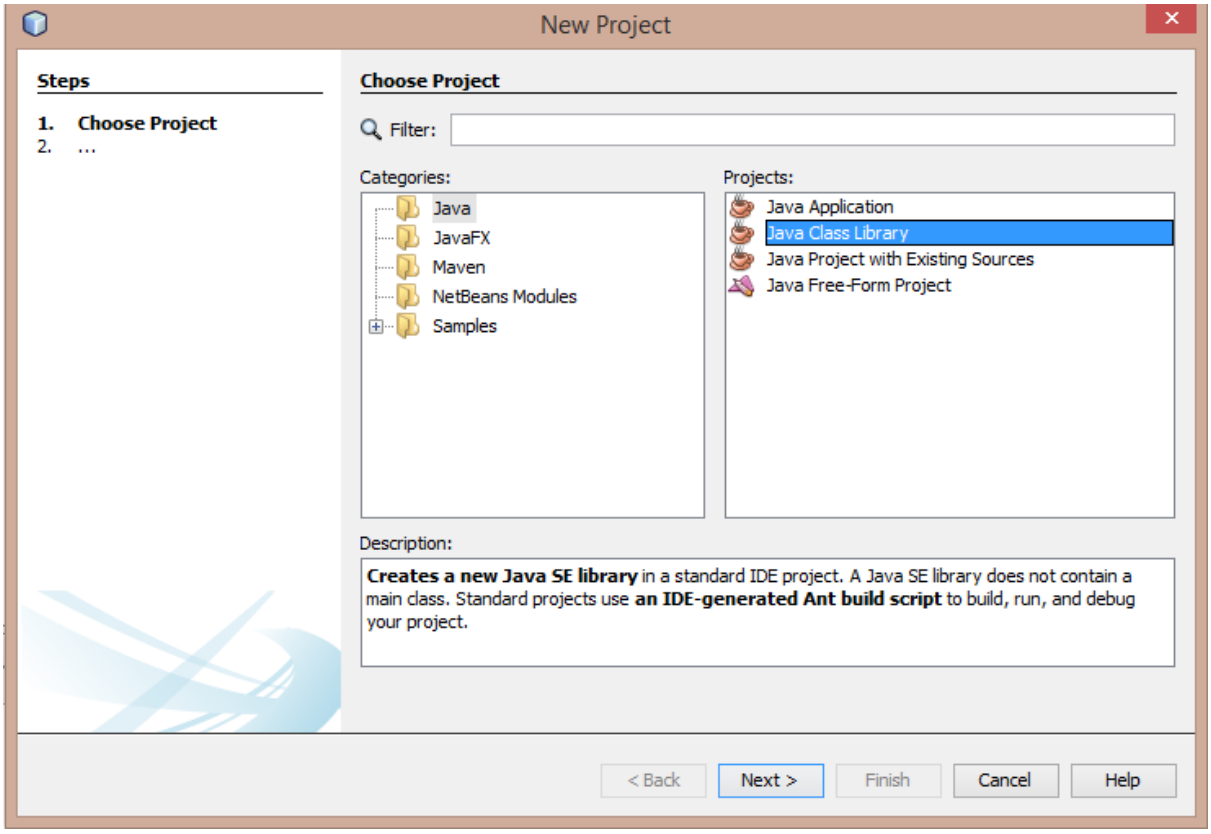
2. Membuat Project Java

Langkah selanjutnya adalah menyiapkan project java pada netbeans. Pada program kali ini kita akan membuat 3 project baru yaitu **Application-API, Application-Server** dan **Application-Client.**

2.1.Application API

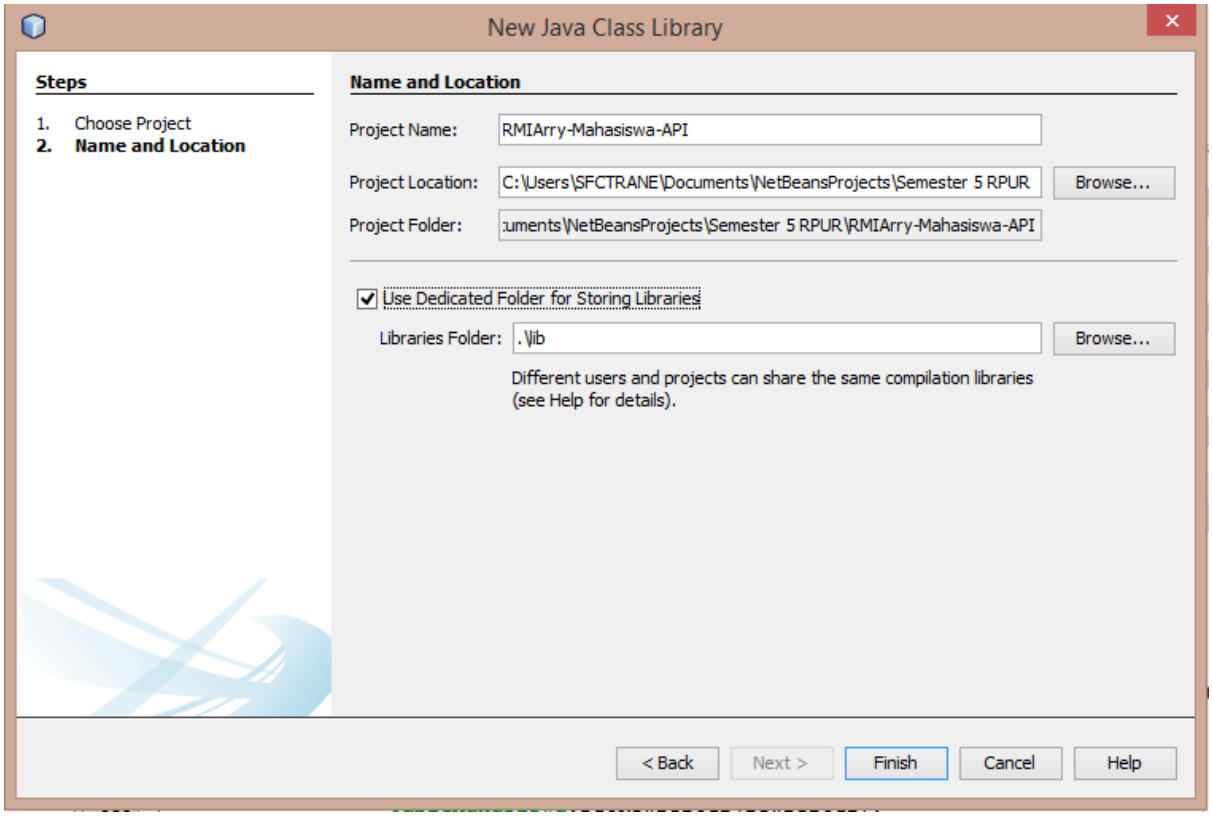
Porject pertama yang kita buat adalah project untuk **Application-API** yang nantinya project ini akan digunakan oleh server dan client sebagai remote object.

Untuk membuat project klik **File > New Project**, nanti akan muncul tampilan seperti ini:

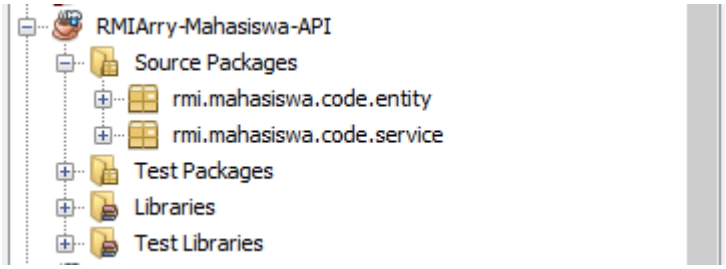


Pilih **Java Class Library** untuk project Application-API, lalu klik next.

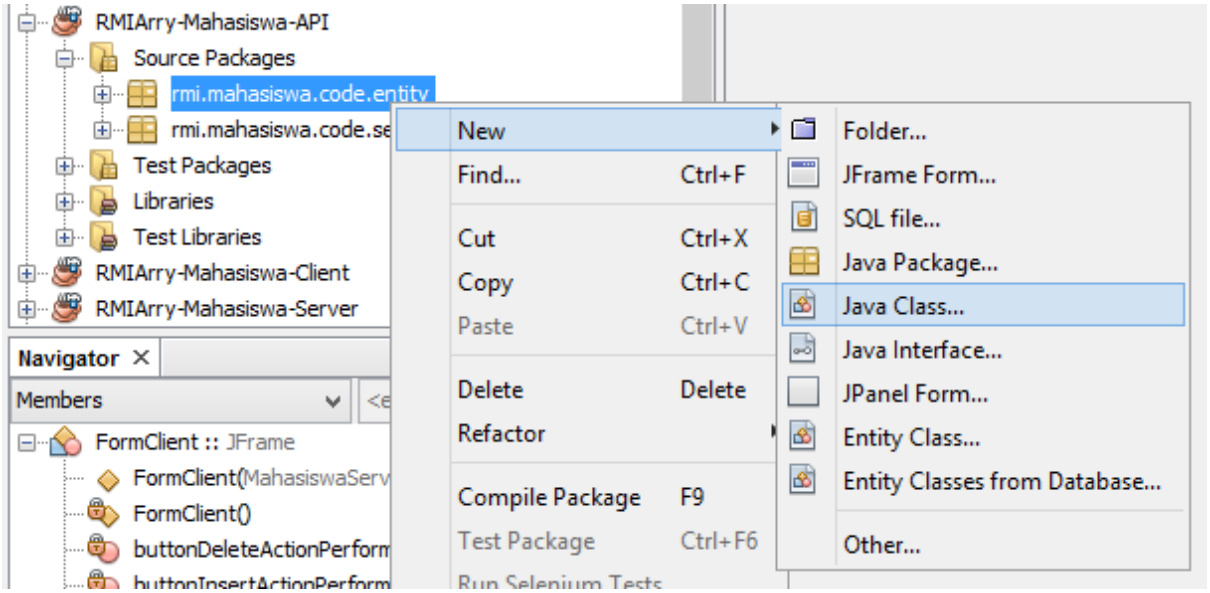
Setelah itu isilah nama project sesuai dengan yang kita inginkan, lalu klik finish. Untuk memudahkan, beri codename 'API' pada akhiran nama project API. Di sini saya memberi nama project API dengan nama **'RMIArry-Mahasiswa-API'**.



