

Java NetBeans

MODUL PRAKTIKUM
MIRZA YOGY KURNIAWAN

CONTENTS

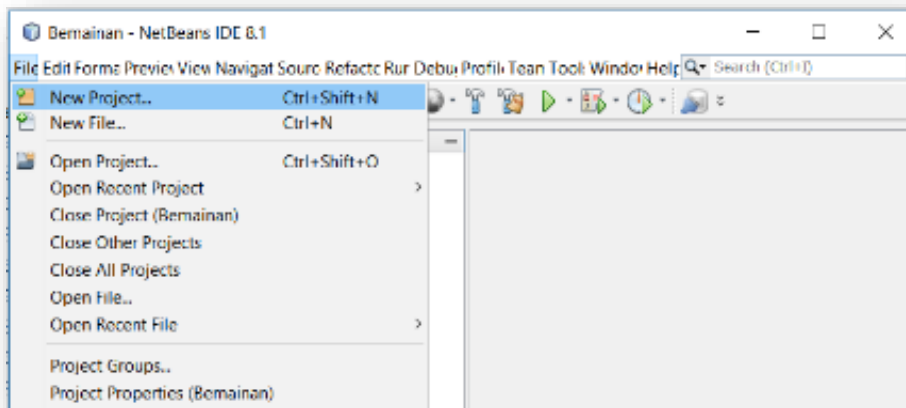
Java Basic dengan NetBeans.....	4
Hello World	4
Class.....	7
Constructor	14
Latihan 1.1.....	18
Overloading	18
Latihan 1.2	19
Inheritance.....	19
Overriding.....	20
MySQL and Text Based Frame	21
Database.....	21
Membuat Project Baru	21
PenerbitTampilFrame	26
Tampil Frame	26
Tampil Data.....	28
Listener	30
PenerbitTambahFrame.....	33
Tampil Frame	33
Listener	35
MySQL and Visual Based Frame	38
Database.....	38
PetugasTampilFrame.....	39
Tampil Frame	39
Tampil Data.....	43
Listener	45
PetugasTambahFrame.....	47
Tampil Frame	47

BAB 1

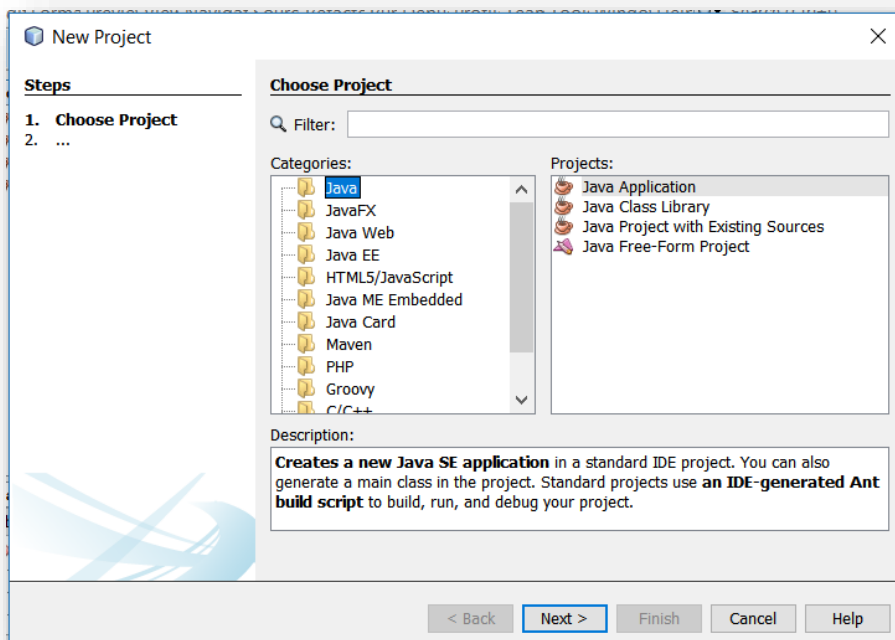
JAVA BASIC DENGAN NETBEANS

1.1 HELLO WORLD

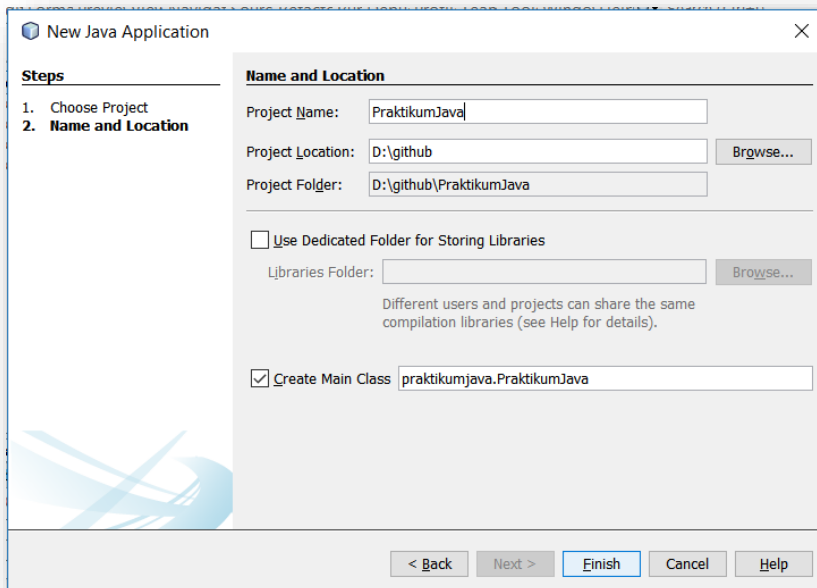
Memulai belajar bahasa pemrograman rasanya tidak afdol kalau tidak dimulai dengan membuat “Hello World”. Maka kita mulai dengan membuat project NetBeans baru terlebih dahulu



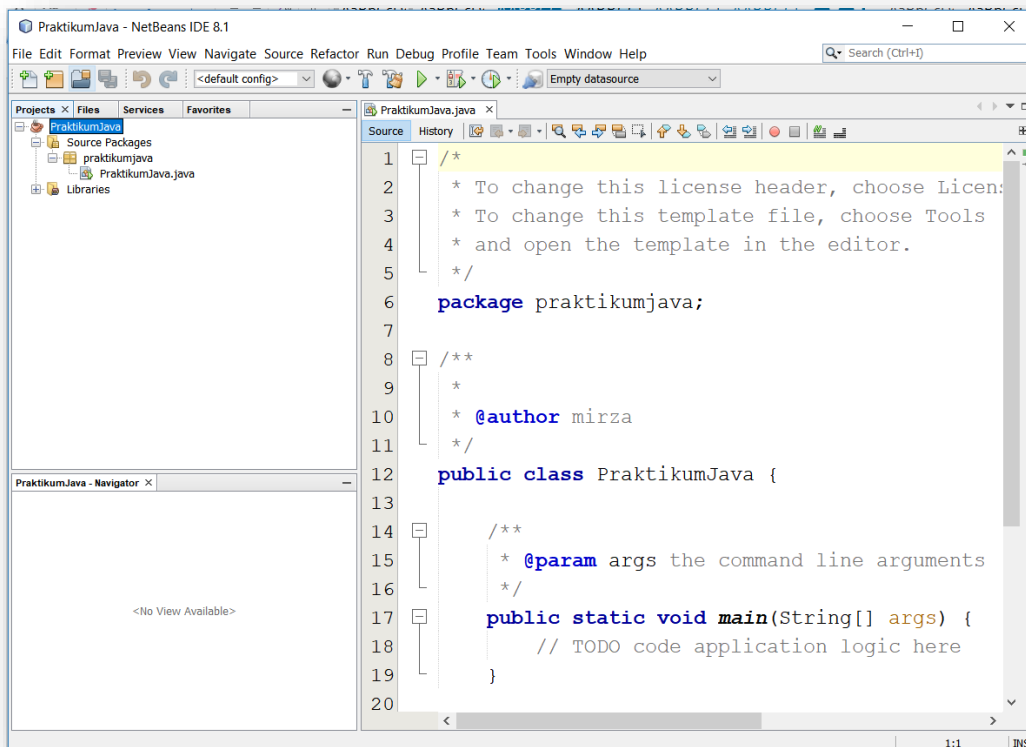
Pilih project Java – Java Application



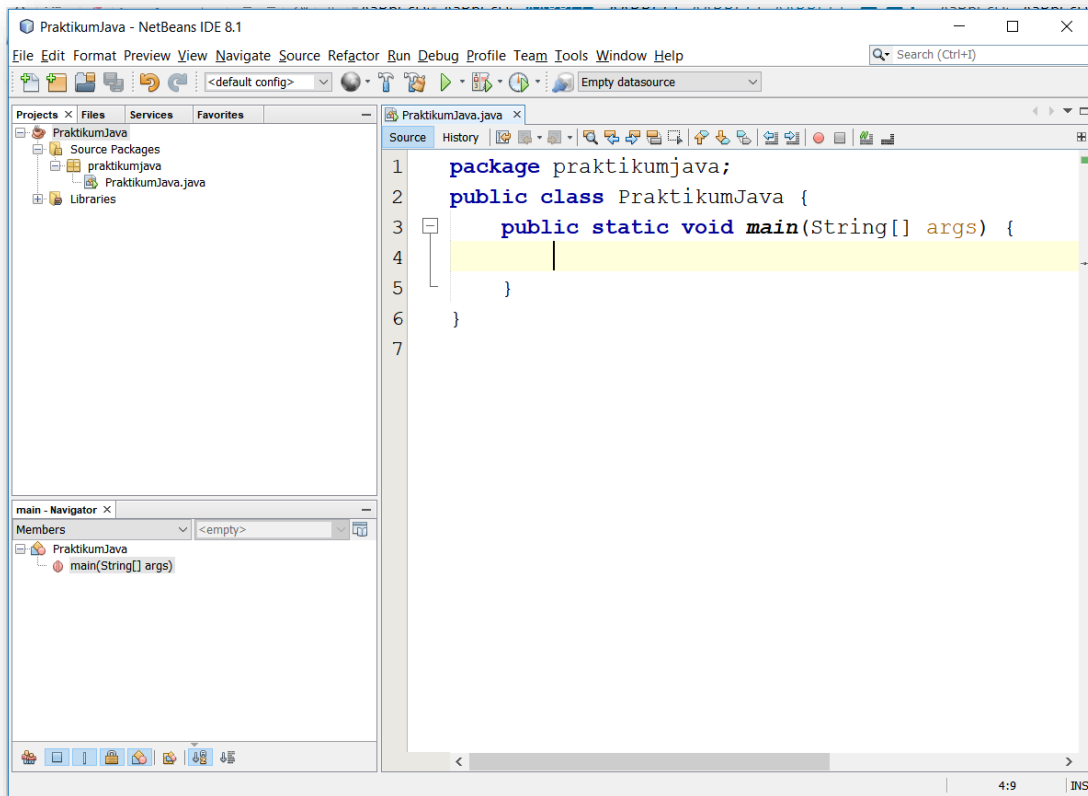
Isikan **Project Name**, pilih lokasi project yang diinginkan, biarkan sisinya klik **Finish**



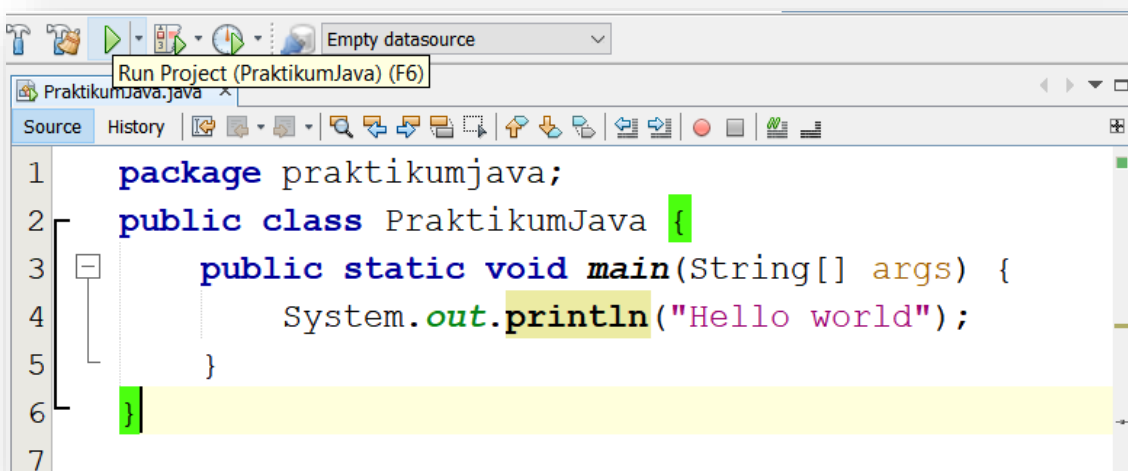
Berikut tampilan awal Project pada **class utama**



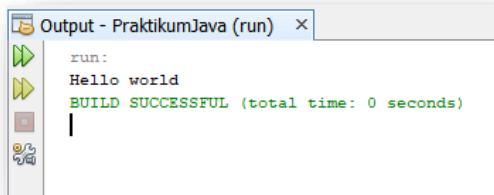
Komentar yang kurang begitu penting bisa dihapus sehingga tampil seperti berikut



Tuliskan perintah seperti berikut



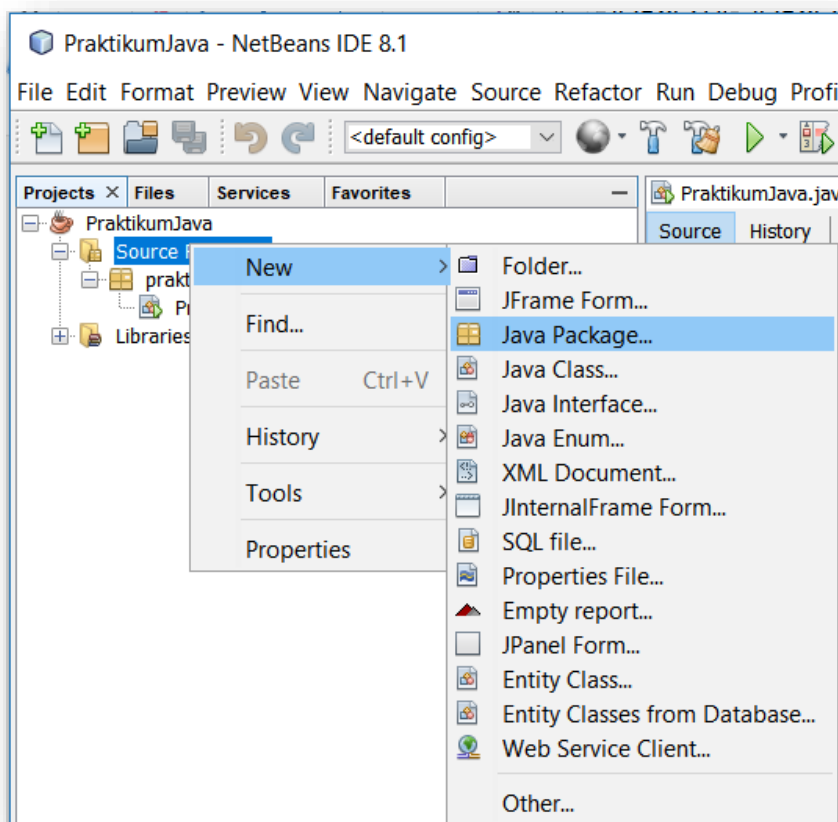
Jalankan project dengan menekan tombol **F6**, maka didapatkan pada bagian **Output** tulisan sebagai berikut



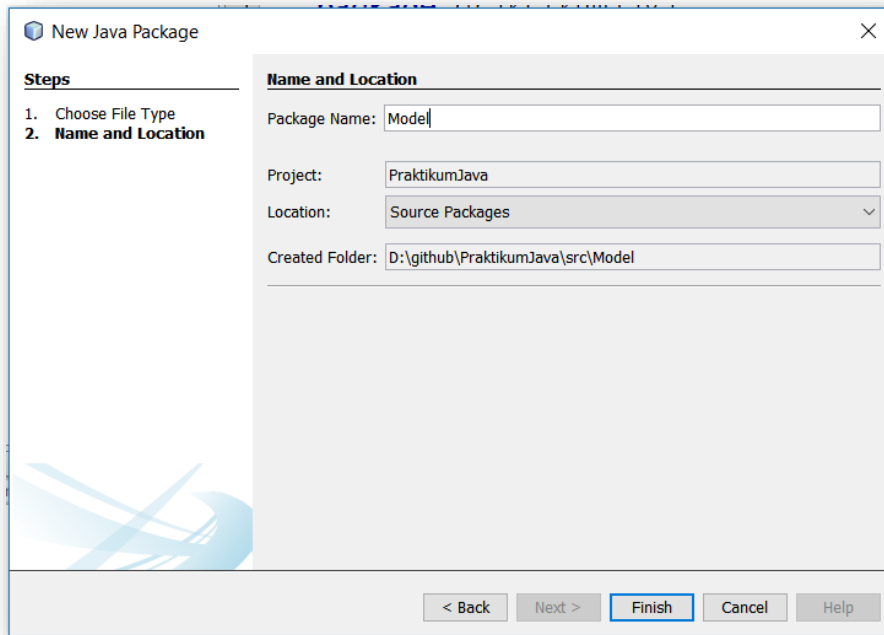
1.2 CLASS

Perbedaan **Object Oriented Programming (OOP/PBO)** dengan paradigma lain adalah penggunaan object yang diciptakan dari class.

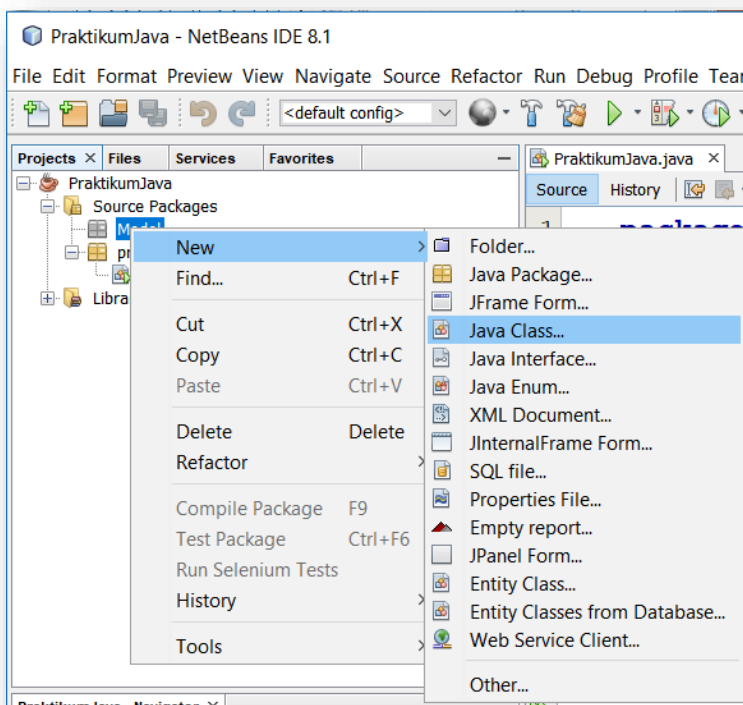
Pada praktek ini, untuk membuat class, terlebih dahulu kita buat **package** atau foldernya. Klik kanan pada **Source Packages** kemudian **New – Java Package**



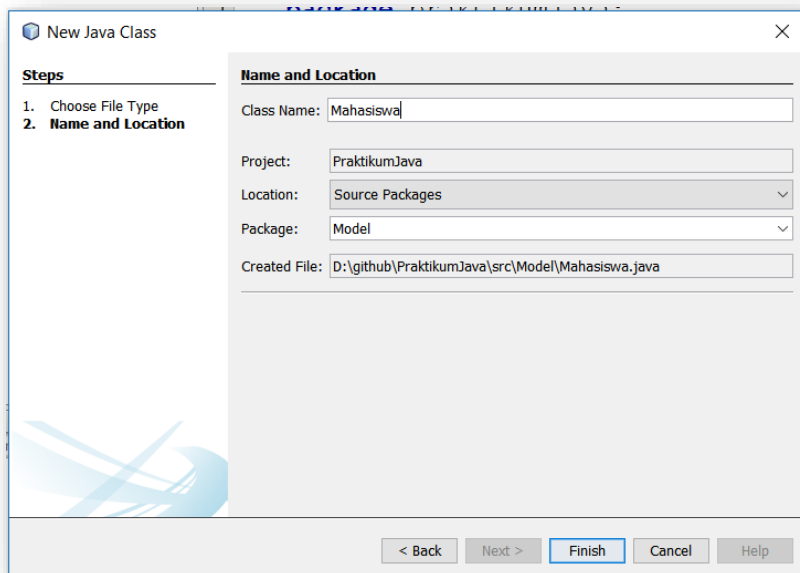
Isikan pada **Package Name** dengan nama **Model**, kemudian klik **Finish**



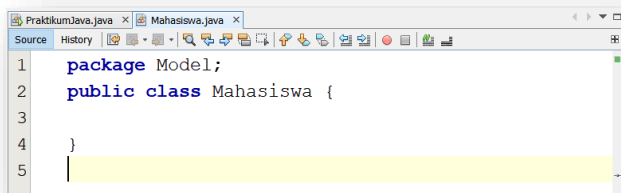
Buat class baru dengan klik kanan pada package **Model** – **New** – **Java Class**..



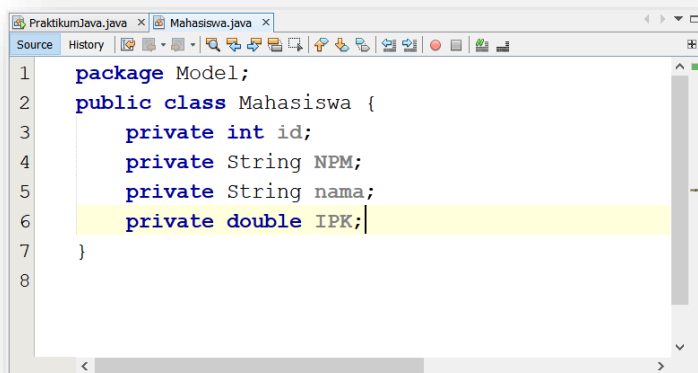
Isikan **Class Name** dengan **Mahasiswa** kemudian klik **Finish**



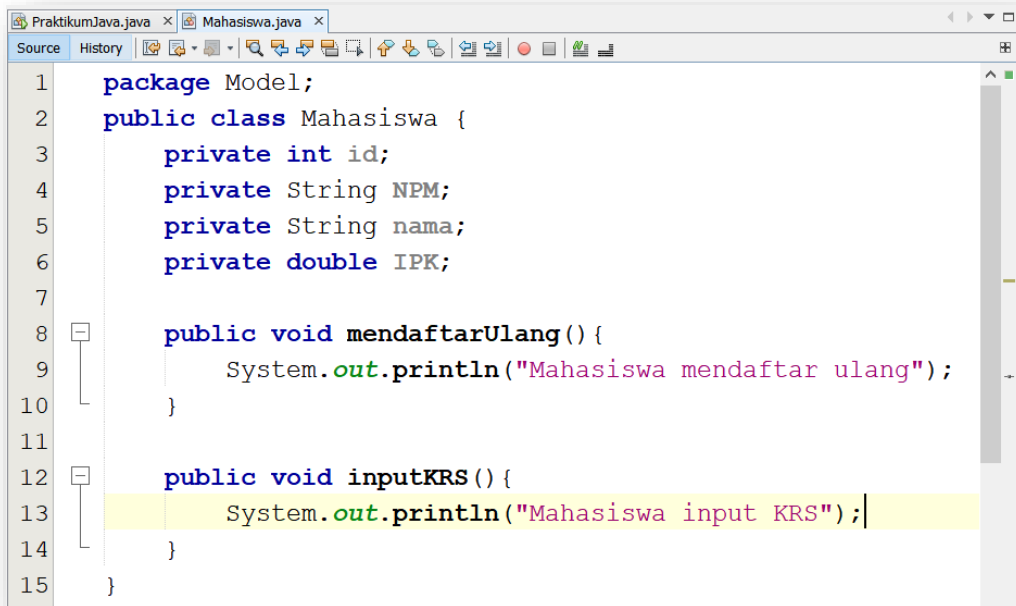
Pada class **Mahasiswa** bersihkan komentar yang kurang begitu penting



Kemudian isikan **atribut** seperti berikut

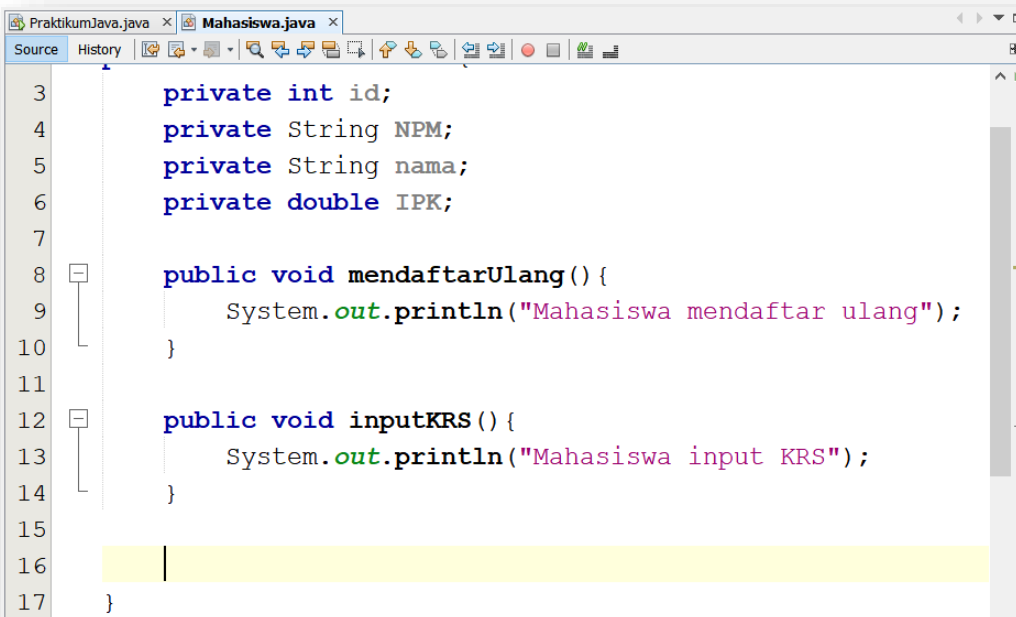


Buat **method** berikut



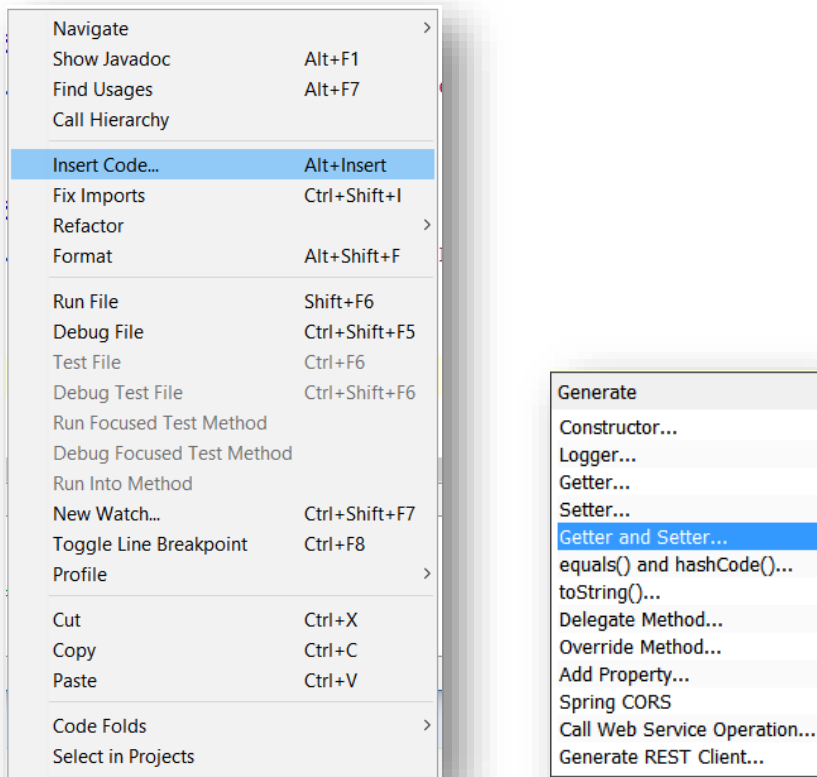
```
1 package Model;
2 public class Mahasiswa {
3     private int id;
4     private String NPM;
5     private String nama;
6     private double IPK;
7
8     public void mendaftarkanUlang() {
9         System.out.println("Mahasiswa mendaftarkan ulang");
10    }
11
12    public void inputKRS() {
13        System.out.println("Mahasiswa input KRS");
14    }
15 }
```

Beri jarak dibawah method terakhir

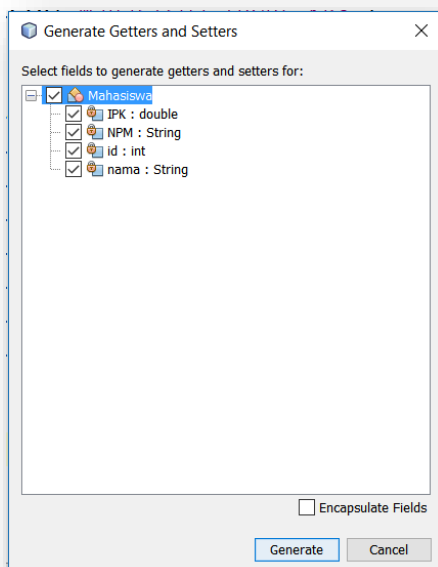


```
3     private int id;
4     private String NPM;
5     private String nama;
6     private double IPK;
7
8     public void mendaftarkanUlang() {
9         System.out.println("Mahasiswa mendaftarkan ulang");
10    }
11
12    public void inputKRS() {
13        System.out.println("Mahasiswa input KRS");
14    }
15
16
17 }
```

Klik kanan pilih **Insert Code...** kemudian pilih **Getter and Setter...**



Centang seperti berikut kemudian klik **Generate**



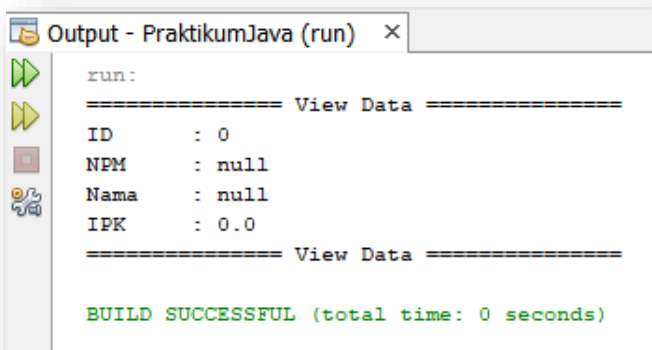
Tambahkan method **viewData()**

```
public void viewData() {
    System.out.println("===== View Data =====");
    System.out.println("ID \t: "+id);
    System.out.println("NPM \t: "+NPM);
    System.out.println("Nama \t: "+nama);
    System.out.println("IPK \t: "+IPK);
    System.out.println("===== View Data =====");
    System.out.println("");
}
```

Kembali ke **class utama** dimana dituliskan **Hello world**, kita ganti isinya dengan **instance** dari class **Mahasiswa** dengan nama **mahasiswa**. Jangan lupa untuk import terlebih dahulu class **Mahasiswa** yang ada di package **Model**.

```
1  package praktikumjava;
2
3  import Model.Mahasiswa;
4
5  public class PraktikumJava {
6      public static void main(String[] args) {
7          Mahasiswa mahasiswa = new Mahasiswa();
8          mahasiswa.viewData();
9      }
10 }
```