Selanjutya buatlah 2 buah package seperti gambar dibawah ini.



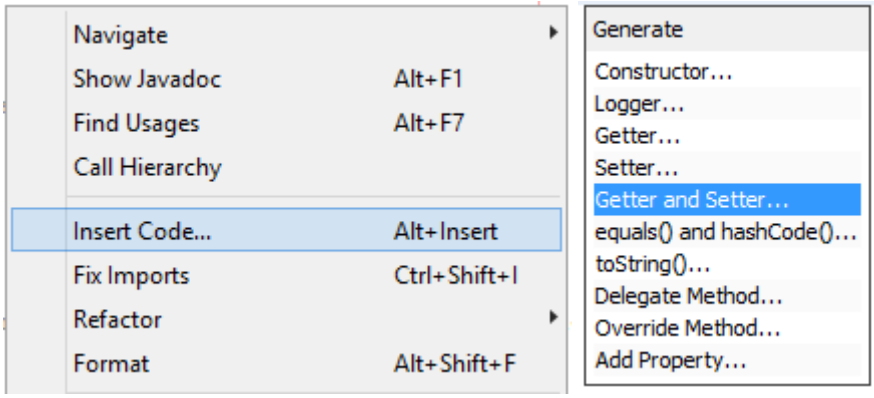
Setelah itu buatlah sebuah Java Class dengan nama ***Mahasiswa*** pada package entity dengan cara klik kanan pada package lalu klik **New > Java Class.**



Kemudian buatlah variable didalam class tersebut untuk mewakili tiap tiap entitas yang sebeumnya telah kita buat pada table dalam database.

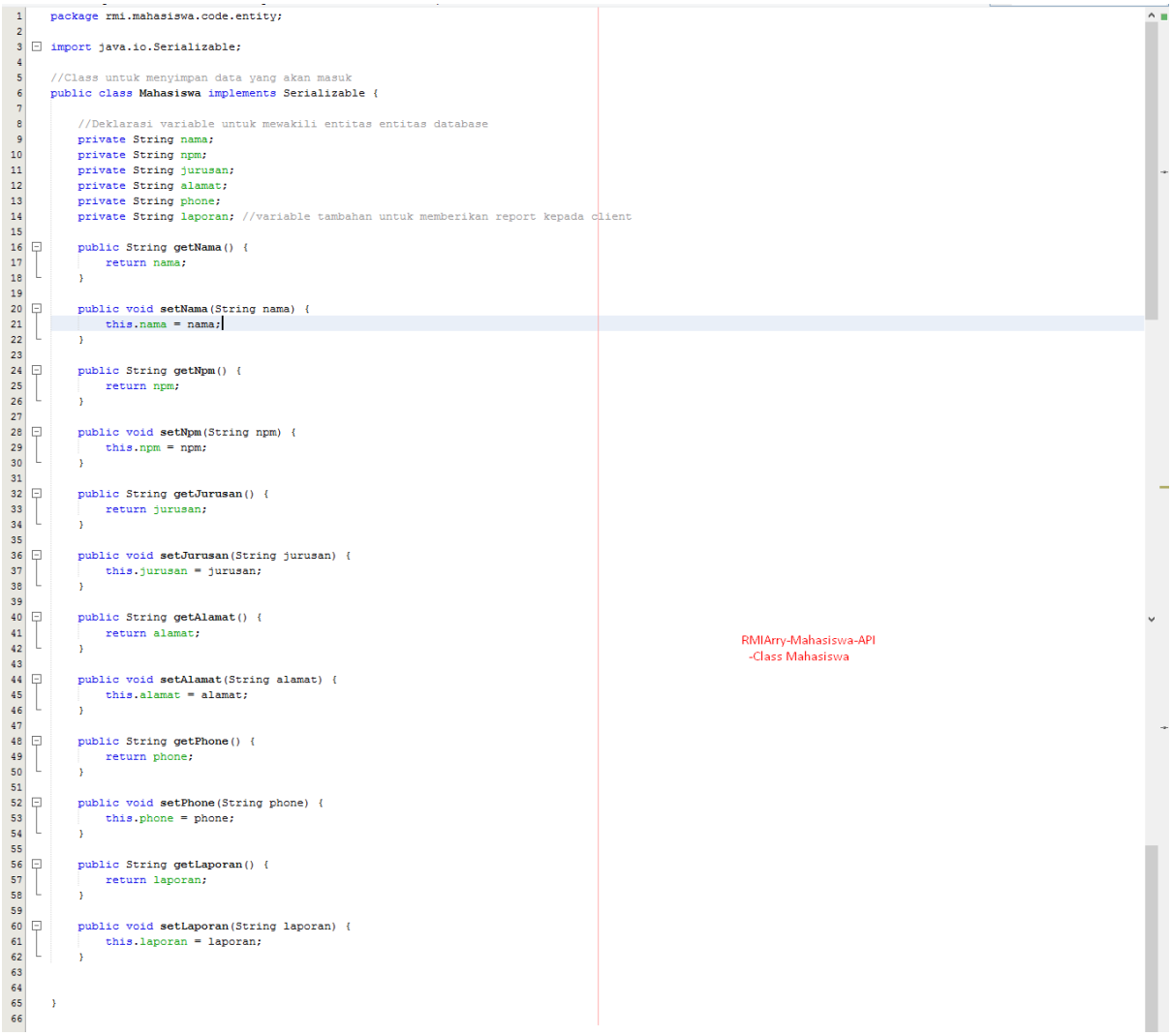
```
1  package rmi.mahasiswa.code.entity;
2
3  import java.io.Serializable;
4
5  //Class untuk menyimpan data yang akan masuk
6  public class Mahasiswa implements Serializable {
7
8      //Deklarasi variable untuk mewakili entitas entitas database
9      private String nama;
10     private String npm;
11     private String jurusan;
12     private String alamat;
13     private String phone;
14     private String laporan; //variable tambahan untuk memberikan report kepada client
15
16 }
17
18
```

Selanjutnya buatlah method **'Set'** dan **'Get'** untuk tiap variable dengan cara klik kanan pada kolom source code pilih **Insert Code** > **Getter and Setter.**



Lalu centang semua variable yang telah kita buat di class Mahasiswa, klik Generate.

Setelah itu akan muncul method Get dan Set seperti gambar dibawah ini:




```java
package rmi.mahasiswa.code.entity;

import java.io.Serializable;

//Class untuk menyimpan data yang akan masuk
public class Mahasiswa implements Serializable {

    //Deklarasi variable untuk mewakili entitas entitas database
    private String nama;
    private String npm;
    private String jurusan;
    private String alamat;
    private String phone;
    private String laporan; //variable tambahan untuk memberikan report kepada client

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getNpm() {
        return npm;
    }

    public void setNpm(String npm) {
        this.npm = npm;
    }

    public String getJurusan() {
        return jurusan;
    }

    public void setJurusan(String jurusan) {
        this.jurusan = jurusan;
    }

    public String getAlamat() {
        return alamat;
    }

    public void setAlamat(String alamat) {
        this.alamat = alamat;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getLaporan() {
        return laporan;
    }

    public void setLaporan(String laporan) {
        this.laporan = laporan;
    }


}

```

Langkah selanjutnya adalah membuat kelass interface dengan nama ***MahasiswaService*** pada package service, kelas inilah yang akan menjebatani antara client dan server.

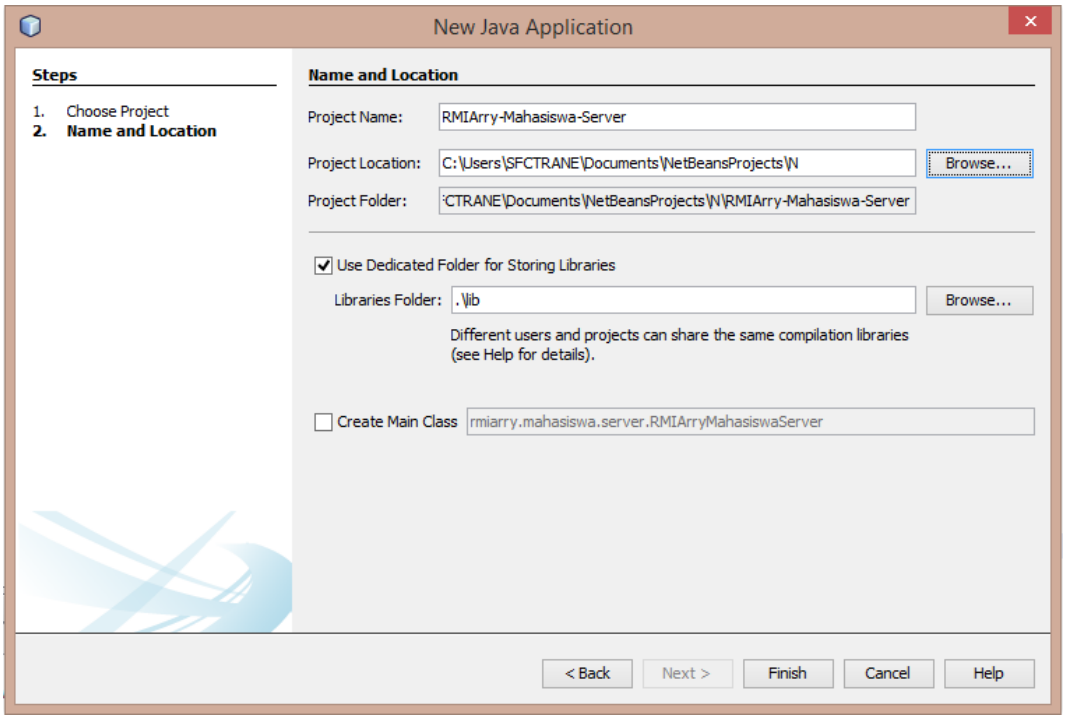
Klik kanan pada package service lalu pilih **New > Java Interface**. Lalu tulis kodingan dibawah ini:

```java
package rmi.mahasiswa.code.service;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
import rmi.mahasiswa.code.entity.Mahasiswa;

//Class Interface yang nantinya akan menjebatani permintaan client ke server
//sebagai Remote Object
public interface MahasiswaService extends Remote {

    Mahasiswa insertMahasiswa(Mahasiswa mahasiswa) throws RemoteException;

    Mahasiswa updateMahasiswa(Mahasiswa mahasiswa) throws RemoteException;

    Mahasiswa deleteMahasiswa(Mahasiswa mahasiswa) throws RemoteException;

    Mahasiswa getMahasiswa(String npm) throws RemoteException;

    List<Mahasiswa> getAllMahasiswa() throws RemoteException;
}
```