Simpan dan jalankan project dengan menekan tombol **F6**



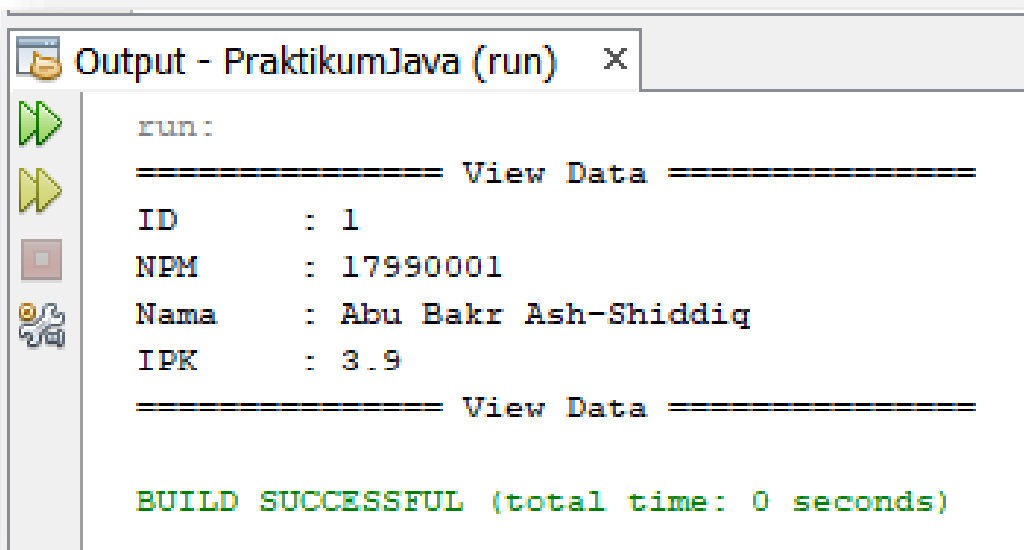
```
Output - PraktikumJava (run) x
run:
===== View Data =====
ID      : 0
NPM     : null
Nama    : null
IPK     : 0.0
===== View Data =====

BUILD SUCCESSFUL (total time: 0 seconds)
```

Nilai yang ditampilkan masih 0 dan null, karena memang belum kita isi nilainya. Kita isikan dengan method setter yang sudah di generate di class mahasiswa. Method setter dijalankan pada **class utama**

```
5 public class PraktikumJava {  
6     public static void main(String[] args) {  
7         Mahasiswa mahasiswa = new Mahasiswa();  
8  
9         mahasiswa.setId(1);  
10        mahasiswa.setNPM("17990001");  
11        mahasiswa.setNama("Abu Bakr Ash-Shiddiq");  
12        mahasiswa.setIPK(3.9);  
13  
14        mahasiswa.viewData();  
15    }  
16 }
```

Simpan dan jalankan project dengan menekan tombol F6



```
Output - PraktikumJava (run) X  
run:  
==== View Data =====  
ID      : 1  
NPM     : 17990001  
Nama    : Abu Bakr Ash-Shiddiq  
IPK     : 3.9  
==== View Data =====  
  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pengisian nilai berhasil. Selain dengan menggunakan method setter, nilai juga bisa ditentukan pada method **constructor** yang jalan ketika object di instance.

1.3 CONSTRUCTOR

Constructor adalah method yang dieksekusi ketika object di instance, penamaan constructor **tidak** menggunakan void ataupun tipe return lainnya, harus sama dengan nama class, dan tidak memiliki nilai return. Pada class Mahasiswa maka penamaan constructornya minimal "Mahasiswa()"

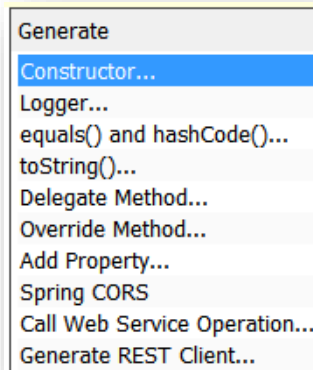
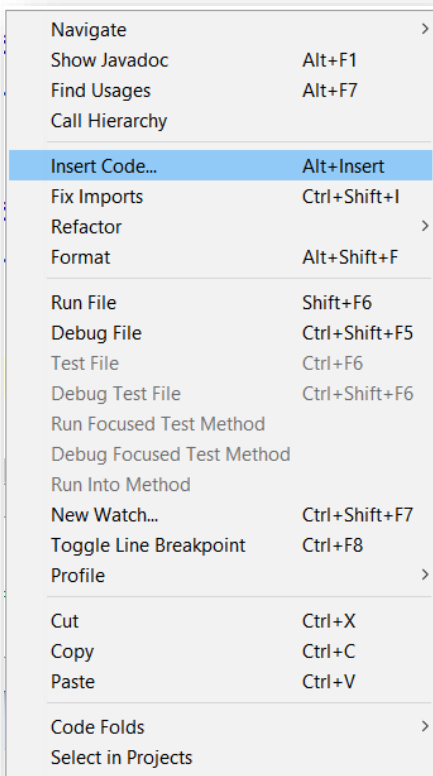
Kembali ke class Mahasiswa, arahkan cursor dibawah atribut "private double IPK"

```

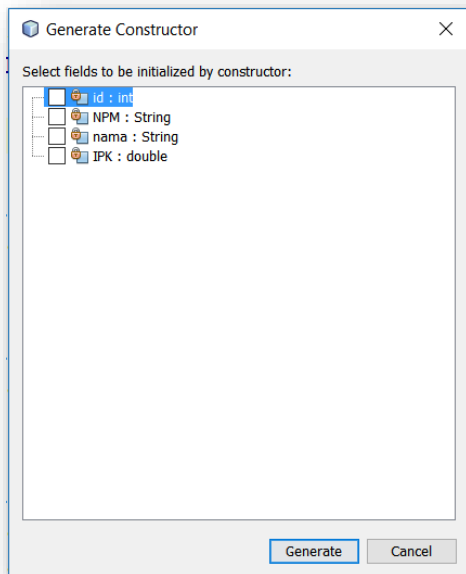
1  package Model;
2  public class Mahasiswa {
3      private int id;
4      private String NPM;
5      private String nama;
6      private double IPK;
7
8
9

```

Klik kanan, pilih **Insert Code...** kemudian pilih **Constructor...**

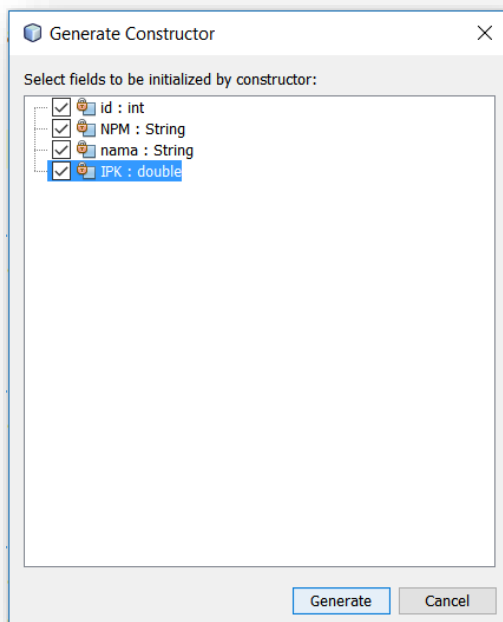


Tanpa memberikan centang langsung klik tombol **Generate**, maka menghasilkan constructor sebagai berikut



```
8  public Mahasiswa() {  
9      }  
10  
11
```

Sekali lagi lakukan **Klik kanan, Insert Code...**, dan pilih **Constructor...** kemudian berikan centang kepada semua atribut



```

8  public Mahasiswa() {
9      }
10
11  public Mahasiswa(int id, String NPM, String nama, double IPK) {
12      this.id = id;
13      this.NPM = NPM;
14      this.nama = nama;
15      this.IPK = IPK;
16  }

```

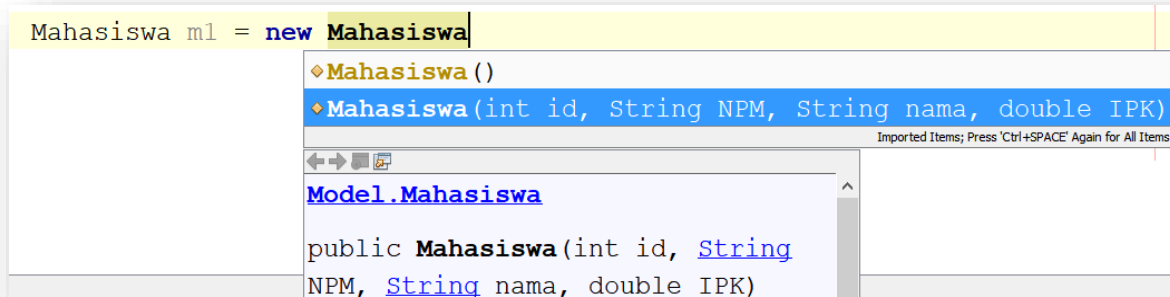
Hasil generate diatas merupakan sebuah method yang jalan ketika object di instance yang disebut sebagai method **Constructor**. Pada class mahasiswa terdapat 2 (dua) constructor yang sedikit berbeda. Constructor yang pertama tidak memiliki parameter/argument, sedangkan constructor kedua memiliki parameter dari seluruh atribut yang dimiliki oleh class Mahasiswa. Constructor pertama diperlukan agar instance object yang sudah kita buat tetap dikenali

```

7      Mahasiswa mahasiswa = new Mahasiswa();

```

Kembali ke **class utama**, buat **instance** baru menyesuaikan dengan constructor yang baru dibuat



Perhatikan ketika dituliskan statement "Mahasiswa m1 = new Mahasiswa" kemudian tekan tombol **ctrl+space** maka akan muncul rekomendasi seperti diatas. Ketika dipilih pilihan kedua maka muncul

```

Mahasiswa m1 = new Mahasiswa(0, NPM, nama, 0)

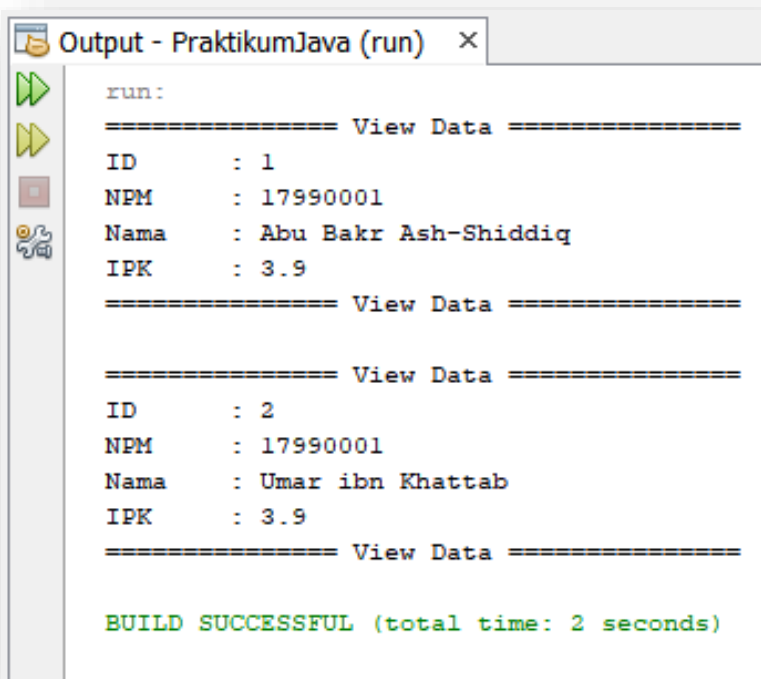
```


Isi sesuai yang diinginkan dan panggil method **viewData()**

```
16      Mahasiswa m1 = new Mahasiswa(  
17          2,  
18          "17990001",  
19          "Umar ibn Khattab",  
20          3.9);  
21  
22      m1.viewData();
```

Pengisian boleh saja mendatar, tanpa harus ganti baris seperti dicontoh.

Simpan dan jalankan project dengan menekan tombol **F6**



```
Output - PraktikumJava (run) x  
run:  
===== View Data =====  
ID      : 1  
NPM     : 17990001  
Nama    : Abu Bakr Ash-Shiddiq  
IPK     : 3.9  
===== View Data =====  
  
===== View Data =====  
ID      : 2  
NPM     : 17990001  
Nama    : Umar ibn Khattab  
IPK     : 3.9  
===== View Data =====  
  
BUILD SUCCESSFUL (total time: 2 seconds)
```

Dapat dilihat pada **Output** kita sudah bisa membuat 2 (dua) object, dengan cara instance yang berbeda, yang pertama dengan constructor standar, kemudian nilai diisi secara manual, yang kedua dengan constructor yang langsung mengisi nilai ke atribut

LATIHAN 1.1

Buat sebuah class dalam package **Model** beserta atribut, method, method viewData, dan constructornya. Instance 2(dua) object dari class tersebut pada class utama. Instance yang pertama menggunakan constructor tanpa parameter/argument, instance yang kedua menggunakan parameter/argument.

1.4 OVERLOADING

Overloading adalah penamaan method yang sama, pada class yang sama, tapi memiliki parameter yang berbeda. Contoh, jika mahasiswa memiliki method **inputKRS()**, maka kita masih bisa menggunakan nama method yang sama hanya saja dibedakan dengan adanya parameter, misalkan **inputKRS(int jmlSKS)**, bisa lagi dengan nama **inputKRS(double IPSemester)**, dan seterusnya selama tipe data parameter yang dipakai berbeda.

Kembali ke class Mahasiswa cari method **inputKRS()**

```
public void inputKRS(){
    System.out.println("Mahasiswa input KRS");
}
```

Buat overloading dari inputKRS dengan perintah copy dan paste

```
public void inputKRS() {
    System.out.println("Mahasiswa input KRS");
}

public void inputKRS() {
    System.out.println("Mahasiswa input KRS");
}
```

Muncul pesan error yang jika disorot pada tanda seru (!) akan muncul tulisan "method inputKRS() is already defined in class Mahasiswa". Ini dikarenakan nama method yang sama persis. Modifikasi method tersebut

```
public void inputKRS() {
    System.out.println("Mahasiswa input KRS");
}

public void inputKRS(int jumlahSKS) {
    System.out.println("Mahasiswa input KRS sebanyak: "+jumlahSKS+ "SKS");
}
```

Pesan error hilang dikarenakan method tersebut meskipun namanya sama, dianggap sebagai method yang berbeda karena memiliki parameter/argument yang berbeda, dimana method pertama tanpa parameter, sedangkan method yang kedua memiliki parameter **int jumlahSKS**.

Sebelumnya kita sudah mempraktekkan **overloading** pada pembuatan constructor, yang mana ada constructor tanpa parameter **Mahasiswa()** kemudian overloading-nya adalah constructor dengan parameter **Mahasiswa(int id, String NPM, String nama, double IPK)**

LATIHAN 1.2

Buat beberapa overloading dari method **inputKRS()** dengan argument :

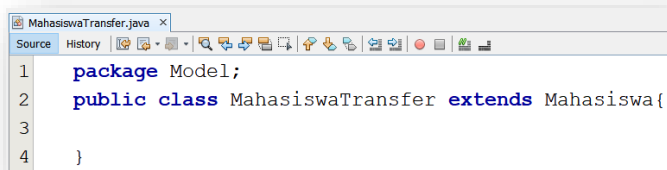
1. **(float IPSemester).**
2. **(String namaMahasiswa).**
3. **(String namaDosenPembimbing).**
4. **(int jumlahKRS, float IPSemester).**
5. **(int jumlahMK, float IPK).**

Jelaskan dan perbaiki jika ada error

1.5 INHERITANCE

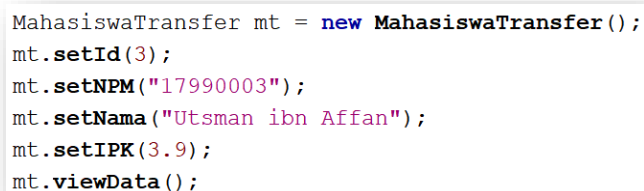
Inheritance secara bahasa adalah **pewarisan**, dalam konsep **OOP** inheritance berarti pewarisan seluruh atribut dan seluruh method dari sebuah class kepada sub-class, **terkecuali constructor**. Pewarisan ditandai dengan adanya kata kunci **extends** pada sub-class yang diwarisi.

Pada package **Model** buat class baru dengan nama **MahasiswaTransfer**



```
1 package Model;
2 public class MahasiswaTransfer extends Mahasiswa{
3
4 }
```

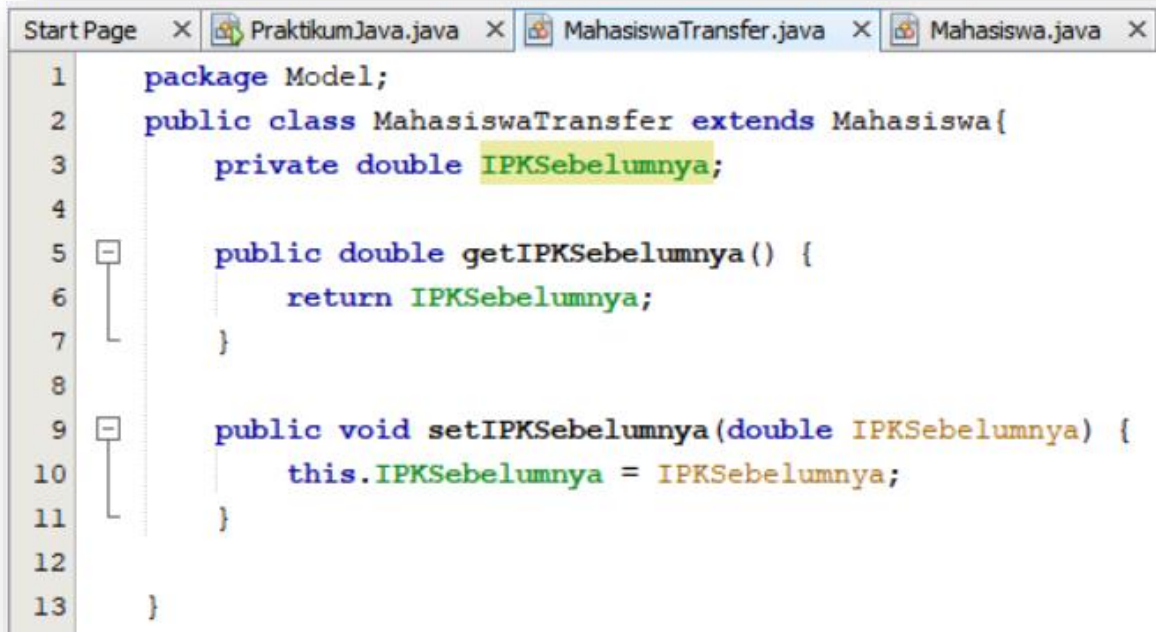
Kembali ke **class utama** tambahkan instance berikut, jangan lupa untuk import class MahasiswaTransfer dari package Model.



```
MahasiswaTransfer mt = new MahasiswaTransfer();
mt.setId(3);
mt.setNPM("17990003");
mt.setNama("Utzman ibn Affan");
mt.setIPK(3.9);
mt.viewData();
```

Dapat kita lihat class MahasiswaTransfer berjalan sama seperti class Mahasiswa meskipun isinya masih kosong, hal ini disebabkan adanya perintah **extends Mahasiswa** pada deklarasi class MahasiswaTransfer yang menyebabkan seluruh atribut dan method yang dimiliki oleh class Mahasiswa sebagai superclass menjadi dimiliki juga oleh MahasiswaTransfer sebagai subclass.

Subclass bisa memiliki atribut sendiri yang tidak dimiliki oleh superclass, misalkan jika untuk MahasiswaTransfer terdapat atribut IPK Sebelumnya, maka kita tambahkan pada class MahasiswaTransfer.



```
1 package Model;
2 public class MahasiswaTransfer extends Mahasiswa{
3     private double IPKSebelumnya;
4
5     public double getIPKSebelumnya() {
6         return IPKSebelumnya;
7     }
8
9     public void setIPKSebelumnya(double IPKSebelumnya) {
10        this.IPKSebelumnya = IPKSebelumnya;
11    }
12
13 }
```

Dengan demikian hanya object dari class MahasiswaTransfer yang memiliki atribut dan method IPKSebelumnya.

1.6 OVERRIDING

Overriding adalah penamaan method yang sama antara subclass dan superclass, sehingga sering disebut method di-override di subclass. Misalkan pada class Mahasiswa terdapat method **inputKRS()** yang berisikan pesan "Mahasiswa Input KRS". Maka untuk melakukan overriding terhadap ini adalah dengan membuat method yang sama persis **inputKRS()** pada subclass dari Mahasiswa yaitu MahasiswaTransfer tanpa harus dibedakan parameternya, dapat diisi pesan yang berbeda untuk melihat perbedaan hasil.

BAB 2

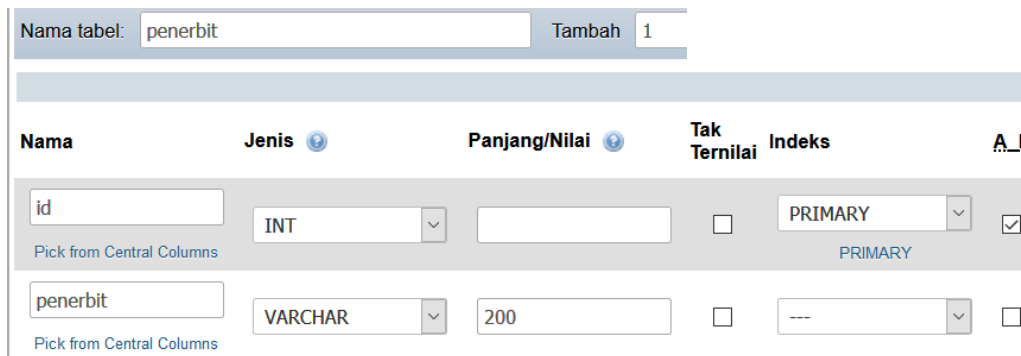
TEXT BASED FRAME AND MYSQL

2.1 DATABASE

Buka phpmyadmin, buat sebuah database dengan nama **db_perpus**.

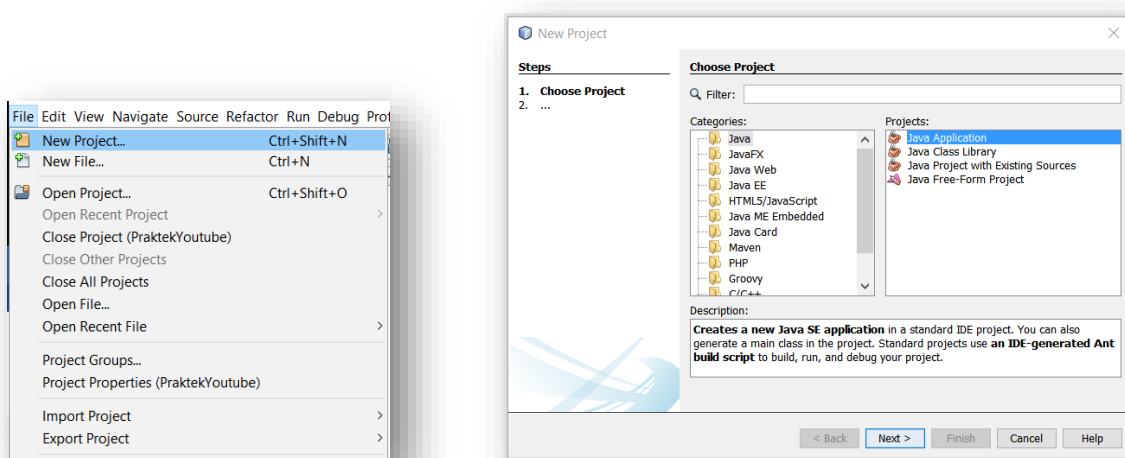


Buat sebuah table dengan nama **penerbit**, berisikan hanya 2 field **id**, dan **penerbit**, dengan tipe data sebagai berikut

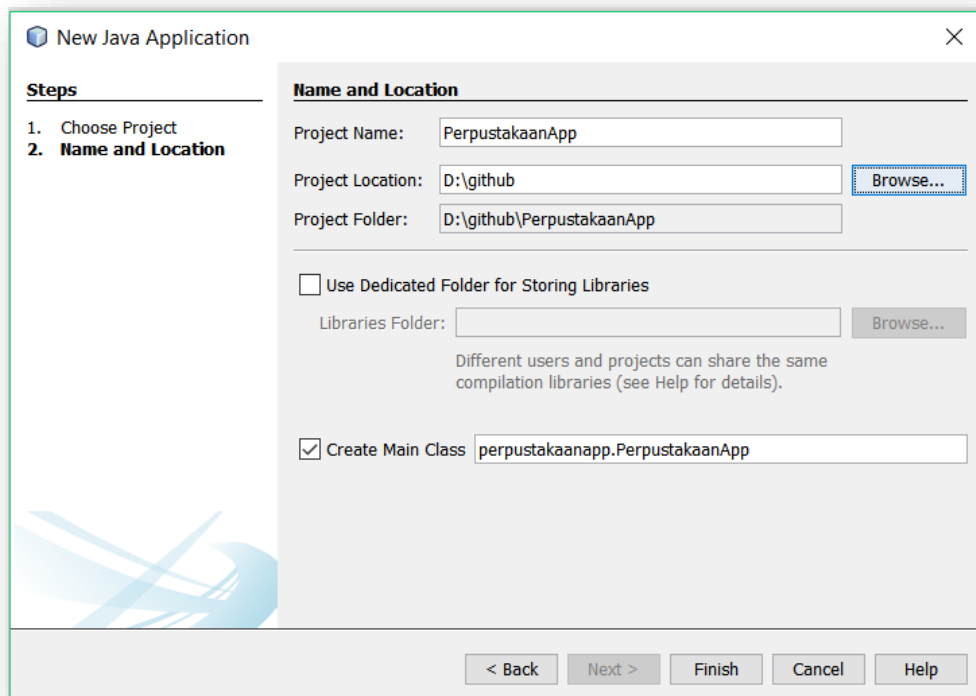


2.2 MEMBUAT PROJECT BARU

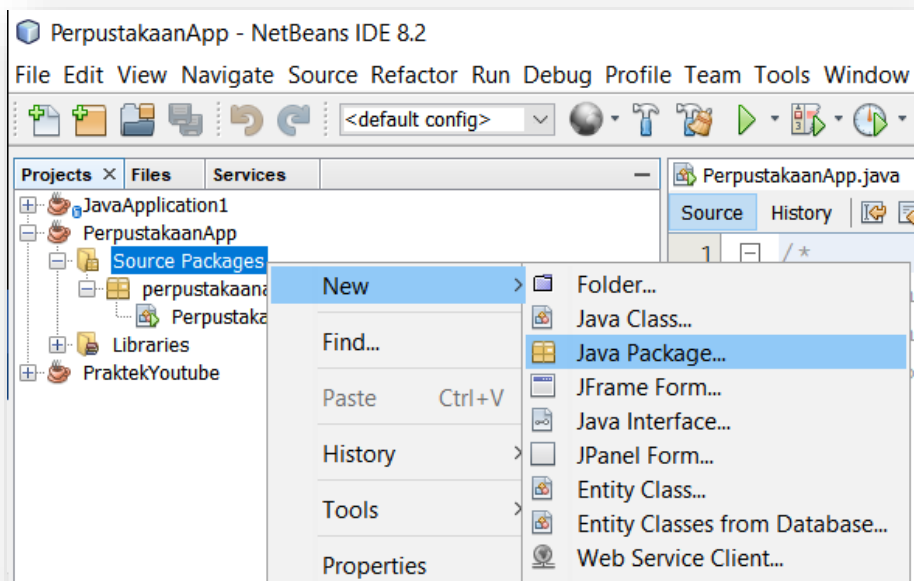
Buat project baru **New Project...** atau **Ctrl+Shift+N**, pilih **Java - Java Application**, klik **Next >**



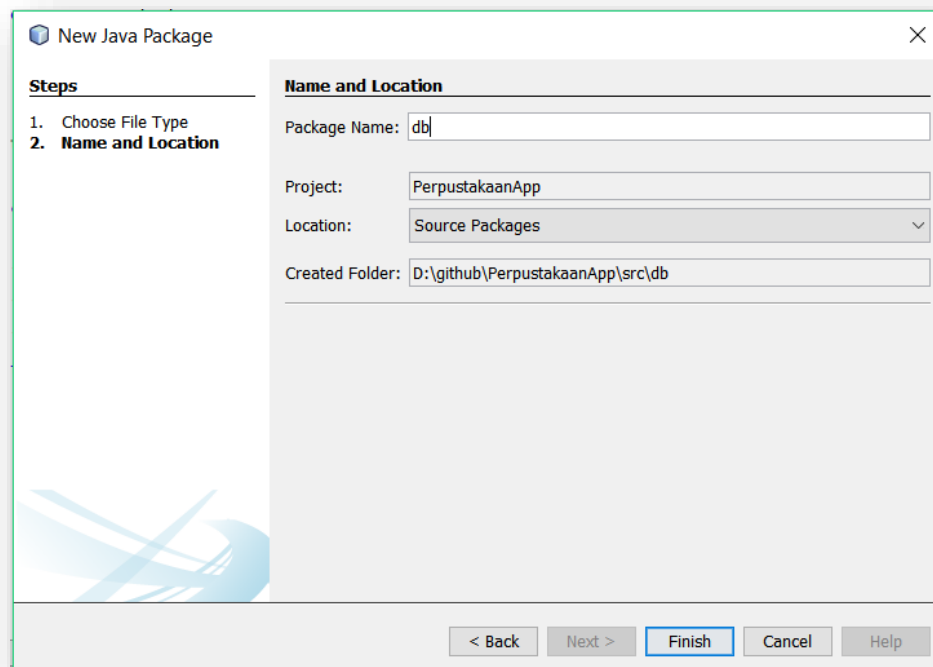
Isi **Project Name**, **Location**, dan **Folder**, kemudian klik **Finish**



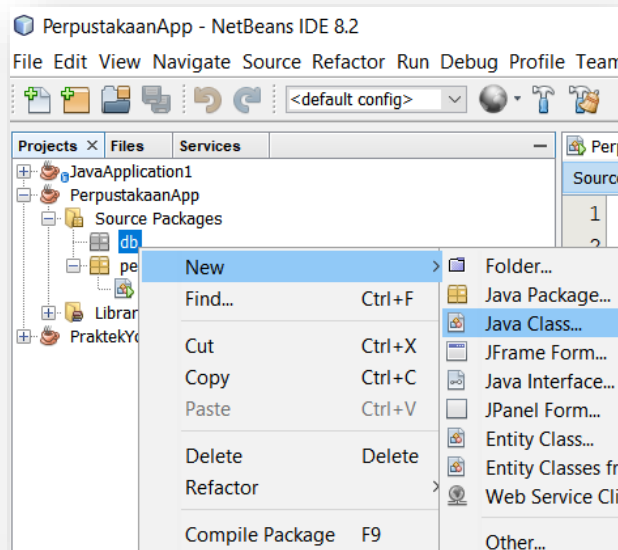
Buat package db dengan cara klik kanan pada **Source Packages** – **New** – **Java Package...**



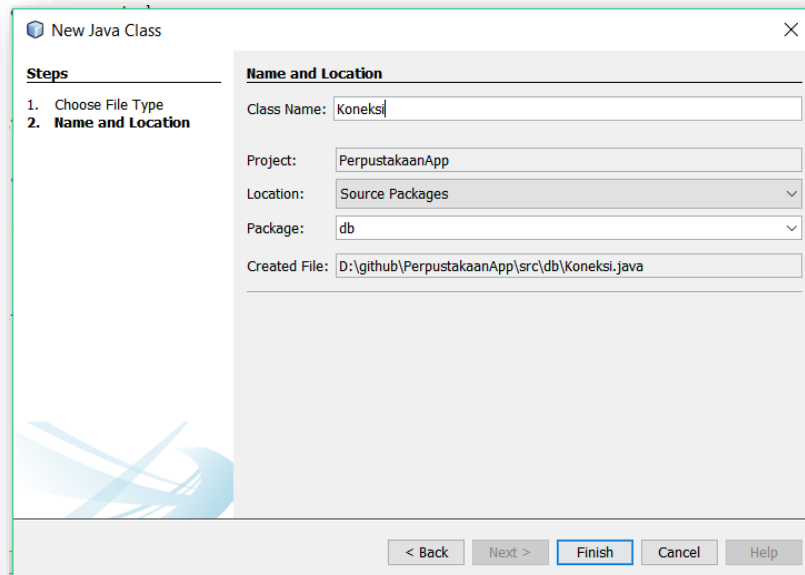
Isikan dengan **db**, kemudian **Finish**



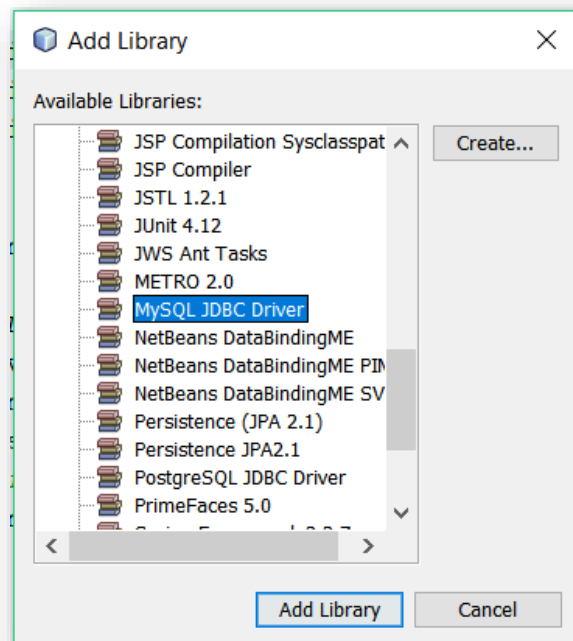
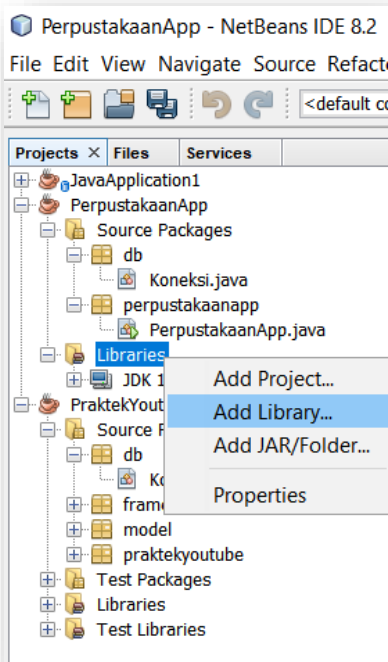
Buat sebuah class baru dengan nama **Koneksi** didalam package **db** dengan cara klik kanan pada package **db** – **New – Java Class...**



Isi kan **Class Name** dengan **Koneksi**, kemudian **Finish**



Tambahkan Library dengan klik kanan pada package **Libraries** – **Add Library...** kemudian pilih **MySQL JDBC Driver**, klik **Add Library**



Kembali ke class **Koneksi** isikan perintah berikut