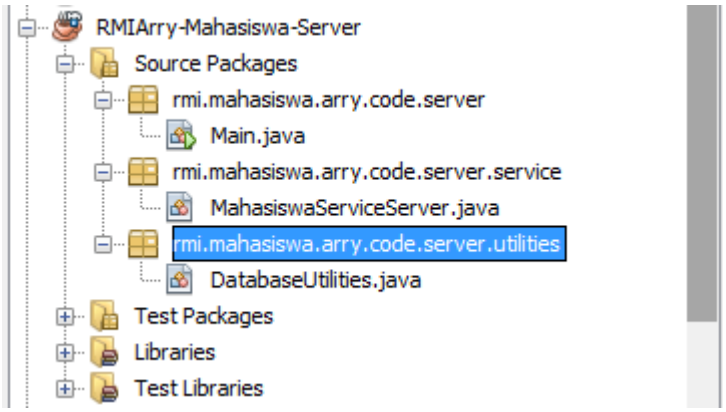
2.2. Application-Server

Setelah project Application-API selesai langkah selanjutnya adalah membuat project untuk **Application-Server** dengan cara klik **File > New Project > pilih Java Application**.

Lalu beri nama untuk project Application-Server sesuai dengan yang kita inginkan, beri codename Server diakhiran nama project untuk mempermudah kita mengenali project, disini saya memberi nama untuk project Server dengan nama **RMIArry-Mahasiswa-Server**.

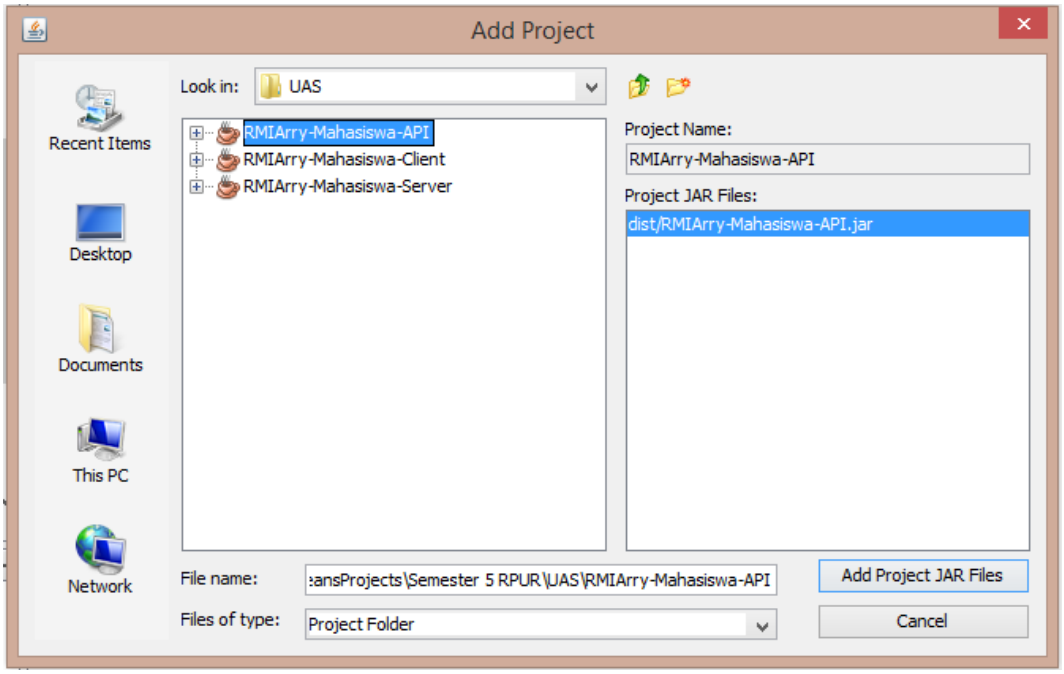


Setelah project terbuat, buatlah tiga buah project beserta classnya seperti dibawah ini:

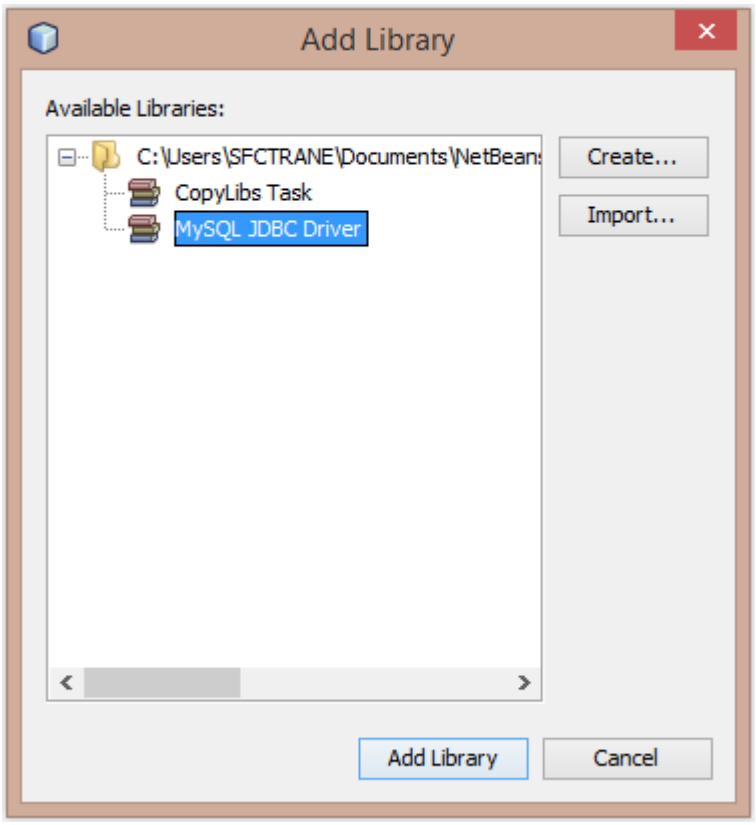


Kemudian import project Application-API yang sebelumnya kita buat kedalam project Server dengan cara klik kanan pada **Libraries > Add Project..**

Akan muncul tampilan seperti dibawah ini, pilih project Application-API lalu klik Add Project.



Selanjutnya import driver MySQL JDBC dengan cara klik kanan pada **Libraries > Add Librarys**



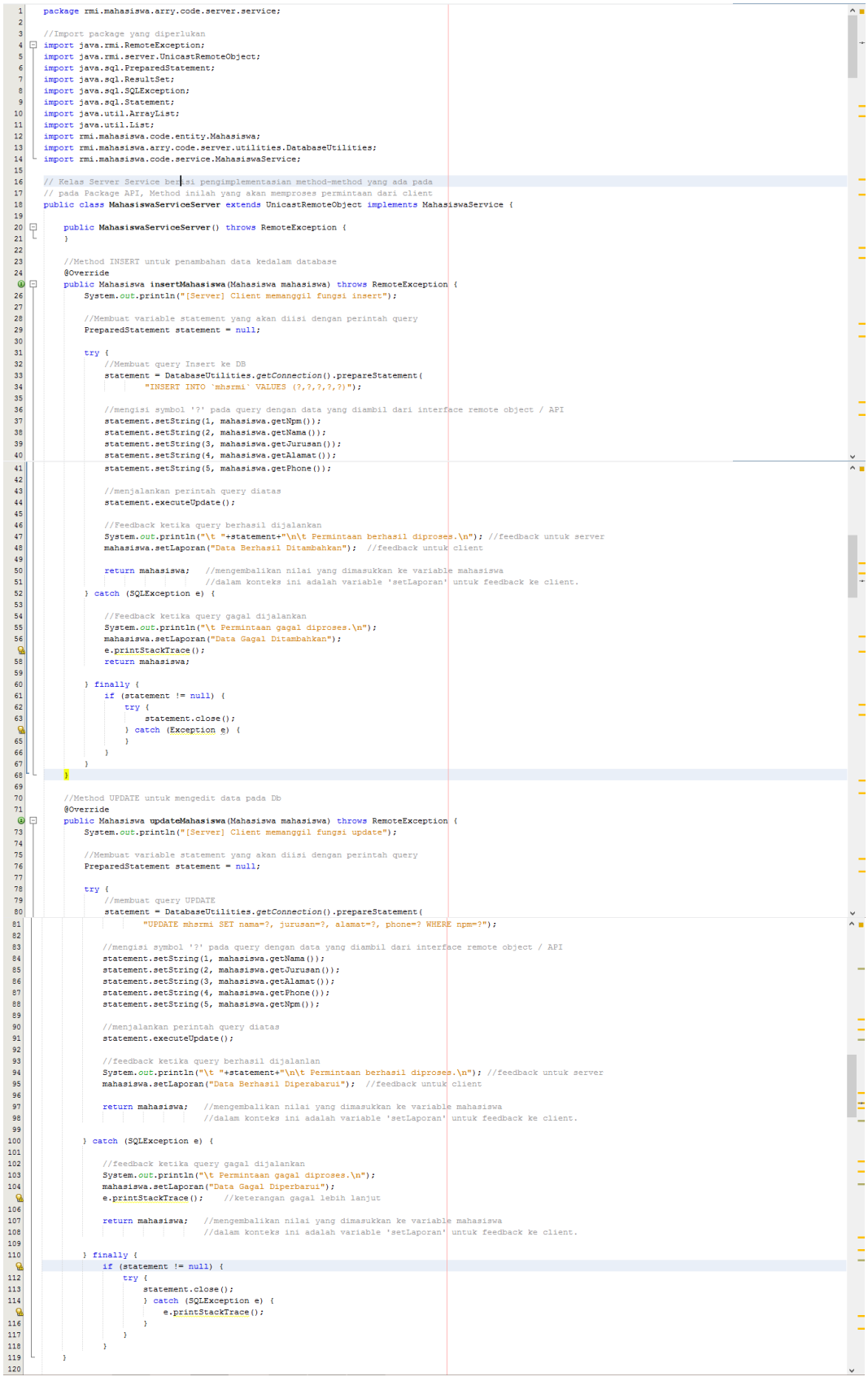
Lalu pilih **MySQL JDBC Driver**.