```

1  package db;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.SQLException;
6
7  public class Koneksi {
8
9      private final String URL = "jdbc:mysql://localhost:3306/db_perpus";
10     private final String USER = "root";
11     private final String PASS = "";
12
13     public Connection getConnection() {
14         Connection con;
15         try {
16             Class.forName("com.mysql.jdbc.Driver");
17             con = DriverManager.getConnection(URL, USER, PASS);
18             System.out.println("Koneksi Berhasil");
19             return con;
20         } catch (ClassNotFoundException | SQLException ex) {
21             System.err.println("Koneksi Gagal");
22             return con=null;
23         }
24     }
25
26     public static void main(String[] args) {
27         Koneksi koneksi = new Koneksi();
28         koneksi.getConnection();
29     }
30 }

```

Coba lakukan **Run** pada class **Koneksi** dengan perintah **Shift+F6** perhatikan pesannya apakah berhasil atau gagal.

Buat package baru dengan nama **model** kemudian didalamnya buat sebuah class baru dengan nama **Penerbit**. Buat atribut, method constructor, getter and setternya seperti pada contoh halaman 13.

```

1 package model;
2 public class Penerbit {
3     private int id;
4     private String penerbit;
5
6     public Penerbit() {
7     }
8
9     public Penerbit(int id, String penerbit) {
10         this.id = id;
11         this.penerbit = penerbit;
12     }
13
14     public int getId() {
15         return id;
16     }
17
18     public void setId(int id) {
19         this.id = id;
20     }
21
22     public String getPenerbit() {
23         return penerbit;
24     }
25
26     public void setPenerbit(String penerbit) {
27         this.penerbit = penerbit;
28     }
29 }

```

2.3 PENERBITTAMPILFRAME

2.3.1 TAMPIL FRAME

Buat sebuah package baru dengan nama **frame**, kemudian didalamnya buat sebuah class dengan nama **PenerbitTampilFrame** kemudian tambahkan atribut berikut:

```

public class PenerbitTampilFrame extends JFrame {

    JLabel jLabel1 = new JLabel("Cari");
    JTextField eCari = new JTextField();
    JButton bCari = new JButton("Cari");

    String header [] = {"Id", "Penerbit"};
    TableModel tableModel = new DefaultTableModel(header, 0);
    JTable tPenerbit = new JTable(tableModel);
    JScrollPane jScrollPane = new JScrollPane(tPenerbit);

    JButton bTambah = new JButton("Tambah");
    JButton bUbah = new JButton("Ubah");
    JButton bHapus = new JButton("Hapus");
    JButton bBatal = new JButton("Batal");
    JButton bTutup = new JButton("Tutup");

    Penerbit penerbit;
}

```

Buat method `setKomponen()` yang berfungsi menaruh komponen yang sudah kita deklarasikan sebelumnya

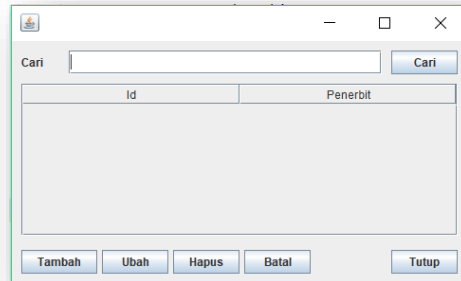
```
public void setKomponen() {  
    getContentPane().setLayout(null);  
    getContentPane().add(jLabel1);  
    getContentPane().add(eCari);  
    getContentPane().add(jScrollPane);  
    getContentPane().add(bCari);  
    getContentPane().add(bTambah);  
    getContentPane().add(bUbah);  
    getContentPane().add(bHapus);  
    getContentPane().add(bBatal);  
    getContentPane().add(bTutup);  
  
    jLabel1.setBounds(10,10,50,25);  
    eCari.setBounds(60,10,330,25);  
    bCari.setBounds(400,10,70,25);  
    bTutup.setBounds(400,220,70,25);  
    bTambah.setBounds(10,220,80,25);  
    bUbah.setBounds(95,220,70,25);  
    bHapus.setBounds(170,220,70,25);  
    bBatal.setBounds(245,220,70,25);  
    jScrollPane.setBounds(10,45,460,160);  
  
    setVisible(true);  
}
```

Buat constructor untuk class ini dengan menyertakan pemanggilan method `setKomponen()`

```
public PenerbitTampilFrame() {  
    setSize(500,300);  
    setLocationRelativeTo(null);  
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);  
    setKomponen();  
}
```

Buat method main untuk class ini dan coba Run dengan **Shift+F6**

```
public static void main(String[] args) {
    PenerbitTampilFrame penerbitTampilFrame = new PenerbitTampilFrame();
}
```



2.3.2 TAMPIL DATA

Kita lanjutkan dengan membuat method **getPenerbitList()** untuk mengambil data dari database

```
public ArrayList<Penerbit> getPenerbitList(String keyword) {
    ArrayList<Penerbit> penerbitList = new ArrayList<Penerbit>();
    Koneksi koneksi = new Koneksi();
    Connection connection = koneksi.getConnection();

    String query = "SELECT * FROM penerbit "+keyword;
    Statement statement;
    ResultSet resultSet;

    try {
        statement = connection.createStatement();
        resultSet = statement.executeQuery(query);
        while(resultSet.next()) {
            penerbit = new Penerbit(resultSet.getInt("id"),
                                     resultSet.getString("penerbit"));
            penerbitList.add(penerbit);
        }
    } catch (SQLException | NullPointerException ex) {
        System.err.println("Koneksi Null Gagal");
    }
    return penerbitList;
}
```

Method berikutnya adalah **selectPenerbit()** yang berfungsi memanggil **getPenerbitList()** kemudian meletakkannya kedalam table **tPenerbit**.

```
public final void selectPenerbit(String keyword){
    ArrayList<Penerbit> list = getPenerbitList(keyword);
    DefaultTableModel model = (DefaultTableModel)tPenerbit.getModel();
    Object[] row = new Object[2];

    for (int i = 0; i < list.size(); i++) {
        row[0] = list.get(i).getId();
        row[1] = list.get(i).getPenerbit();

        model.addRow(row);
    }
}
```

Berikutnya method **resetTable()** yang berfungsi melakukan reset pada table pada setiap terjadi perubahan.

```
public final void resetTable(String keyword){
    DefaultTableModel model = (DefaultTableModel)tPenerbit.getModel();
    model.setRowCount(0);
    selectPenerbit(keyword);
}
```

Terakhir tambahkan method **resetTable()** kedalam method **setKomponen()** pada baris sebelum **setVisible(true)**

```
bhapus.setBounds(170,220,70,25);
bBatal.setBounds(245,220,70,25);
jScrollPane.setBounds(10,45,460,160);

resetTable("");
setVisible(true);
}
```

Jalankan dengan perintah **Shift+F6**

2.3.3 LISTENER

Agar tombol – tombol yang ada bisa berfungsi kita tambahkan method **setListener()**. Tuliskan kode seperti berikut kemudian tekan **Alt+Enter** dan pilih **Implement all abstract methods**.

```
public void setListener() {  
    bTutup.addActionListener(new ActionListener() {  
        Implement all abstract methods  
    }  
}
```

Maka akan terbentuk method baru

```
public void setListener() {  
    bTutup.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            throw new UnsupportedOperationException("Not supported yet.");  
        }  
    });  
}
```

Ganti dengan perintah berikut

```
public void setListener() {  
    bTutup.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            dispose();  
        }  
    });  
}
```

Tambahkan method **setListener()** ke dalam method **setKomponen()**, kemudian **Shift+F6** dan coba tekan tombol **Tutup**

```
hapus.setBounds(170,220,70,25);
bBatal.setBounds(245,220,70,25);
jScrollPane.setBounds(10,45,460,160);

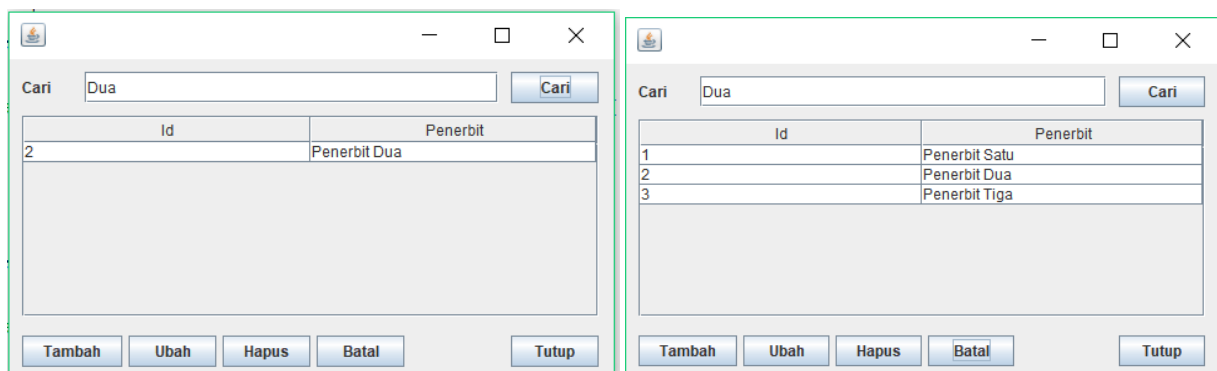
resetTable("");
setListener();
setVisible(true);
}
```

Tambahkan pada perintah untuk tombol **Cari**, **Batal**,

```
bCari.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        resetTable(" WHERE Penerbit like '%" + eCari.getText() + "%'");
    }
});

bBatal.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        resetTable("");
    }
});
```

Uji hasilnya dengan mengetikkan kata kunci dalam **eCari** kemudian klik tombol **Cari**, tekan tombol **Batal** untuk melakukan reset table kembali seperti semula.



Tarik nafas dalam – dalam... berikut isi perintah tombol **Hapus**

```
bHapus.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int i = tPenerbit.getSelectedRow();
        int pilihan = JOptionPane.showConfirmDialog(
            null,
            "Yakin mau hapus ?",
            "Konfirmasi hapus",
            JOptionPane.YES_NO_OPTION);
        if(pilihan==0){
            if(i>=0){
                try {
                    TableModel model = tPenerbit.getModel();
                    Koneksi koneksi = new Koneksi();
                    Connection con = koneksi.getConnection();
                    String executeQuery = "delete from penerbit where id =?";
                    PreparedStatement ps = con.prepareStatement(executeQuery);
                    ps.setString(1, model.getValueAt(i,0).toString());
                    ps.executeUpdate();
                } catch (SQLException ex) {
                    System.err.println(ex);
                }
            }else{
                JOptionPane.showMessageDialog(null, "Pilih data yang ingin dihapus");
            }
        }
        resetTable("");
    }
});
```

Sebelum mengisi perintah untuk tombol **Tambah** dan **Ubah** kita harus punya dulu frame input yang digunakan untuk menginput ataupun mengubah data yang dipilih pada table

2.4 PENERBITTAMBAHFRAME

2.4.1 TAMPIL FRAME

Buat sebuah class dengan nama **PenerbitTambahFrame** kemudian tambahkan atribut berikut:

```
public class PenerbitTambahFrame extends JFrame {
    int status;

    private final int SEDANG_TAMBAH = 101;
    private final int SEDANG_UBAH = 102;

    JLabel jLabel1 = new JLabel("Id");
    JLabel jLabel2 = new JLabel("Penerbit");

    JTextField eId = new JTextField();
    JTextField ePenerbit = new JTextField();

    JButton bSimpan = new JButton("Simpan");
    JButton bBatal = new JButton("Batal");
}
```

Buat method `setKomponen()` yang berfungsi menaruh komponen yang sudah kita deklarasikan sebelumnya

```
public void setKomponen() {
    getContentPane().setLayout(null);
    getContentPane().add(jLabel1);
    getContentPane().add(jLabel2);
    getContentPane().add(eId);
    getContentPane().add(ePenerbit);
    getContentPane().add(bSimpan);
    getContentPane().add(bBatal);

    jLabel1.setBounds(70, 10, 50, 25);
    jLabel2.setBounds(30, 40, 50, 25);

    eId.setBounds(100, 10, 50, 25);
    ePenerbit.setBounds(100, 40, 270, 25);

    bSimpan.setBounds(160, 70, 100, 25);
    bBatal.setBounds(270, 70, 100, 25);

    eId.setEditable(false);
    setVisible(true);
    ePenerbit.requestFocus();
}
```

Buat constructor untuk class ini dengan menyertakan pemanggilan method setKomponen()

```
public PenerbitTambahFrame() {
    status = SEDANG_TAMBAH;
    setSize(420,180);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    setKomponen();
}

public PenerbitTambahFrame(Penerbit penerbit) {
    status = SEDANG_UBAH;
    setSize(420,180);
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    eId.setText(String.valueOf(penerbit.getId()));
    ePenerbit.setText(penerbit.getPenerbit());
    setKomponen();
}
```

Terdapat 2 (dua) konstruktor yang berfungsi untuk membedakan kondisi program apakah sedang akan menambah atau mengubah data.

Class ini akan dipanggil melalui class **PenerbitTampilFrame** sehingga kita kembali ke class tersebut dan mengisikan perintah untuk tombol **Tambah** dan **Ubah**

```
bUbah.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int i = tPenerbit.getSelectedRow();
        if(i>=0){
            TableModel model = tPenerbit.getModel();
            penerbit = new Penerbit();
            penerbit.setId(Integer.parseInt(model.getValueAt(i, 0).toString()));
            penerbit.setPenerbit(model.getValueAt(i, 1).toString());
            PenerbitTambahFrame penerbitTambahFrame = new PenerbitTambahFrame(penerbit);
        }else{
            JOptionPane.showMessageDialog(null, "Pilih data yang ingin diubah");
        }
    }
});
```

```

bTambah.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        PenerbitTambahFrame penerbitTambahFrame = new PenerbitTambahFrame();
    }
});

```

Hasil bisa dicoba dengan menjalankan **PenerbitTampilFrame** kemudian klik tombol **Tambah** ataupun **Ubah**.

2.4.2 LISTENER

Agar tombol – tombol yang ada bisa berfungsi kita tambahkan method **setListener()**. Tuliskan kode seperti berikut kemudian tekan **Alt+Enter** dan pilih **Implement all abstract methods**.