Langkah selanjutnya adalah mengisi class-class yang ada pada project Server dengan kode program sebagai berikut:

Class Utilities:

```
package rmi.mahasiswa.arry.code.server.utilities;

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

//Class untuk membuat koneksi dengan database
public class DatabaseUtilities{
    private static Connection connection;

    public static Connection getConnection() {
        if (connection == null) {
            try {
                DriverManager.registerDriver(new com.mysql.jdbc.Driver());
                //membuat koneksi dengan localhost dengan db 'mhsiswa', dengan user 'root'
                connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mhsiswa", "root", "");
            } catch (SQLException ex) {
                System.out.println(ex);
            }
        }
        return connection;
    }
}
```

Class Service:



```
package rmi.mahasiswa.arry.code.server.service;

//Import package yang diperlukan
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import rmi.mahasiswa.code.entity.Mahasiswa;
import rmi.mahasiswa.arry.code.server.utilities.DatabaseUtilities;
import rmi.mahasiswa.code.service.MahasiswaService;

// Kelas Server Service berisi pengimplementasian method-method yang ada pada
// pada Package API, Method inilah yang akan memproses permintaan dari client
public class MahasiswaServiceServer extends UnicastRemoteObject implements MahasiswaService {

    public MahasiswaServiceServer() throws RemoteException {
    }

    //Method INSERT untuk penambahan data kedalam database
    @Override
    public Mahasiswa insertMahasiswa(Mahasiswa mahasiswa) throws RemoteException {
        System.out.println("[Server] Client memanggil fungsi insert");

        //Membuat variable statement yang akan diisi dengan perintah query
        PreparedStatement statement = null;

        try {
            //Membuat query Insert ke DB
            statement = DatabaseUtilities.getConnection().prepareStatement(
                    "INSERT INTO `mhsrmi` VALUES (?,?,?,?,?)");

            //mengisi symbol '?' pada query dengan data yang diambil dari interface remote object / API
            statement.setString(1, mahasiswa.getNpm());
            statement.setString(2, mahasiswa.getNama());
            statement.setString(3, mahasiswa.getJurusan());
            statement.setString(4, mahasiswa.getAlamat());
            statement.setString(5, mahasiswa.getPhone());

            //menjalankan perintah query diatas
            statement.executeUpdate();

            //Feedback ketika query berhasil dijalankan
            System.out.println("\t "+statement+"\n\t Permintaan berhasil diproses.\n"); //feedback untuk server
            mahasiswa.setLaporan("Data Berhasil Ditambahkan");  //feedback untuk client

            return mahasiswa;   //mengembalikan nilai yang dimasukkan ke variable mahasiswa
                                //dalam konteks ini adalah variable 'setLaporan' untuk feedback ke client.
        } catch (SQLException e) {

            //Feedback ketika query gagal dijalankan
            System.out.println("\t Permintaan gagal diproses.\n");
            mahasiswa.setLaporan("Data Gagal Ditambahkan");
            e.printStackTrace();
            return mahasiswa;

        } finally {
            if (statement != null) {
                try {
                    statement.close();
                } catch (Exception e) {
                }
            }
        }
    }

    //Method UPDATE untuk mengedit data pada Db
    @Override
    public Mahasiswa updateMahasiswa(Mahasiswa mahasiswa) throws RemoteException {
        System.out.println("[Server] Client memanggil fungsi update");

        //Membuat variable statement yang akan diisi dengan perintah query
        PreparedStatement statement = null;

        try {
            //membuat query UPDATE
            statement = DatabaseUtilities.getConnection().prepareStatement(
                    "UPDATE mhsrmi SET nama=?, jurusan=?, alamat=?, phone=? WHERE npm=?");

            //mengisi symbol '?' pada query dengan data yang diambil dari interface remote object / API
            statement.setString(1, mahasiswa.getNama());
            statement.setString(2, mahasiswa.getJurusan());
            statement.setString(3, mahasiswa.getAlamat());
            statement.setString(4, mahasiswa.getPhone());
            statement.setString(5, mahasiswa.getNpm());

            //menjalankan perintah query diatas
            statement.executeUpdate();

            //feedback ketika query berhasil dijalanlan
            System.out.println("\t "+statement+"\n\t Permintaan berhasil diproses.\n"); //feedback untuk server
            mahasiswa.setLaporan("Data Berhasil Diperabarui");  //feedback untuk client

            return mahasiswa;   //mengembalikan nilai yang dimasukkan ke variable mahasiswa
                                //dalam konteks ini adalah variable 'setLaporan' untuk feedback ke client.

        } catch (SQLException e) {

            //feedback ketika query gagal dijalankan
            System.out.println("\t Permintaan gagal diproses.\n");
            mahasiswa.setLaporan("Data Gagal Diperbarui");
            e.printStackTrace();    //keterangan gagal lebih lanjut

            return mahasiswa;   //mengembalikan nilai yang dimasukkan ke variable mahasiswa
                                //dalam konteks ini adalah variable 'setLaporan' untuk feedback ke client.

        } finally {
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }
```

```
    //Method DELETE untuk penghapusan data pada db
    @Override
    public Mahasiswa deleteMahasiswa(Mahasiswa mahasiswa) throws RemoteException {
        System.out.println("[Server] Client memanggil fungsi delete");

        //Membuat variable statement
        PreparedStatement statement = null;
        try {
            //mengisi statement dengan perintah query DELETE
            statement = DatabaseUtilities.getConnection().prepareStatement(
                    "DELETE FROM mhsrmi WHERE npm=?");

            //Mengisi simbol '?' pada query dengan Npm mahasiswa yang didapat dari interface remote object
            statement.setString(1, mahasiswa.getNpm());

            //Menjalankan query
            statement.executeUpdate();

            //feedback ketika query berhasil dijalankan
            System.out.println("\t "+statement+"\n\t Permintaan berhasil diproses.\n");
            mahasiswa.setLaporan("Data Berhasil Dihapus");
            return mahasiswa;

        } catch (SQLException e) {
            //feedback ketika query gagal dijalankan
            System.out.println("\t Permintaan gagal diproses.\n");
            mahasiswa.setLaporan("Data Gagal Dihapus!");
            e.printStackTrace(); //keterangan kegagalan lebih lanjut
            return mahasiswa;

        } finally {
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    //Mengambil data yang ada pada db
    @Override
    public Mahasiswa getMahasiswa(String npm) throws RemoteException {
        System.out.println("[Server] Client memanggil data pada database");

        //Membuat variable Statement
        PreparedStatement statement = null;
        try {
            //mengisi statement dengan perintah query untuk mendapatkan data berdasarkan NPM dari interface Remote object
            statement = DatabaseUtilities.getConnection().prepareStatement(
                    "SELECT * FROM mhsrmi WHERE Npm=? ");

            //Menjalankan perintah query
            ResultSet resultSet = statement.executeQuery();

            //membuat objek 'Mahasiswa'
            Mahasiswa mahasiswa = null;
            //Menyimpan data data yang didapat berdasarkan NPM pada varibel mahasiswa
            if (resultSet.next()) {
                mahasiswa = new Mahasiswa();

                //record variable mahasiswa
                mahasiswa.setNpm(npm);
                mahasiswa.setNama(resultSet.getString("nama"));
                mahasiswa.setJurusan(resultSet.getString("jurusan"));
                mahasiswa.setAlamat(resultSet.getString("alamat"));
                mahasiswa.setPhone(resultSet.getString("phone"));
            }

            //feedback ketika query berhasil dijalankan
            System.out.println("\t Data berhasi ditampilkan.\n");
            return mahasiswa;

        } catch (SQLException e) {
            //feedback ketika query gagal dijalankan
            e.printStackTrace(); //keterangan kegagalan
            return null;

        } finally {
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    //Menagambil serluruh data yang ada pada Db
    @Override
    public List<Mahasiswa> getAllMahasiswa() throws RemoteException {
        System.out.println("[Server] Client memanggil data pada database");

        //membuat variable statement
        Statement statement = null;
        try {

            //membuat koneksi dengan database
            statement = DatabaseUtilities.getConnection().createStatement();

            //menjalankan query untuk mendapatkan seluruh data yang ada pada table mhsrmi
            ResultSet resultSet = statement.executeQuery("SELECT * FROM mhsrmi ");

            //membuat variable tipe data Array
            List<Mahasiswa> list = new ArrayList<>();

            //infinity looping sampai seluruh data tersimpan
            while (resultSet.next()) {
                Mahasiswa mahasiswa = new Mahasiswa();

                //menyimpan setiap data pada variable mahasiswa
                mahasiswa.setNama(resultSet.getString("nama"));
                mahasiswa.setNpm(resultSet.getString("npm"));
                mahasiswa.setJurusan(resultSet.getString("jurusan"));
                mahasiswa.setAlamat(resultSet.getString("alamat"));
                mahasiswa.setPhone(resultSet.getString("phone"));
                list.add(mahasiswa);
            }
```

```
            //feedback ketika query berhasil dijalankan
            System.out.println("\t Data berhasil ditampilkan.\n");

            return list;

        } catch (SQLException e) {
            //feedback ketika query gagal dijalankan
            e.printStackTrace();
            return null;

        } finally {
            if (statement != null) {
                try {
                    statement.close();
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    //endofclass. Arry Febryan - 20160225200
}
```