```

public void setListener() {
    bBatal.addActionListener(new ActionListener());
}

```

Implement all abstract methods

Isikan perintah berikut

```

public void setListener() {
    bBatal.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            dispose();
        }
    });
}

```

Dengan cara yang sama, tambahkan perintah untuk tombol **Simpan**

```
bSimpan.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            Koneksi koneksi = new Koneksi();
            Connection con = koneksi.getConnection();
            PreparedStatement ps;
            if(status==SEDANG_TAMBAH) {
                String executeQuery = "insert into penerbit (penerbit) values (?)";
                ps = con.prepareStatement(executeQuery);
                ps.setString(1, ePenerbit.getText());
            }else{
                String executeQuery = "update penerbit set penerbit=? where id=?";
                ps = con.prepareStatement(executeQuery);
                ps.setString(1, ePenerbit.getText());
                ps.setString(2, eId.getText());
            }
            ps.executeUpdate();
        } catch (SQLException ex) {
            System.err.println(ex);
        }
        dispose();
    }
});
```

Tambahkan method **setListener()** ke dalam method **setKomponen()**

```
eId.setEditable(false);
setVisible(true);
ePenerbit.requestFocus();
setListener();
}
```

Agar ketika dilakukan proses **Tambah** maupun **Ubah** table langsung melakukan reset dan menampilkan data yang sudah diupdate maka kembali ke class **PenerbitTampilFrame** pada method **setListener()** tambahkan perintah berikut

```
addWindowListener(new java.awt.event.WindowAdapter() {  
    public void windowActivated(java.awt.event.WindowEvent evt) {  
        resetTable("");  
    }  
});
```

Jalankan class **PenerbitTampilFrame** dengan **Shift+F6** dan coba fitur **Tambah** dan **Ubah**

The screenshot shows a window titled "PenerbitTampilFrame". It contains a search bar labeled "Cari" with the text "Dua" entered, and a "Cari" button. Below the search bar is a table with two columns: "Id" and "Penerbit". The table contains three rows of data:

Id	Penerbit
1	Penerbit Satu
2	Penerbit Dua
3	Penerbit Tiga

At the bottom of the window are five buttons: "Tambah", "Ubah", "Hapus", "Batal", and "Tutup".

The screenshot shows a dialog box for editing a publisher. It has two input fields: "Id" and "Penerbit". The "Id" field is empty, and the "Penerbit" field contains the text "Empat". Below the input fields are two buttons: "Simpan" (Save) and "Batal" (Cancel).

The screenshot shows the main window after clicking the "Ubah" button. The search bar is empty, and the table now contains four rows of data:

Id	Penerbit
1	Penerbit Satu
2	Penerbit Dua
3	Penerbit Tiga
4	Empat

The buttons at the bottom remain the same: "Tambah", "Ubah", "Hapus", "Batal", and "Tutup".

BAB 3

VISUAL BASED FRAME AND MYSQL

3.1 DATABASE

Buat sebuah table dengan nama **petugas**



Buat tabel

Nama: Jumlah kolom:

berisikan hanya 4 field **id**, **nama_petugas**, **username**, dan **password**, dengan tipe data sebagai berikut

Nama	Jenis	Panjang/Nilai	A_I
<input type="text" value="id"/>	INT	<input type="text"/>	<input checked="" type="checkbox"/>
<small>Pick from Central Columns</small>			
<input type="text" value="nama_petugas"/>	VARCHAR	200	<input type="checkbox"/>
<small>Pick from Central Columns</small>			
<input type="text" value="username"/>	VARCHAR	200	<input type="checkbox"/>
<small>Pick from Central Columns</small>			
<input type="text" value="password"/>	VARCHAR	200	<input type="checkbox"/>
<small>Pick from Central Columns</small>			

Buat sebuah class **Petugas** didalam package **model** dengan isi atribut

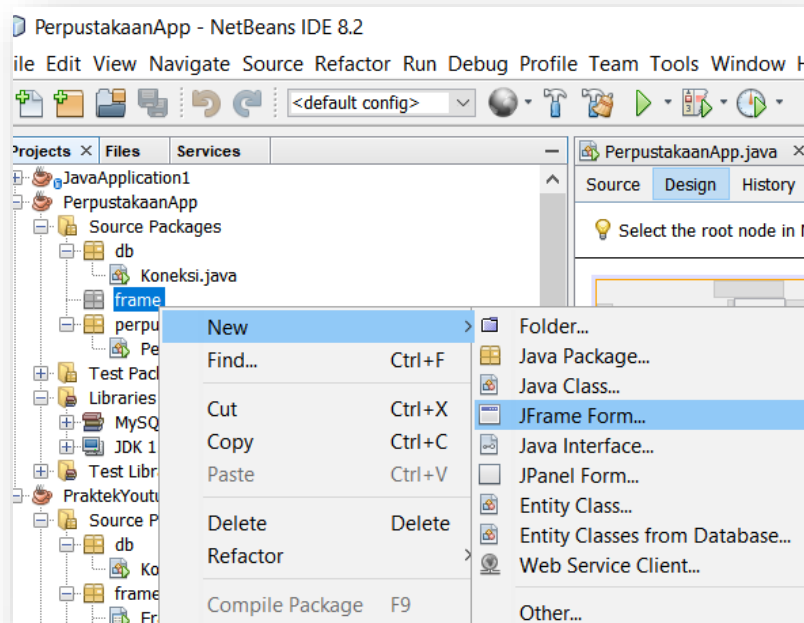
```
package model;  
public class Petugas {  
    private int id;  
    private String namaPetugas;  
    private String username;  
    private String password;  
}
```

Buat Getter and Setter dan 2 constructor seperti pada halaman 13

3.2 PETUGASTAMPILFRAME

3.2.1 TAMPIL FRAME

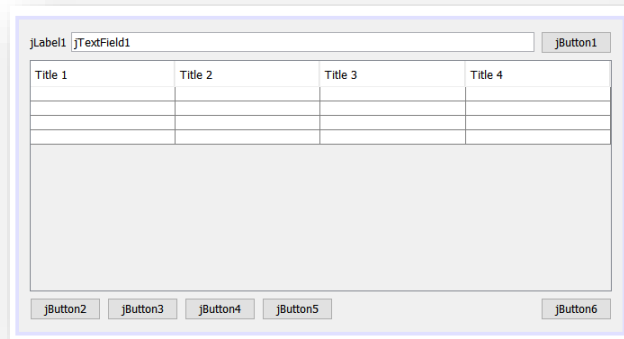
Buat sebuah class JFrame bernama **PetugasTampilFrame** dengan cara klik kanan package **frame** – **New** – **JFrame Form...**



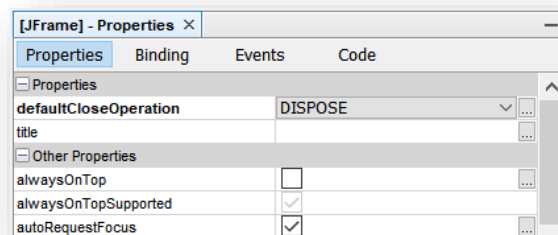
Isikan **Class Name** dengan **PetugasTampilFrame** kemudian klik **Finish**. Kemudian tambahkan komponen **Swing Control** berikut

1. JLabel 1 (satu) buah
2. JTextField 1 (satu) buah
3. JButton 6 (enam) buah
4. JScrollPane 1 (satu) buah
5. JTable 1 (satu) buah

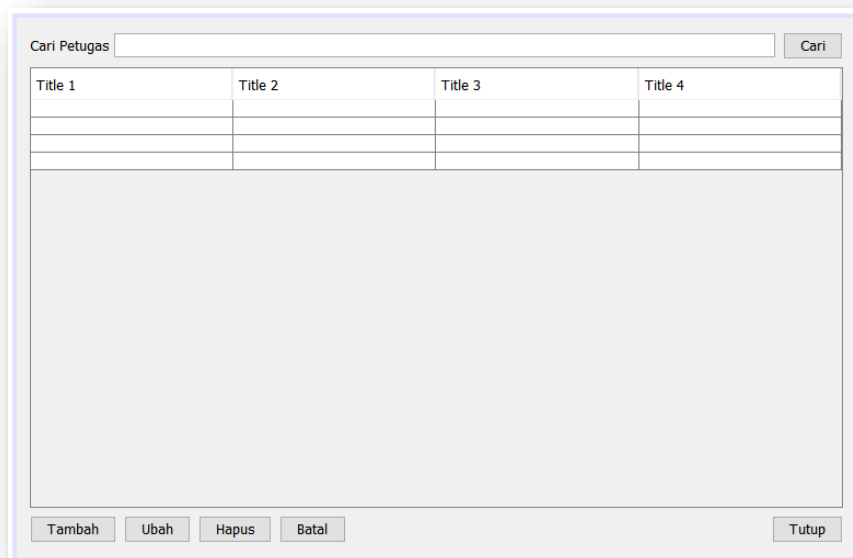
Sebelum meletakkan JTable terlebih dahulu tambahkan JScrollPane, atur seukuran table, kemudian letakkan JTable didalam JScrollPane tersebut, atur komponen lainnya sehingga rapi seperti contoh berikut



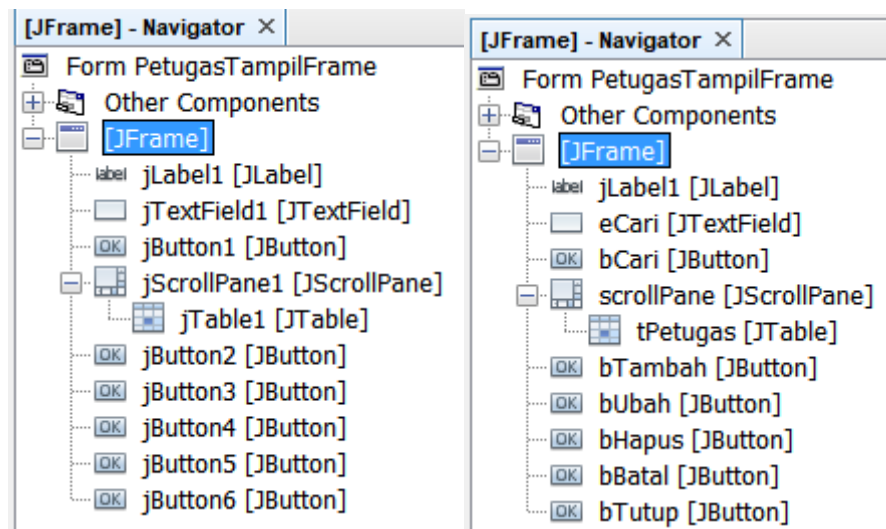
Ganti properties (terletak di sebelah kanan bawah GUI) **defaultCloseOperation** milik **Jframe**, menjadi **DISPOSE**, agar jika form ditutup aplikasi tidak langsung berhenti



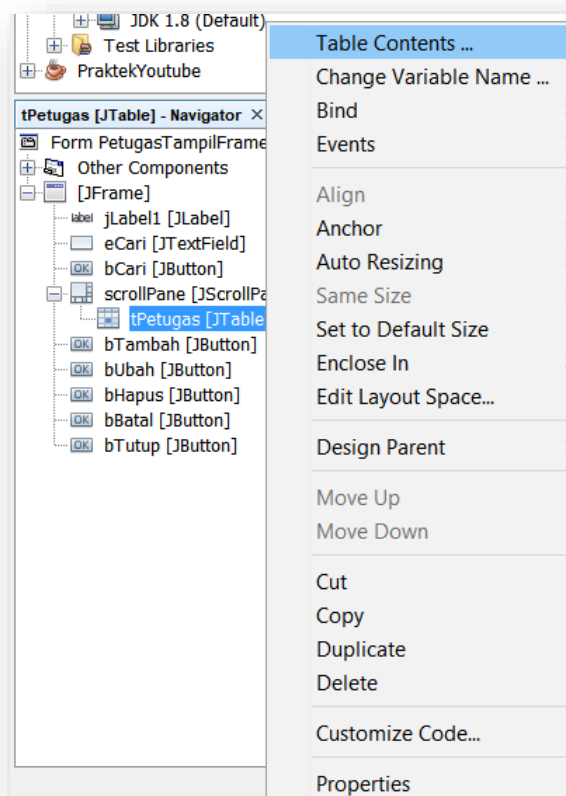
Ubah text pada komponen, dengan cara pilih komponen, tekan **F2** kemudian ganti sesuai dengan gambar berikut



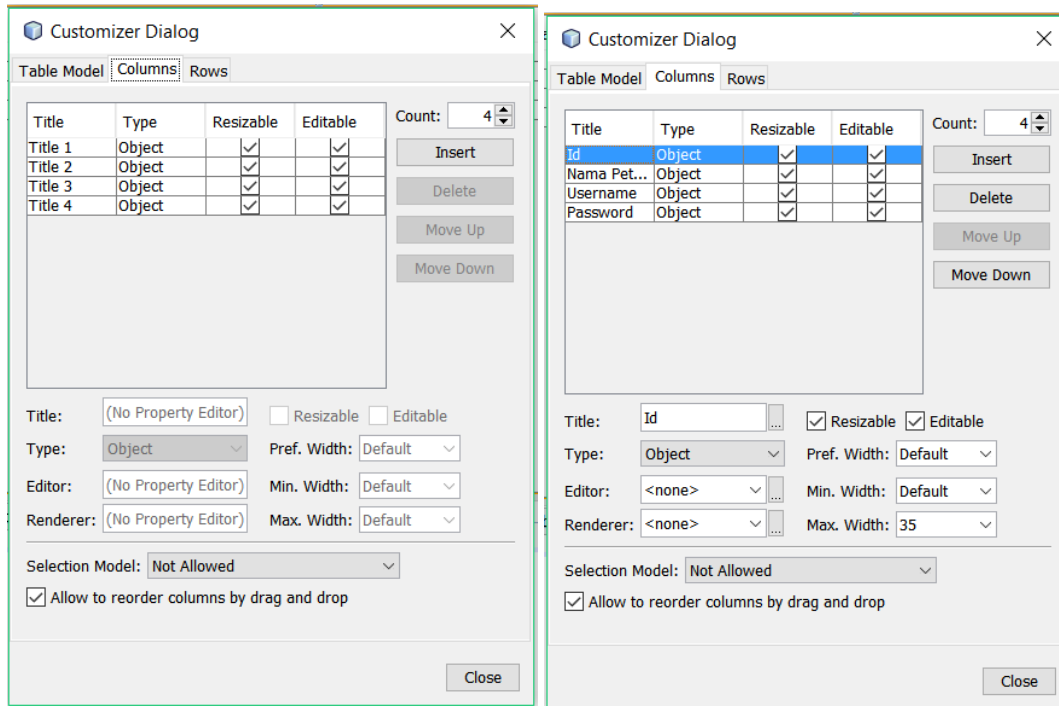
Pada sebelah kiri bawah ganti nama komponen dengan memilih komponen, tekan **F2**, ganti namanya.



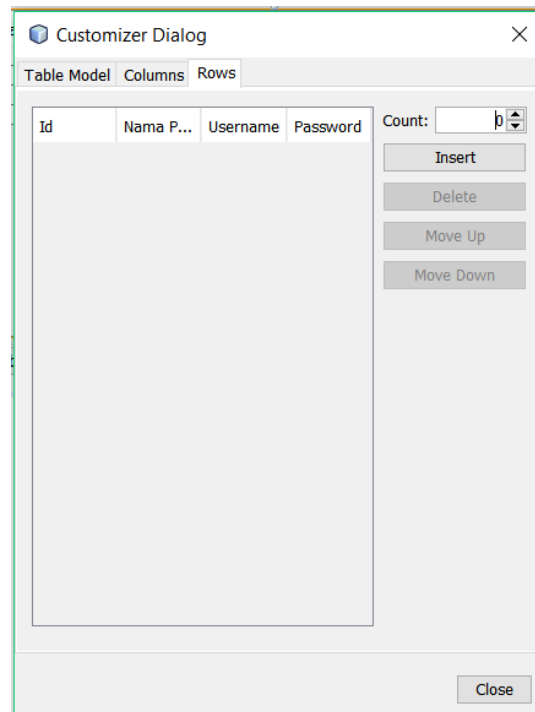
Klik kanan **tPetugas** kemudian pilih **Table Contents ...**



Menuju tab **Columns** khusus untuk Id kita buat **Max. Width 35** sedangkan yang lain tetap biarkan pada kondisi Default

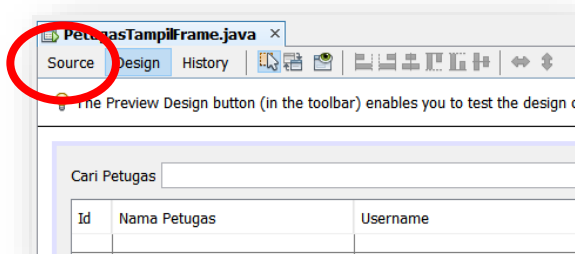


Menuju tab **Rows**, ubah count dari 4 menjadi 0



3.2.2 TAMPIL DATA

Kita lanjutkan dengan membuat method **getPetugasList()** untuk mengambil data dari database dengan cara pindah dulu ke mode **Source**



Kemudian tambahkan atribut **Petugas petugas** diantara deklarasi class dan constructor

```
public class PetugasTampilFrame extends javax.swing.JFrame {
    Petugas petugas;

    public PetugasTampilFrame() {
        initComponents();
    }
}
```

Kemudian tuliskan method **getPetugasList()** setelah constructor

```

public ArrayList<Petugas> getPetugasList(String keyword) {
    ArrayList<Petugas> petugasList = new ArrayList<Petugas>();
    Koneksi koneksi = new Koneksi();
    Connection connection = koneksi.getConnection();

    String query = "SELECT * FROM petugas "+keyword;
    Statement st;
    ResultSet rs;

    try {
        st = connection.createStatement();
        rs = st.executeQuery(query);
        while(rs.next()){
            petugas = new Petugas(
                rs.getInt("id"),
                rs.getString("nama_petugas"),
                rs.getString("username"),
                rs.getString("password"));
            petugasList.add(petugas);
        }
    } catch (SQLException | NullPointerException ex) {
        System.err.println("Koneksi Null Gagal");
    }
    return petugasList;
}

```

Dilanjutkan dengan method **selectPetugas()**

```

public void selectPetugas(String keyword) {
    ArrayList<Petugas> list = getPetugasList(keyword);
    DefaultTableModel model = (DefaultTableModel)tPetugas.getModel();
    Object[] row = new Object[4];

    for (int i = 0; i < list.size(); i++) {
        row[0] = list.get(i).getId();
        row[1] = list.get(i).getNamaPetugas();
        row[2] = list.get(i).getUsername();
        row[3] = list.get(i).getPassword();

        model.addRow(row);
    }
}

```

Dilanjutkan dengan method **resetTable()**

```
public final void resetTable(String keyword) {
    DefaultTableModel model = (DefaultTableModel) tPetugas.getModel();
    model.setRowCount(0);
    selectPetugas(keyword);
}
```

Tambahkan perintah berikut pada **constructor**

```
public PetugasTampilFrame() {
    initComponents();
    setLocationRelativeTo(null);
    resetTable("");
}
```

Sekarang **JTable** seharusnya sudah mampu menampilkan isi table **petugas**. Jalankan dengan **Shift+F6** jika menampilkan data kosong, maka periksa isi table di database.

3.2.3 LISTENER

Pembuatan frame dengan metode visual tidak memerlukan method `setListener()` untuk memberikan `eventListener` pada komponen. Jika ingin memberikan `eventListener` pada tombol **Tutup** maka cukup klik 2x tombol **Tutup** pada saat mode **Design** maka kita dapati method berikut, dapat diisikan dengan perintah berikut

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

Tombol **Batal**

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

Tombol Cari

```
private void bCariActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    resetTable(" WHERE nama_petugas like '%" + eCari.getText() + "%' OR"
        + " username like '%" + eCari.getText() + "%'");
}
```

Tombol... Hapus

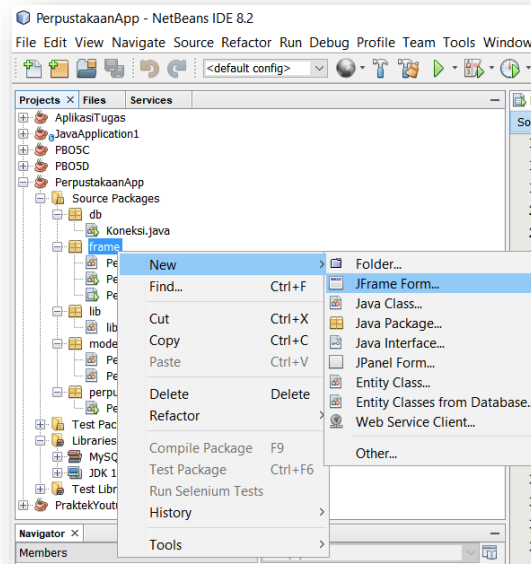
```
private void bHapusActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int i = tPetugas.getSelectedRow();
    int pilihan = JOptionPane.showConfirmDialog(
        null,
        "Yakin mau hapus ?",
        "Konfirmasi hapus",
        JOptionPane.YES_NO_OPTION);
    if(pilihan==0){
        if(i>=0){
            try {
                TableModel model = tPetugas.getModel();
                Koneksi koneksi = new Koneksi();
                Connection con = koneksi.getConnection();
                String executeQuery = "delete from petugas where id =?";
                PreparedStatement ps = con.prepareStatement(executeQuery);
                ps.setString(1, model.getValueAt(i,0).toString());
                ps.executeUpdate();
            } catch (SQLException ex) {
                System.err.println(ex);
            }
        }else{
            JOptionPane.showMessageDialog(null, "Pilih data yang ingin dihapus");
        }
    }
    resetTable("");
}
```

Sebelum mengisikan perintah untuk tombol **Tambah** dan **Ubah** kita harus punya dulu frame input yang digunakan untuk menginput ataupun mengubah data yang dipilih pada table

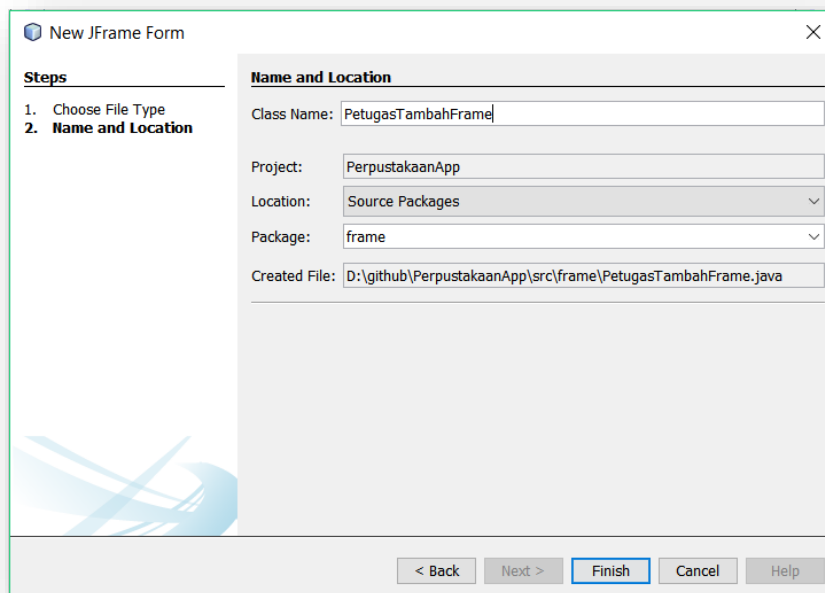
3.3 PETUGASTAMBAHFRAME

3.3.1 TAMPIL FRAME

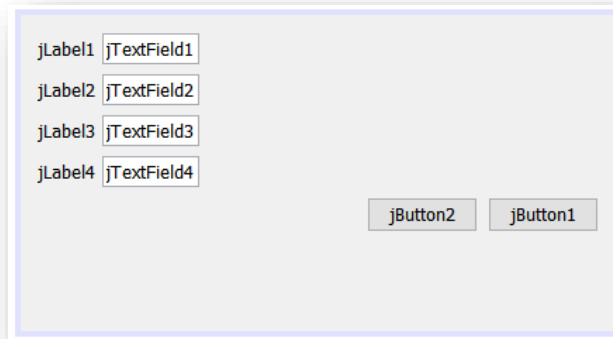
Buat JFrame baru dalam package JFrame



Isikan dengan nama **PetugasTambahFrame** kemudian klik **Finish**



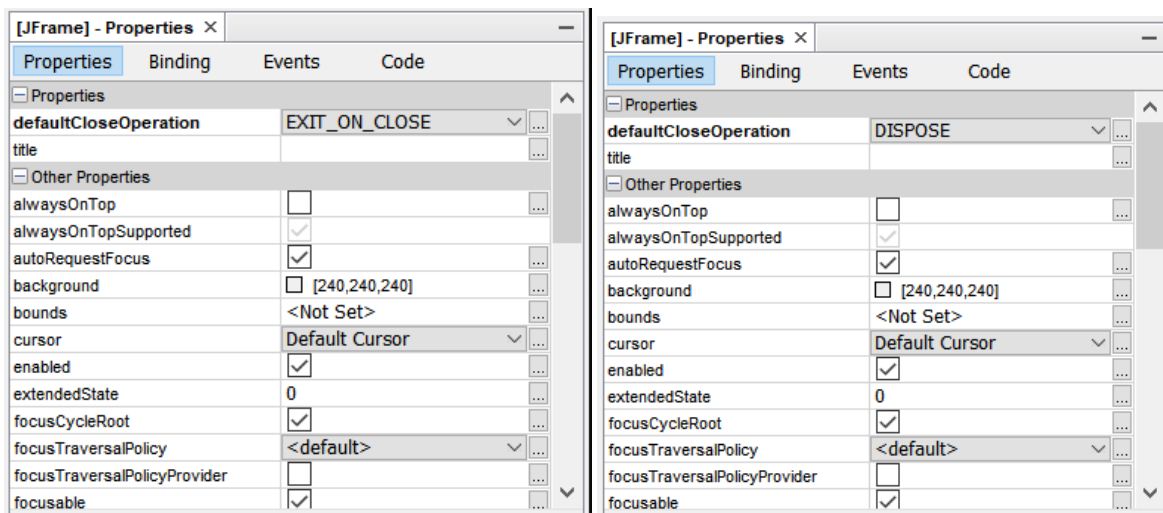
Letakkan komponen berikut



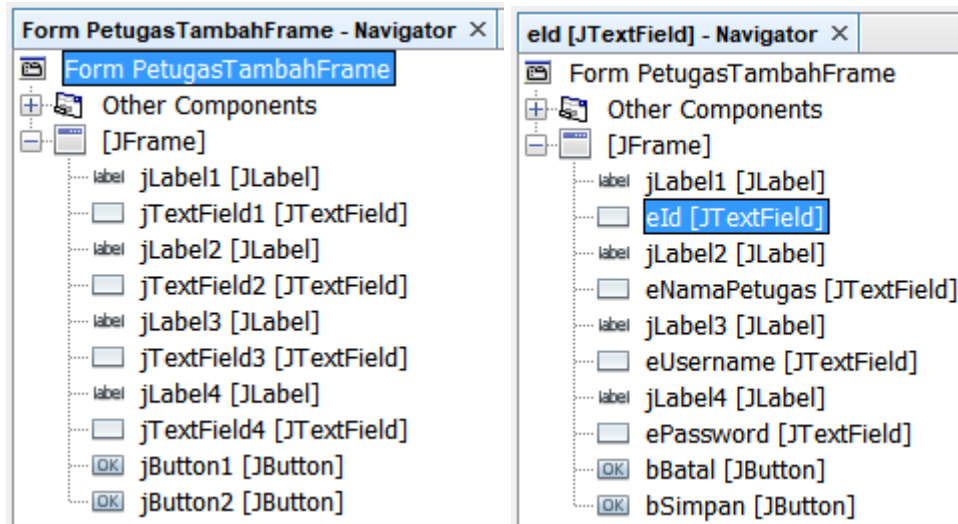
Ganti tulisan pada komponen tersebut dengan memilih komponen kemudian tekan **F2** dan ganti seperti berikut



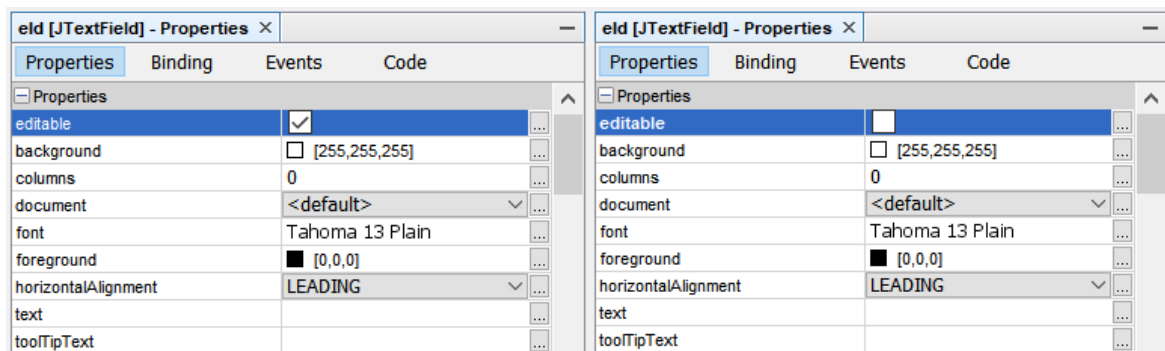
Klik frame 1 kali kemudian pada windows NetBeans bagian **kanan bawah**, ganti properties EXIT_ON_CLOSE menjadi DISPOSE



Ganti nama-nama komponen, dapat dilihat pada NetBeans bagian **kiri bawah**, ganti dengan cara pilih komponen, tekan **F2** ganti sesuai petunjuk



Pilih **eld** ganti properties **editable** menjadi **false** dengan cara menghilangkan centangnya



3.3.2 LISTENER

Tambahkan 1 (satu) variable dan 2 (dua) konstanta diantara deklarasi class dengan constructor

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

    int status;

    private final int SEDANG TAMBAH = 101;
    private final int SEDANG UBAH = 102;

    public PetugasTambahFrame() {
        initComponents();
    }
}
```

Tambahkan beberapa perintah pada **constructor** kemudian buat **overloading** nya dengan parameter **Petugas**

```
public PetugasTambahFrame() {
    initComponents();
    status = SEDANG_TAMBAH;
    setLocationRelativeTo(null);
    eNamaPetugas.requestFocus();
}

public PetugasTambahFrame(Petugas petugas) {
    initComponents();
    status = SEDANG_UBAH;
    setLocationRelativeTo(null);
    eId.setText(String.valueOf(petugas.getId()));
    eNamaPetugas.setText(petugas.getNamaPetugas());
    eUsername.setText(petugas.getUsername());
    ePassword.setText(petugas.getPassword());
    eNamaPetugas.requestFocus();
}
```

Ketika **PetugasTambahFrame()** di-instance maka atribut status bernilai **SEDANG_TAMBAH** sehingga tombol **simpan** akan mengarah ke perintah **INSERT**, berbeda ketika **PetugasTambahFrame(petugas)** yang di-instance maka atribut status bernilai **SEDANG_UBAH** dan ketika tombol **simpan** ditekan akan mengarah kepada perintah **UPDATE** terhadap data yang sedang dipilih

Isi tombol **Batal** dengan perintah

```
private void bBatalActionPerformed(java.awt.event.ActionEvent evt) {
    dispose();
}
```