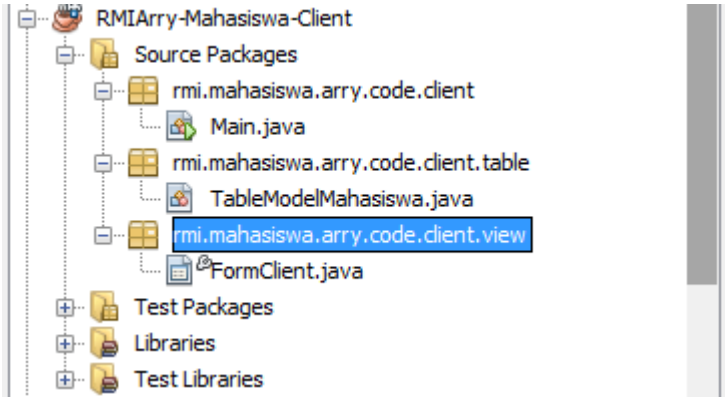
Class Main:

```
package rmi.mahasiswa.arry.code.server;

import java.rmi.AlreadyBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import rmi.mahasiswa.arry.code.server.service.MahasiswaServiceServer;
import rmi.mahasiswa.arry.code.server.utilities.DatabaseUtilities;

//Class Main Server
public class Main{

    public static void main(String[] args) throws RemoteException {

        System.out.println("=================================================");
        System.out.println("=====       ARRY FEBRYAN - RMI SERVER        =======");
        System.out.println("=================================================");

        try{
        //Port yang digunakan
        int port = 2302;

        //Mendaftarkan Remote Interface untuk server pada port 2302
        Registry server = LocateRegistry.createRegistry(port);
        //Membuat Objek MahasiswaServiceServer
        MahasiswaServiceServer userService = new MahasiswaServiceServer();

        //Mendaftarkan Obejek userService dengan nama "service" pada port RMI
        server.rebind("service", userService);

        //Feedback ketika Server RMI berhasil dijalankan
        System.out.println("[Server] Server sedang berjalan pada port "+port+"...\n");
        )catch(RemoteException e){
            //Feedback ketika Server RMI gagl dijalankan
            System.out.println("[Server] Server gagal dijalankan, baca keterangan lebih lanjut..\n\t "+e);
        }
    }
}
```

2.3. Application Client

Selanjutnya buatlah sebuah project untuk Application-Client dengan cara yang sama seperti membuat project untuk server.

Kemudian buatlah struktur package beserta classnya seperti dibawah ini:



Untuk FormClient kita buat dengan Jframe Form sebagai tatap muka kepada user client.

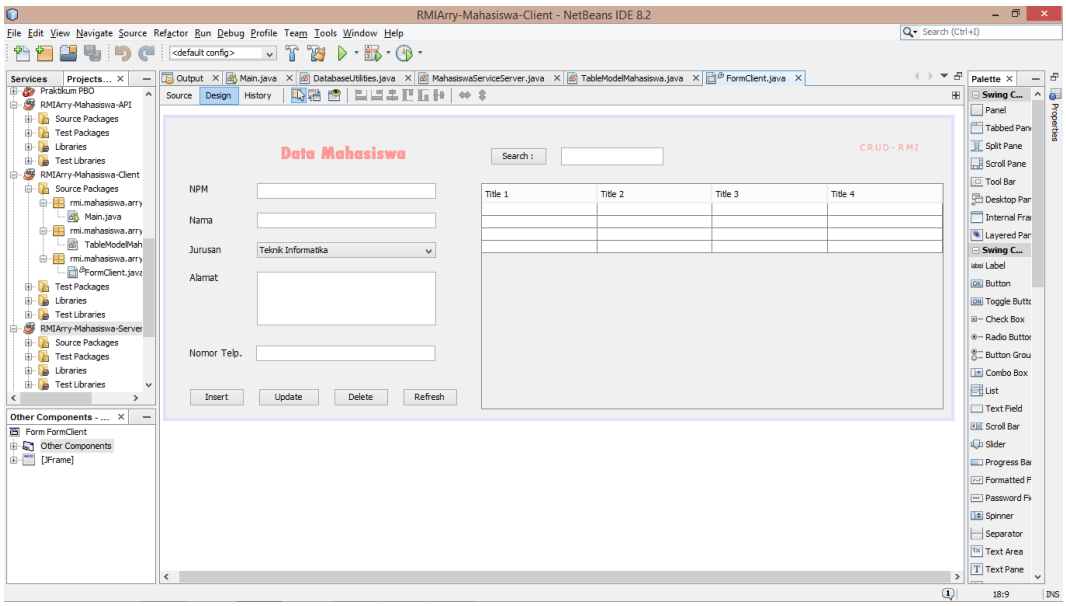
Berikutnya import project Application-API kedalam project client dengan cara yang sama seperti sebelumnya.

Selanjutnya isilah class-class yang ada pada project Client dengan kode program sebagai berikut:

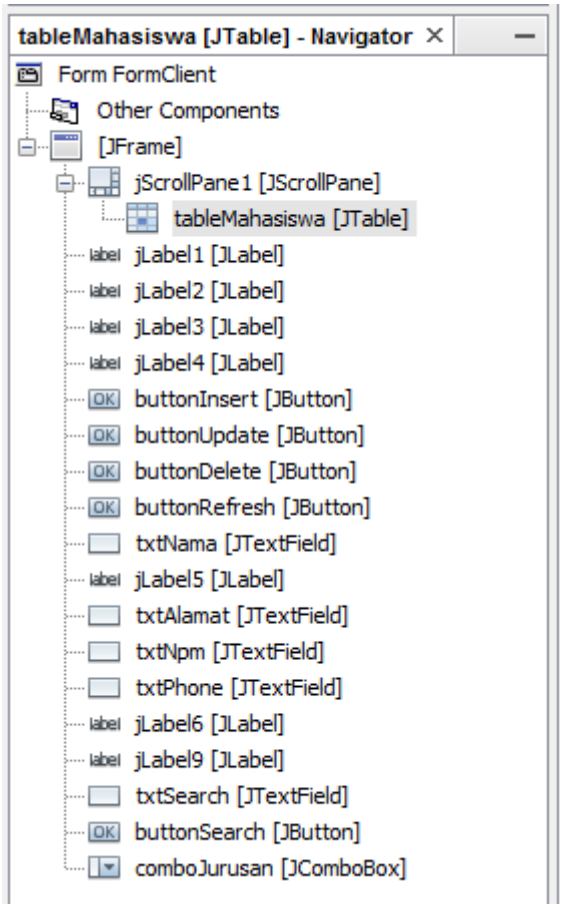
TableModelMahasiswa:

```
package rmi.mahasiswa.arry.code.client.table;

import java.util.ArrayList;
import java.util.List;
import javax.swing.table.AbstractTableModel;
import rmi.mahasiswa.code.entity.Mahasiswa;

//Class untuk menampilkan data yang ada pada database ke table JFrame
public class TableModelMahasiswa extends AbstractTableModel {
    private List<Mahasiswa> list = new ArrayList<Mahasiswa>();

    //untuk mendapatkan data dan menampilkannya ke dalam table
    public Mahasiswa get(int row) {
        return list.get(row);
    }

    //untuk merubah data yang ada pada table
    public void setData(List<Mahasiswa> list) {
        this.list = list;
        fireTableDataChanged();
    }

    //menentukan jumlah baris pada table berdasarkan data yang ada
    @Override
    public int getRowCount() {
        return list.size();
    }

    //menentukan banyak jumlah kolom pada table
    @Override
    public int getColumnCount() {
        return 5; //5 Kolom
    }

    //Mengambil dan menampilkan data ke dalam kolom yang ditentukan
    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        switch (columnIndex) {
            case 0:
                return list.get(rowIndex).getNpm();
            case 1:
               return list.get(rowIndex).getNama();
            case 2:
                return list.get(rowIndex).getJurusan();
            case 3:
                return list.get(rowIndex).getAlamat();
            case 4:
                return list.get(rowIndex).getPhone();

            default:
                return null;
        }
    }

    //Menentukan judul setiap kolom
    @Override
    public String getColumnName(int column) {
        switch (column) {
            case 0:
                return "Npm";
            case 1:
                return "Nama";
            case 2:
                return "Jurusan";
            case 3:
                return "Alamat";
            case 4:
                return "Phone";

            default:
                return null;
        }
    }

}
```