Tombol **Simpan**

```

private void bSimpanActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Koneksi koneksi = new Koneksi();
        Connection con = koneksi.getConnection();
        PreparedStatement ps;
        if(status==SEDANG_TAMBAH){
            String executeQuery = "insert into petugas "
                + "(nama_petugas,username,password) values (?,?,?)";
            ps = con.prepareStatement(executeQuery);
            ps.setString(1, eNamaPetugas.getText());
            ps.setString(2, eUsername.getText());
            ps.setString(3, ePassword.getText());
        }else{
            String executeQuery = "update petugas set "
                + "nama_petugas=?, username=?, password=? where id=?";
            ps = con.prepareStatement(executeQuery);
            ps.setString(1, eNamaPetugas.getText());
            ps.setString(2, eUsername.getText());
            ps.setString(3, ePassword.getText());
            ps.setString(4, eId.getText());
        }
        ps.executeUpdate();
    } catch (SQLException ex) {
        System.err.println(ex);
    }
    dispose();
}

```

Kembali ke **PenerbitTampilFrame()** tambahkan perintah untuk tombol **Tambah dan Ubah**

Berikut isi tombol **Tambah**

```

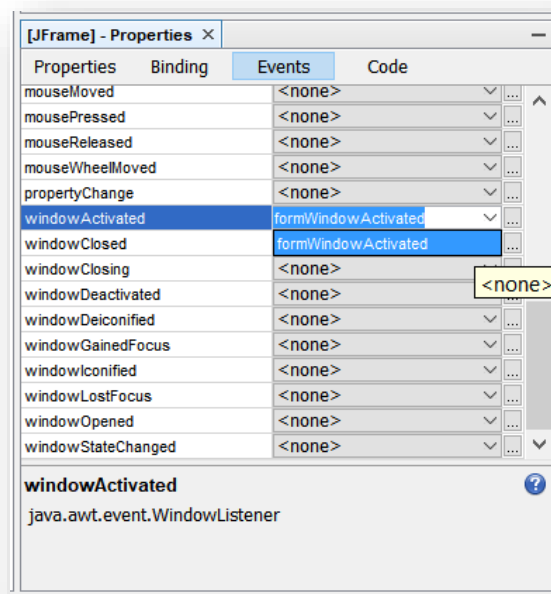
private void bTambahActionPerformed(java.awt.event.ActionEvent evt) {
    PetugasTambahFrame petugasTambahFrame = new PetugasTambahFrame();
    petugasTambahFrame.setVisible(true);
}

```

Berikut isi tombol **Ubah**

```
private void bUbahActionPerformed(java.awt.event.ActionEvent evt) {
    int i = tPetugas.getSelectedRow();
    if(i>=0){
        TableModel model = tPetugas.getModel();
        petugas = new Petugas();
        petugas.setId(Integer.parseInt(model.getValueAt(i, 0).toString()));
        petugas.setNamaPetugas(model.getValueAt(i, 1).toString());
        petugas.setUsername(model.getValueAt(i, 2).toString());
        petugas.setPassword(model.getValueAt(i, 3).toString());
        PetugasTambahFrame petugasTambahFrame = new PetugasTambahFrame(petugas);
        petugasTambahFrame.setVisible(true);
    }else{
        JOptionPane.showMessageDialog(null, "Pilih data yang ingin diubah");
    }
}
```

Sementara ini semua perintah sudah berfungsi tapi ketika data ditambah, diubah, atau dihapus, JTable tidak langsung berubah, hasil dapat dilihat dengan menekan tombol **Batal**, supaya ketika data disimpan JTable mengupdate data maka pada **PetugasTampilFrame** mode **Design** cari pada window **kanan bawah** dengan nama **JFrame Properties** pada tab **Events** scroll kebawah **windowActivated**



Isikan dengan perintah berikut

```
private void formWindowActivated(java.awt.event.WindowEvent evt) {  
    resetTable("");  
}
```

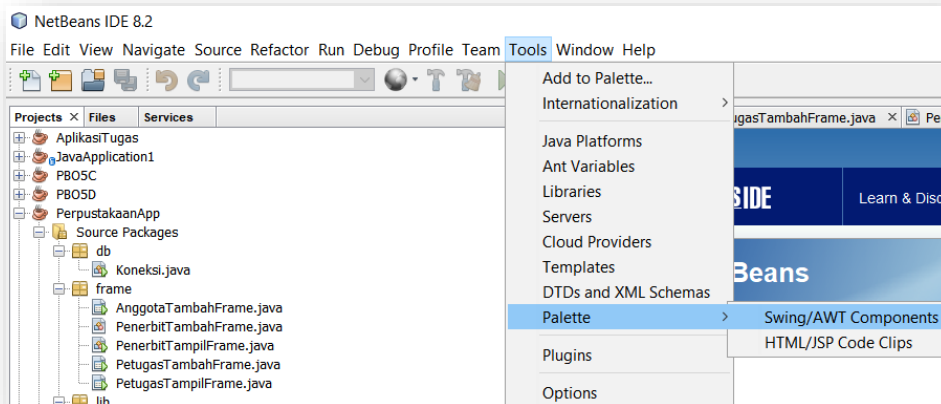
BAB 4

VISUAL BASED FRAME WITH VARIOUS INPUT

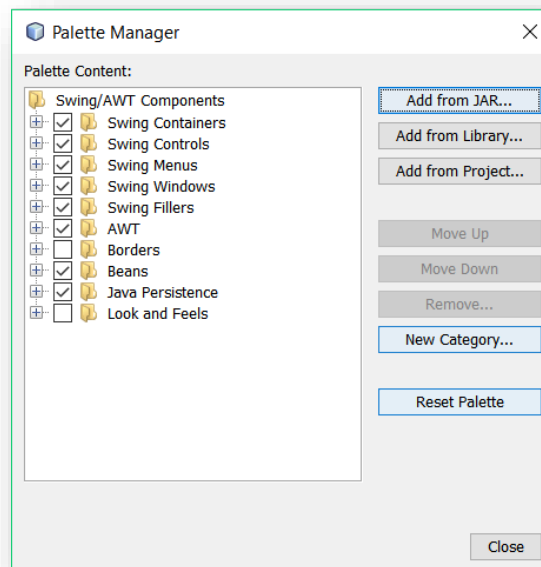
4.1 PERSIAPAN

Proses input tanggal bisa dikerjakan dengan JXDatePicker, namun komponen ini tidak tersedia secara default didalam NetBeans maka kita perlu panggil terlebih dahulu komponen tersebut.

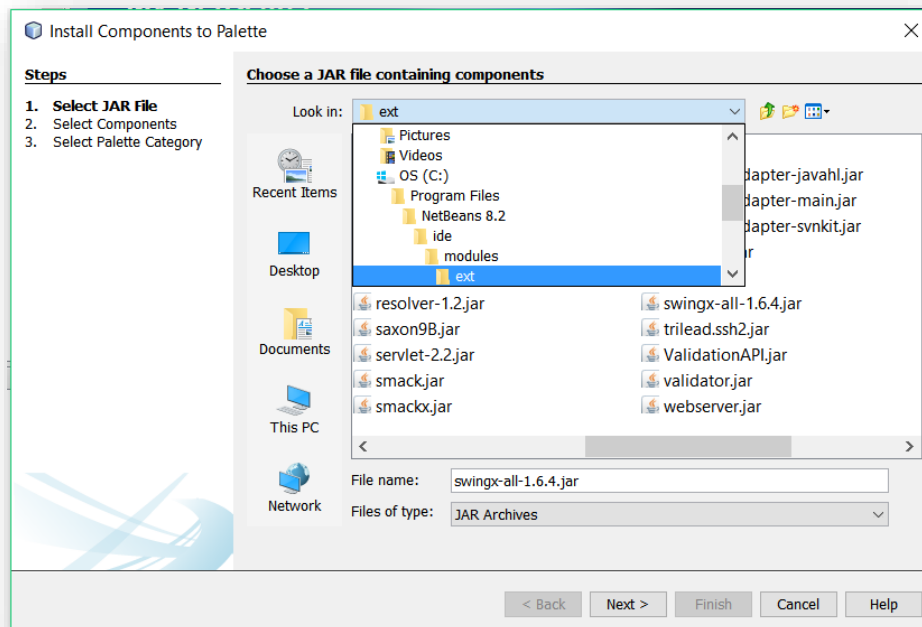
Pilih pada menu **Tools – Palette – Swing/AWT Components**



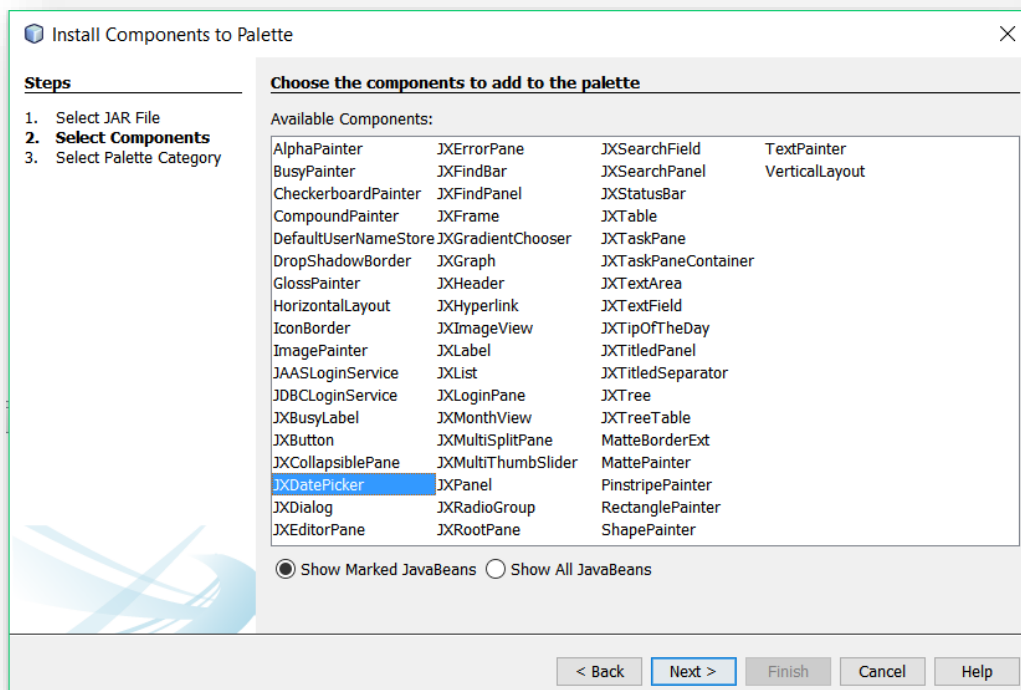
Kemudian pada window Palette Manager pilih **Add from JAR...**



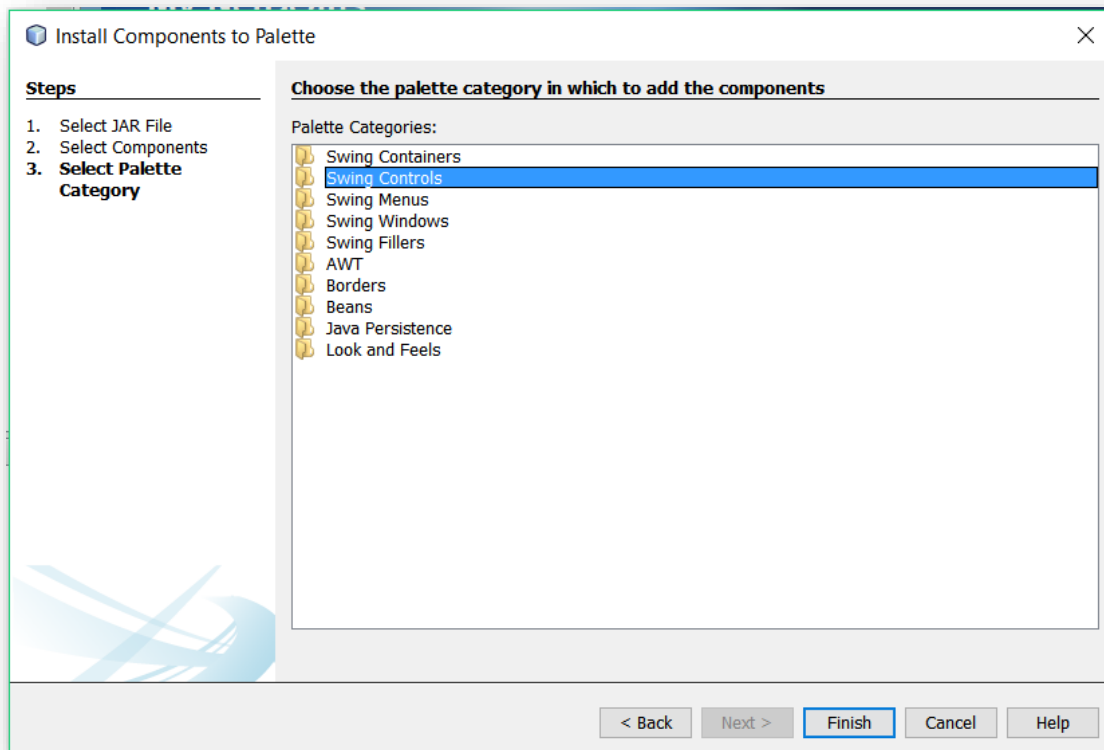
Pilih pada folder install NetBeans [NetBeans]/ide/modules/ext/swingx-all-x.x.x.jar



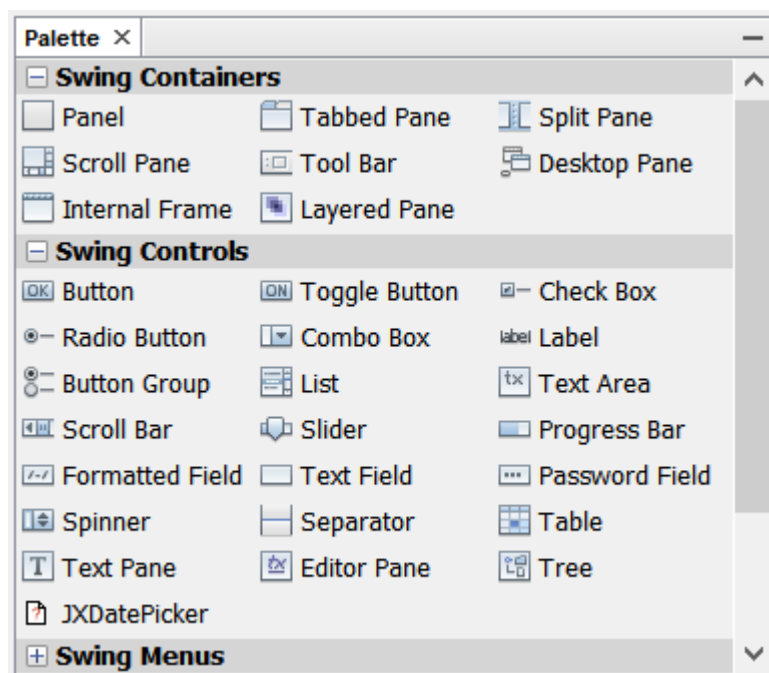
Pilih komponen JXDatePicker





Pilih letak Palette yang akan memunculkan komponen JXDatePicker



Perhatikan letak munculnya JXDatePicker



Nama	Jenis 	Panjang/Nilai 
<input type="text" value="id"/> <small>Pick from Central Columns</small>	<input type="text" value="VARCHAR"/> ▼	<input type="text" value="200"/>
<input type="text" value="nama_anggota"/> <small>Pick from Central Columns</small>	<input type="text" value="VARCHAR"/> ▼	<input type="text" value="200"/>
<input type="text" value="jenis_kelamin"/> <small>Pick from Central Columns</small>	<input type="text" value="VARCHAR"/> ▼	<input type="text" value="200"/>
<input type="text" value="tanggal_lahir"/> <small>Pick from Central Columns</small>	<input type="text" value="DATE"/> ▼	<input type="text"/>
<input type="text" value="agama"/> <small>Pick from Central Columns</small>	<input type="text" value="VARCHAR"/> ▼	<input type="text" value="200"/>
<input type="text" value="id_petugas"/> <small>Pick from Central Columns</small>	<input type="text" value="INT"/> ▼	<input type="text"/>
<input type="text" value="foto_anggota"/> <small>Pick from Central Columns</small>	<input type="text" value="BLOB"/> ▼	<input type="text"/>