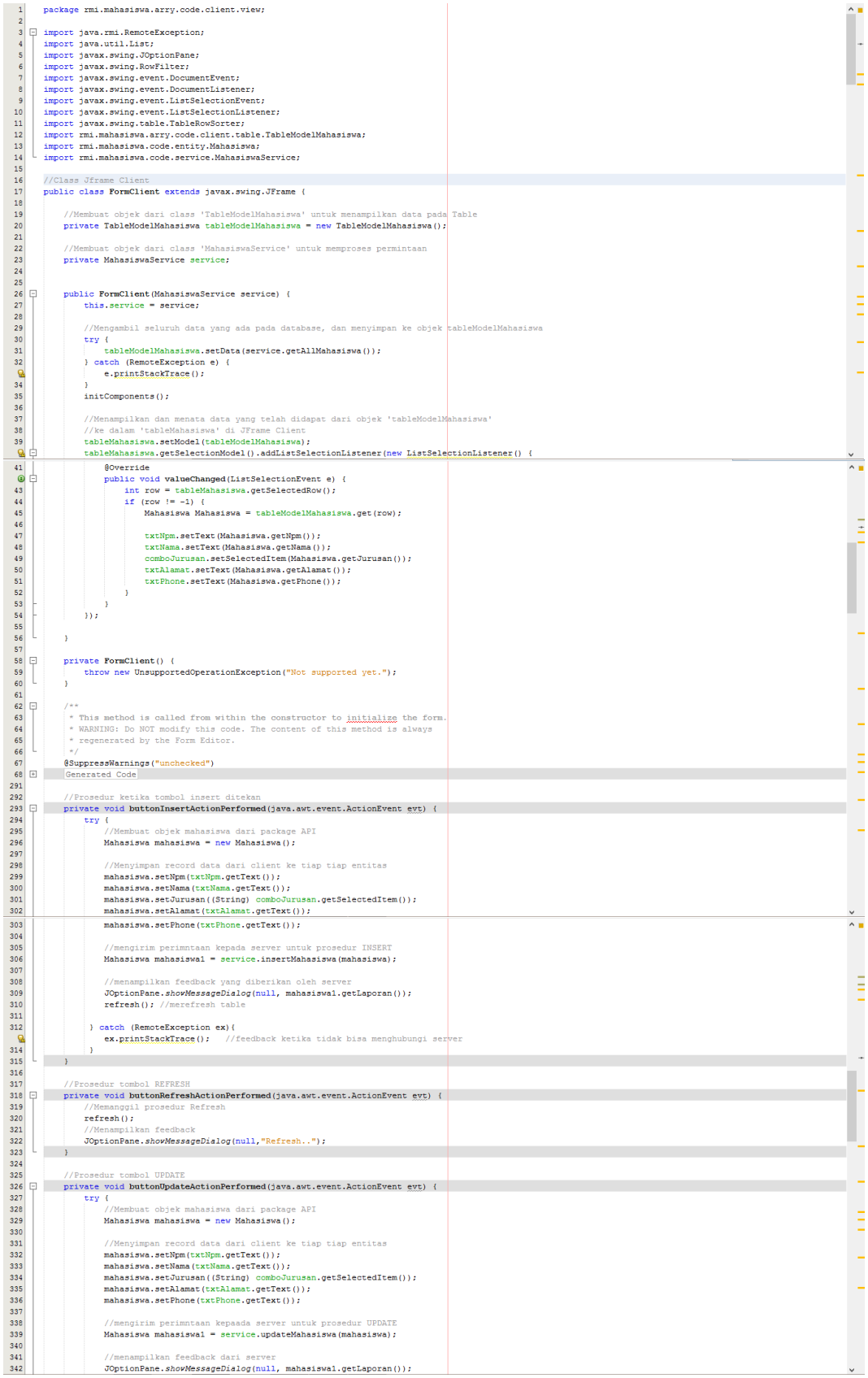
Buatlah desain untuk antarmuka client pada **Jframe FormClient.**



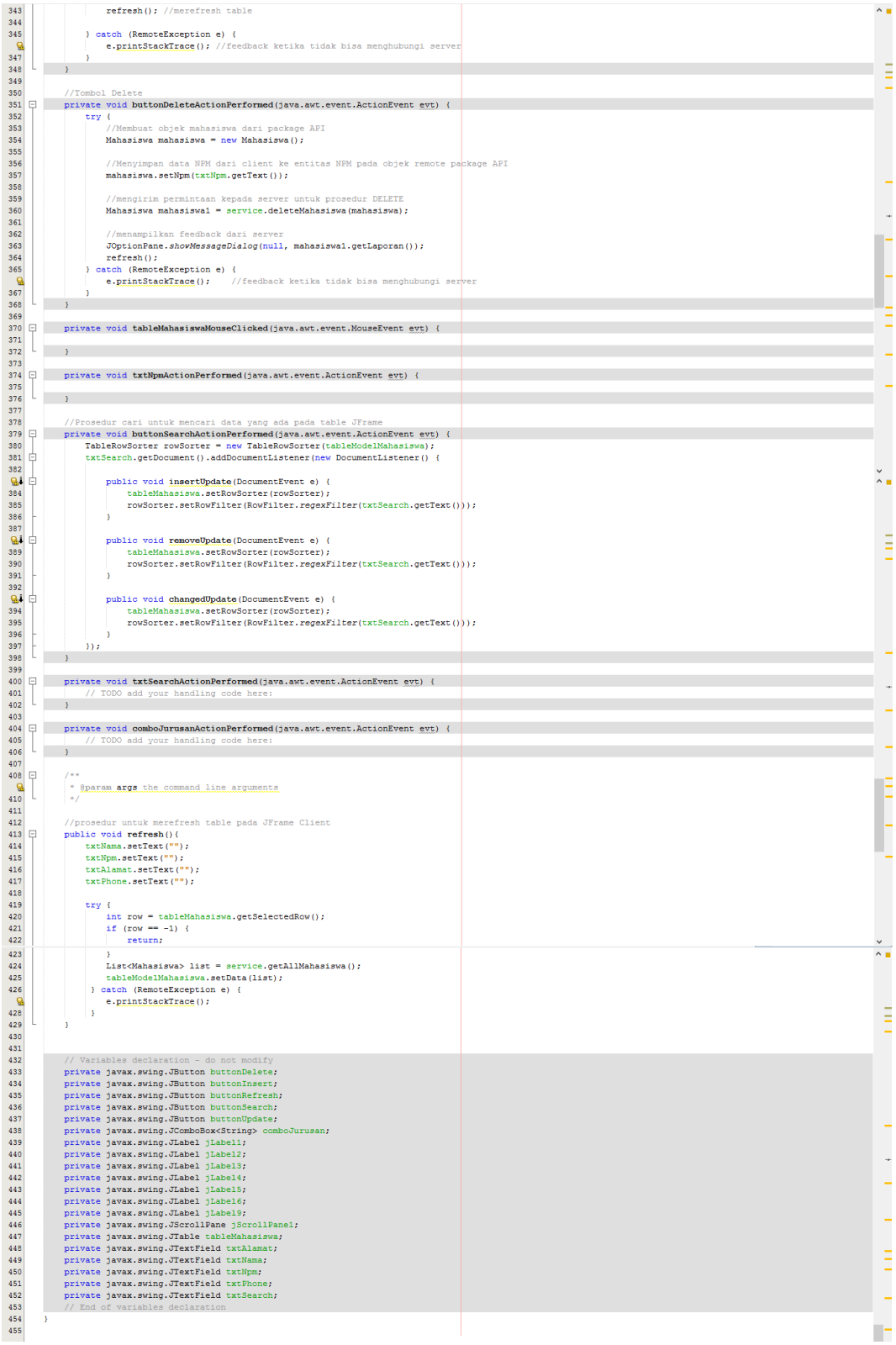
Dengan penamaan setiap variable sebagai berikut:



Selanjutnya tulislah kode program untuk **Jframe FormClient** seperti dibawah ini:

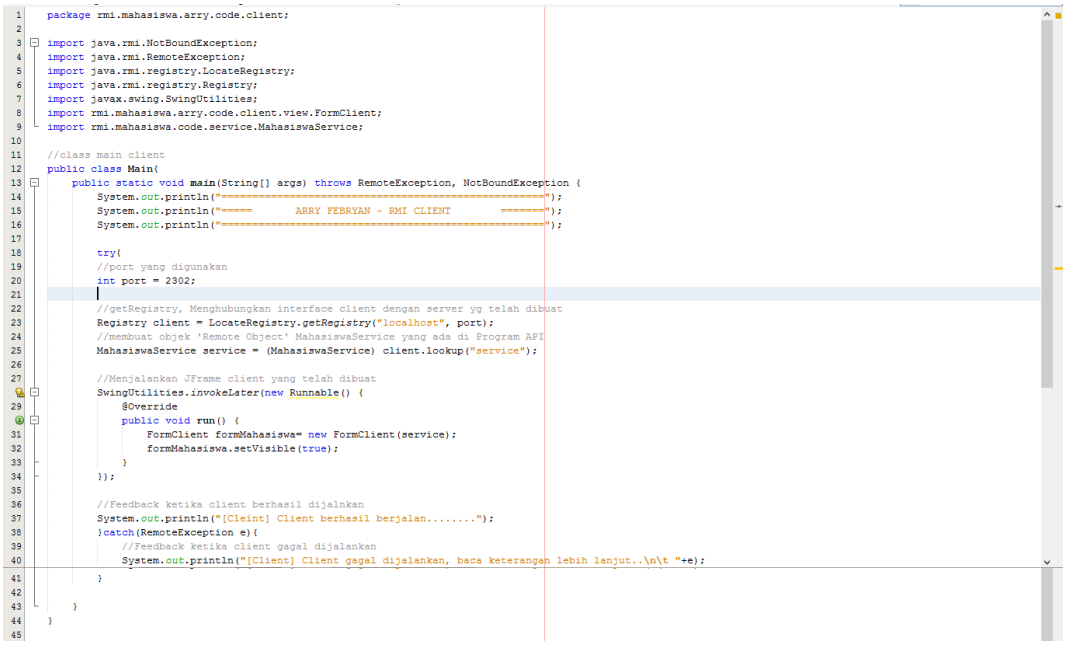


```java
package rmi.mahasiswa.arry.code.client.view;

import java.rmi.RemoteException;
import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.RowFilter;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.TableRowSorter;
import rmi.mahasiswa.arry.code.client.table.TableModelMahasiswa;
import rmi.mahasiswa.code.entity.Mahasiswa;
import rmi.mahasiswa.code.service.MahasiswaService;

//Class Jframe Client
public class FormClient extends javax.swing.JFrame {

    //Membuat objek dari class 'TableModelMahasiswa' untuk menampilkan data pada Table
    private TableModelMahasiswa tableModelMahasiswa = new TableModelMahasiswa();

    //Membuat objek dari class 'MahasiswaService' untuk memproses permintaan
    private MahasiswaService service;


    public FormClient(MahasiswaService service) {
        this.service = service;

        //Mengambil seluruh data yang ada pada database, dan menyimpan ke objek tableModelMahasiswa
        try {
            tableModelMahasiswa.setData(service.getAllMahasiswa());
        } catch (RemoteException e) {
            e.printStackTrace();
        }
        initComponents();

        //Menampilkan dan menata data yang telah didapat dari objek 'tableModelMahasiswa'
        //ke dalam 'tableMahasiswa' di JFrame Client
        tableMahasiswa.setModel(tableModelMahasiswa);
        tableMahasiswa.getSelectionModel().addListSelectionListener(new ListSelectionListener() {
            @Override
            public void valueChanged(ListSelectionEvent e) {
                int row = tableMahasiswa.getSelectedRow();
                if (row != -1) {
                    Mahasiswa Mahasiswa = tableModelMahasiswa.get(row);

                    txtNpm.setText(Mahasiswa.getNpm());
                    txtNama.setText(Mahasiswa.getNama());
                    comboJurusan.setSelectedItem(Mahasiswa.getJurusan());
                    txtAlamat.setText(Mahasiswa.getAlamat());
                    txtPhone.setText(Mahasiswa.getPhone());
                }
            }
        });

    }

    private FormClient() {
        throw new UnsupportedOperationException("Not supported yet.");
    }

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

    //Prosedur ketika tombol insert ditekan
    private void buttonInsertActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            //Membuat objek mahasiswa dari package API
            Mahasiswa mahasiswa = new Mahasiswa();

            //Menyimpan record data dari client ke tiap tiap entitas
            mahasiswa.setNpm(txtNpm.getText());
            mahasiswa.setNama(txtNama.getText());
            mahasiswa.setJurusan((String) comboJurusan.getSelectedItem());
            mahasiswa.setAlamat(txtAlamat.getText());
            mahasiswa.setPhone(txtPhone.getText());

            //mengirim perimntaan kepada server untuk prosedur INSERT
            Mahasiswa mahasiswa1 = service.insertMahasiswa(mahasiswa);

            //menampilkan feedback yang diberikan oleh server
            JOptionPane.showMessageDialog(null, mahasiswa1.getLaporan());
            refresh(); //merefresh table

        } catch (RemoteException ex){
            ex.printStackTrace();   //feedback ketika tidak bisa menghubungi server
        }
    }

    //Prosedur tombol REFRESH
    private void buttonRefreshActionPerformed(java.awt.event.ActionEvent evt) {
        //Memanggil prosedur Refresh
        refresh();
        //Menampilkan feedback
        JOptionPane.showMessageDialog(null,"Refresh..");
    }

    //Prosedur tombol UPDATE
    private void buttonUpdateActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            //Membuat objek mahasiswa dari package API
            Mahasiswa mahasiswa = new Mahasiswa();

            //Menyimpan record data dari client ke tiap tiap entitas
            mahasiswa.setNpm(txtNpm.getText());
            mahasiswa.setNama(txtNama.getText());
            mahasiswa.setJurusan((String) comboJurusan.getSelectedItem());
            mahasiswa.setAlamat(txtAlamat.getText());
            mahasiswa.setPhone(txtPhone.getText());

            //mengirim perimntaan kepada server untuk prosedur UPDATE
            Mahasiswa mahasiswa1 = service.updateMahasiswa(mahasiswa);

            //menampilkan feedback dari server
            JOptionPane.showMessageDialog(null, mahasiswa1.getLaporan());
```



```
            refresh(); //merefresh table

        } catch (RemoteException e) {
            e.printStackTrace(); //feedback ketika tidak bisa menghubungi server
        }
    }

    //Tombol Delete
    private void buttonDeleteActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            //Membuat objek mahasiswa dari package API
            Mahasiswa mahasiswa = new Mahasiswa();

            //Menyimpan data NPM dari client ke entitas NPM pada objek remote package API
            mahasiswa.setNpm(txtNpm.getText());

            //mengirim permintaan kepada server untuk prosedur DELETE
            Mahasiswa mahasiswa1 = service.deleteMahasiswa(mahasiswa);

            //menampilkan feedback dari server
            JOptionPane.showMessageDialog(null, mahasiswa1.getLaporan());
            refresh();
        } catch (RemoteException e) {
            e.printStackTrace();    //feedback ketika tidak bisa menghubungi server
        }
    }

    private void tableMahasiswaMouseClicked(java.awt.event.MouseEvent evt) {

    }

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

    }

    //Prosedur cari untuk mencari data yang ada pada table JFrame
    private void buttonSearchActionPerformed(java.awt.event.ActionEvent evt) {
        TableRowSorter rowSorter = new TableRowSorter(tableModelMahasiswa);
        txtSearch.getDocument().addDocumentListener(new DocumentListener() {

            public void insertUpdate(DocumentEvent e) {
                tableMahasiswa.setRowSorter(rowSorter);
                rowSorter.setRowFilter(RowFilter.regexFilter(txtSearch.getText()));
            }

            public void removeUpdate(DocumentEvent e) {
                tableMahasiswa.setRowSorter(rowSorter);
                rowSorter.setRowFilter(RowFilter.regexFilter(txtSearch.getText()));
            }

            public void changedUpdate(DocumentEvent e) {
                tableMahasiswa.setRowSorter(rowSorter);
                rowSorter.setRowFilter(RowFilter.regexFilter(txtSearch.getText()));
            }
        });
    }

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

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

    /**
     * @param args the command line arguments
     */

    //prosedur untuk merefresh table pada JFrame Client
    public void refresh(){
        txtNama.setText("");
        txtNpm.setText("");
        txtAlamat.setText("");
        txtPhone.setText("");

        try {
            int row = tableMahasiswa.getSelectedRow();
            if (row == -1) {
                return;
            }
            List<Mahasiswa> list = service.getAllMahasiswa();
            tableModelMahasiswa.setData(list);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }


    // Variables declaration - do not modify
    private javax.swing.JButton buttonDelete;
    private javax.swing.JButton buttonInsert;
    private javax.swing.JButton buttonRefresh;
    private javax.swing.JButton buttonSearch;
    private javax.swing.JButton buttonUpdate;
    private javax.swing.JComboBox<String> comboJurusan;
    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 jLabel9;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable tableMahasiswa;
    private javax.swing.JTextField txtAlamat;
    private javax.swing.JTextField txtNama;
    private javax.swing.JTextField txtNpm;
    private javax.swing.JTextField txtPhone;
    private javax.swing.JTextField txtSearch;
    // End of variables declaration
}
```

Class Main:

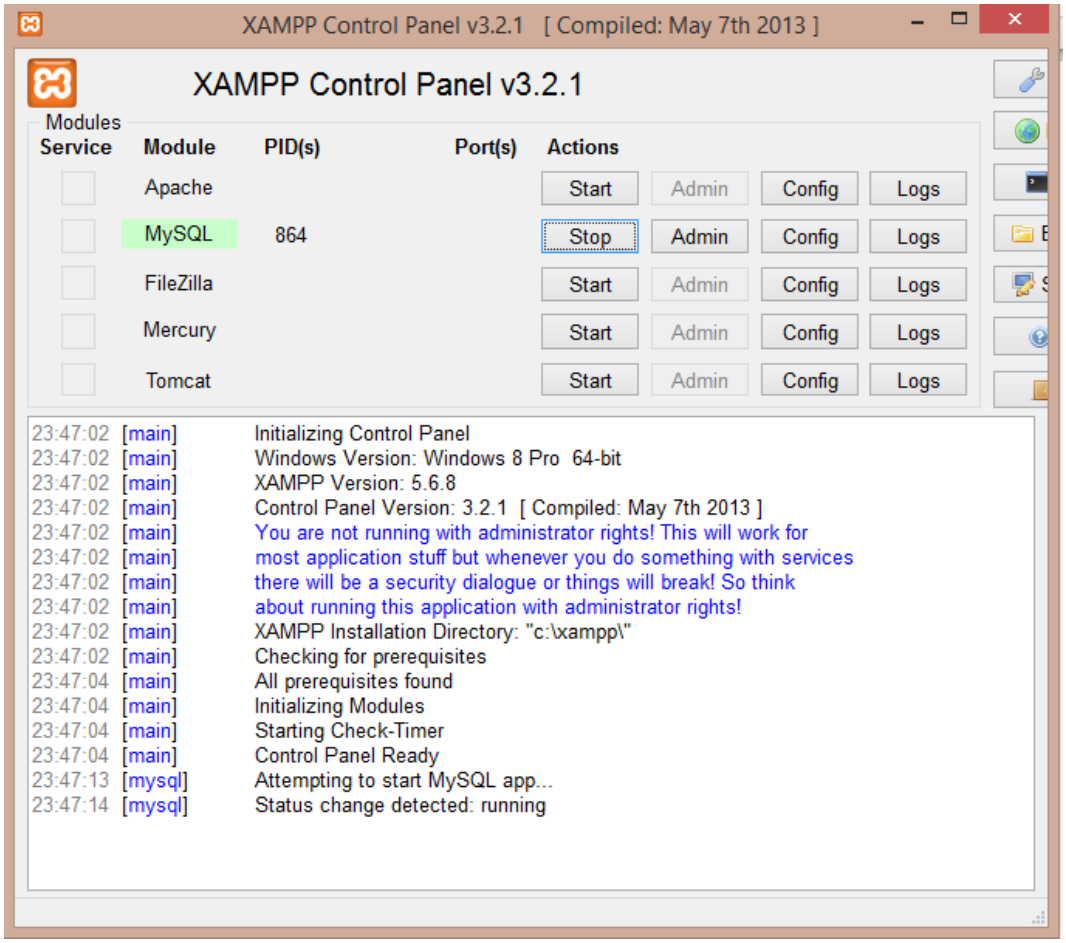


```
package rmi.mahasiswa.arry.code.client;

import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import javax.swing.SwingUtilities;
import rmi.mahasiswa.arry.code.client.view.FormClient;
import rmi.mahasiswa.code.service.MahasiswaService;

//class main client
public class Main{
    public static void main(String[] args) throws RemoteException, NotBoundException {
        System.out.println("================================================");
        System.out.println("======       ARRY FEBRYAN - RMI CLIENT       =======");
        System.out.println("================================================");

        try{
        //port yang digunakan
        int port = 2302;

        //getRegistry, Menghubungkan interface client dengan server yg telah dibuat
        Registry client = LocateRegistry.getRegistry("localhost", port);
        //membuat objek 'Remote Object' MahasiswaService yang ada di Program API
        MahasiswaService service = (MahasiswaService) client.lookup("service");

        //Menjalankan JFrame client yang telah dibuat
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                FormClient formMahasiswa= new FormClient(service);
                formMahasiswa.setVisible(true);
            }
        });

        //Feedback ketika client berhasil dijalnkan
        System.out.println("[Cleint] Client berhasil berjalan.......");
        }catch(RemoteException e){
            //Feedback ketika client gagal dijalankan
            System.out.println("[Client] Client gagal dijalankan, baca keterangan lebih lanjut..\n\t "+e);
        }

    }
}
```

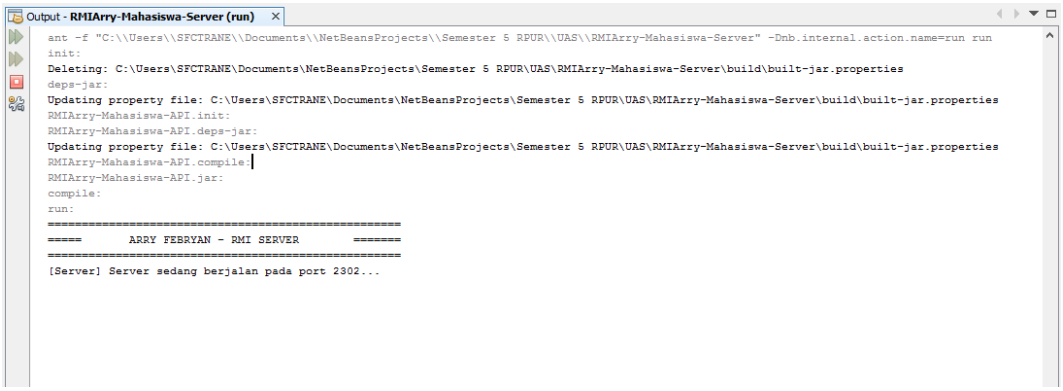
Selesai

D. Testing Aplikasi

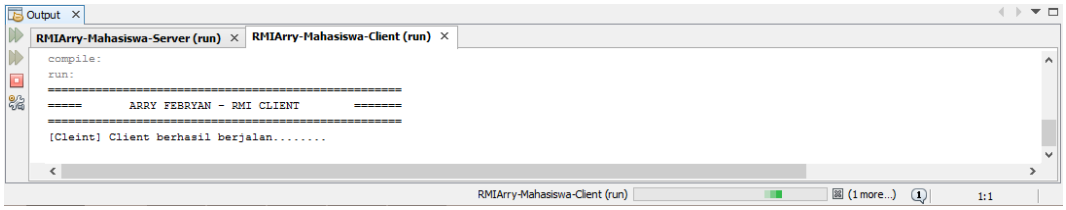
Langkah pertama, pastikan modul MySQL pada XAMPP telah berjalan.

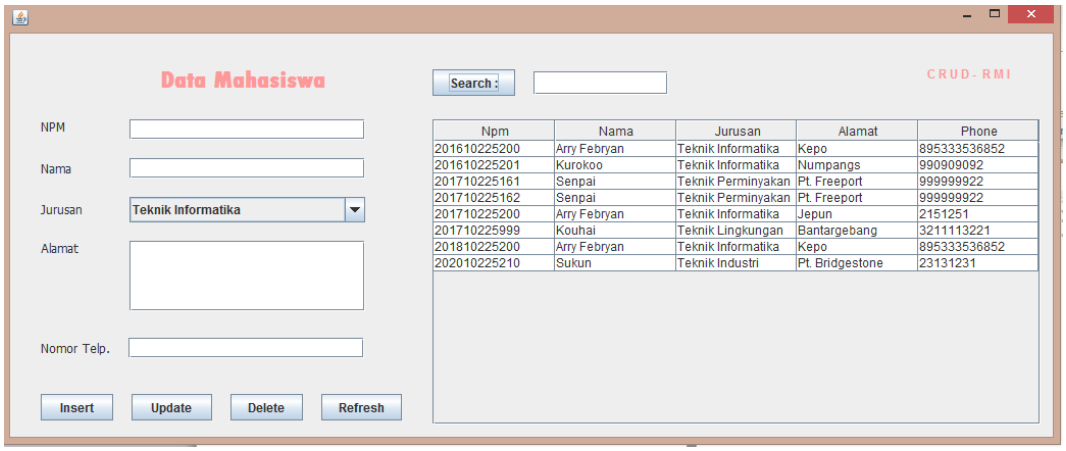


Kemudian jalankan program server terlebih dahulu.

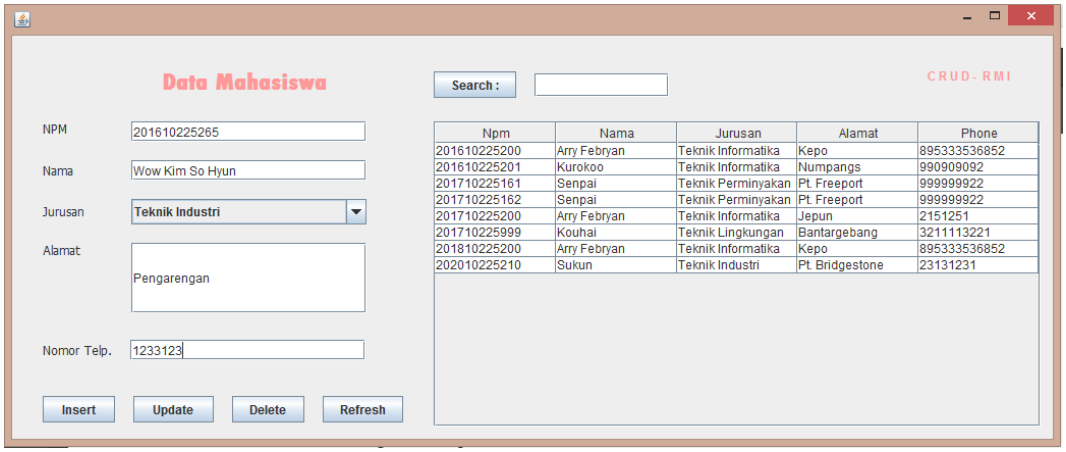


Selanjutnya jalankan program client.

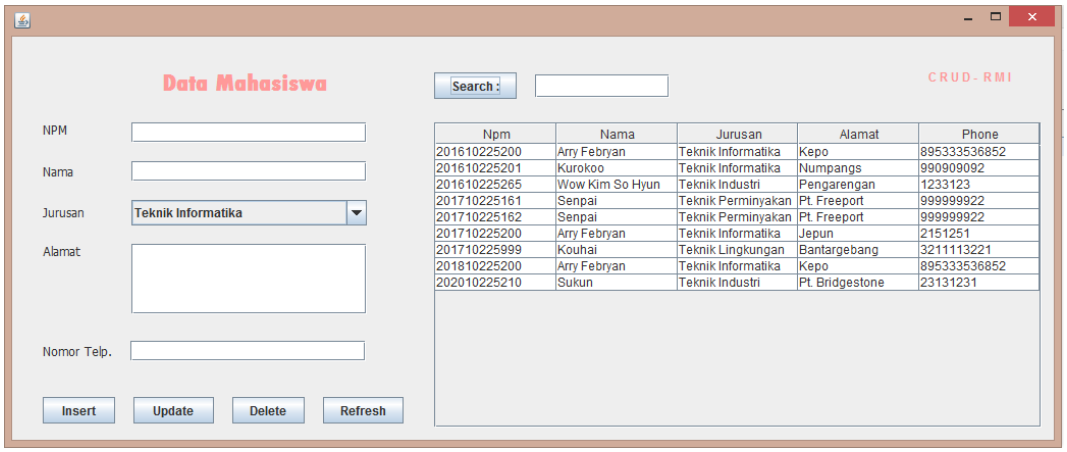




Test input mahasiswa baru:



Hasil:



E. Referensi

Maulana Achmad, “Membuat Aplikasi Desktop Client Server Sederhana dengan Java RMI” 2016.

Ratnasari Nur Rohmah, “Client/Server dengan Java Remote Method Invocation (Java RMI),” *J. Tek. Elektro Dan Komput. Emit.*, vol. 3, p. 5, 2003.

Purnomo Rakhmat, “Pemrograman Jaringan Remote Method Invocation (RMI)”.

http://java.sun.com/javase/6/docs/technotes/guides/rmi/index.html