

MODUL PEMBELAJARAN

TEORI DAN PRAKTIKUM

PROGRAMASI BERORIENTASI OBJEK

(OBJECT ORIENTED PROGRAMMING)

Disusun Oleh :

BEBEN SUTARA, S.Kom., M.T



SMK INFORMATIKA SUMEDANG

2014/2015

PENGANTAR

Puji dan syukur saya panjatkan kehadirat Allah SWT, yang mana atas berkat dan karunianya saya diberikan kesempatan untuk berbagi ilmu yang saya tuangkan ke dalam modul ini.

Ilmu pemrograman merupakan salah satu ciri dari kemampuan siswa dibidang informatika maka dari itu minimal siswa SMK INFORMATIKA SUMEDANG mampu dan paham salah satu bahasa pemrograman sebagai dasar keahliannya. Untuk itu modul ini bertujuan untuk menambah wawasan siswa baik itu untuk memahami dan mempraktekan konsep orientasi objek ke dalam bentuk pemrograman.

Saya mengucapkan banyak terima kasih kepada orang tua yang tidak hentinya selalu memberikan jasa baik berupa ucapan, materi, maupun perbuatan. Dan tidak lupa kepada pihak SMK INFORMATIKA SUMEDANG yang telah mempercayai saya untuk mengampu mata Pelajaran Pemrograman Berorientasi Objek sehingga dapat menambah wawasan saya dalam kontek tersebut.

Mudah-mudahan modul yang saya buat ini dapat bermanfaat bagi yang membacanya. Mohon maaf apabila ada kata-kata yang kurang tepat dan kurang dimengerti mohon dikoreksi. Akhir kata saya berungkapan “*pemrograman itu tidak sulit namun bagaimana kita menyikapi tentang itu, dalam kenyataannya sesuatu yang dianggap sulit belum tentu sulit, itu tergantung bagaimana kita menerimanya...*”.

Salam Sukses

BEBEN SUTARA.

Info :

Email : bebensutara@hotmail.co.id

HP : 085722112771

DAFTAR ISI

PENGANTAR	i
DAFTAR ISI	ii
MODUL 1 PEMROGRAMAN PROSEDURAL DAN OBJECT-ORIENTED PROGRAMMING (OOP)	1
MODUL 2 ISTILAH-ISTILAH DALAM OBJECT-ORIENTED PROGRAMMING	2
1. Class.....	2
2. Objek	2
3. Attributes	2
4. Behavior	3
MODUL 3 KONSEP-KONSEP DALAM OBJECT-ORIENTED PROGRAMMING	4
1. Enkapsulasi (<i>Encapsulation</i>)	4
2. Pewarisan (<i>Inheritance</i>)	4
3. Polimorfisme.....	5
MODUL 4 MENGENALAN NETBEANS IDE 7.4.....	6
1. Sofware Yang Diperlukan.....	6
2. Installasi Software	6
3. Konfigurasi JAVA_HOME, PATH, dan CLASSPATH	10
4. Membuat Project dan Class di Netbeans IDE 7.4.....	12
MODUL 5 KELAS DAN OBJEK	15
1. Class.....	15
2. Objek	15
3. Variabel Static dan Variabel Instan	16
4. Input Data	16
5. Latihan Satu	19
6. Latihan Dua	20
MODUL 6 ENCAPSULATION	22
1. Kendali Akses terhadap Kelas	22
2. Hak akses.....	22
3. Latihan.....	24
MODUL 7 CONSTRUCTOR DAN INHERITANCE	27
1. Construktur	27
2. Multiple Constructor	27
3. Inheritance.....	28
4. Latihan.....	30
MODUL 8 POLYMORPHISME.....	33
1. Pengertian Polymorphism	33
2. Latihan.....	34
MODUL 9 DATABASE	37
1. Installasi Xampp	37
2. Membuat Database	40
3. Membuat Tabel	41

MODUL 9 INTEGRASI DATABASE DENGAN JAVA	43
1. Membuat Library	43
2. Mendaftarkan Driver.....	44
3. Membuat Connection	46
MODUL 10 STATEMENT.....	48
1. Membuat Database Utilities	48
2. Statement Untuk Menyimpan Data Ke MySQL.....	51
3. Statement Untuk Mengubah Data Ke MySQL	54
4. Statement Untuk Menghapus Data Ke MySQL.....	55
5. Membuat Database Service	57
6. SQLInjection di Statement.....	63
MODUL 11 PREPARE STATEMENT	66
1. Membuat Prepare Statemen	66
2. Membuat Database Service Prepare	68
3. SQLInjection di Prepare Statement.....	73
MODUL 12 DATA ACCES OBJECT	75
1. Membuat Kelas Entity	75
2. Membuat Kelas Fasilitas	79
3. Menggunakan Service.....	83
MODUL 13 GUI	86
1. Membuat Project Aplikasi.....	86
2. Membuat Tampilan Aplikasi	88
3. Membuat Menampilkan Data dari MySQL ke Aplikasi	90
4. Menyimpan Data Dari Aplikasi ke MySQL	92
5. Mengubah Data Dari Aplikasi ke MySQL.....	96
6. Menghapus Data Dari Aplikasi ke MySQL.....	100
7. Keluar Dari Aplikasi	101
MODUL 14 MEMBUAT REPORT	102
1. Menginstall iReport-3.7.2	102
2. Menambahkan Library JasperReport.....	105
3. Membuat Fasilitas viewReport	106
4. Membuat Package dan Report.....	110
5. Menambahkan Tombol Lihat Daftar Barang	118
MODUL 15 PENDISTRIBUSIAN APLIKASI.....	121

MODUL 1

PEMROGRAMAN PROSEDURAL DAN OBJECT-ORIENTED PROGRAMMING (OOP)

Dalam dunia pemrograman, biasanya dihadapkan pada dua jenis metode pemrograman, yaitu :

1. pemrograman prosedural (*procedural*)
2. pemrograman berorientasi objek (*object oriented*).

Apa perbedaan kedua metode tersebut?

Bagaimana pemrograman berorientasi objek itu dilakukan?

Apa saja konsep-konsep OOP yang harus dikuasai?

Pemrograman prosedural merupakan suatu metode menulis program yang didasarkan pada "**serangkaian tugas yang diselesaikan dalam bentuk fungsi atau prosedur**". Cara pandang pemrograman prosedural yaitu sebuah program adalah suatu urutan instruksi. Programmer harus mem-break down suatu problem/masalah menjadi sub problem yang lebih sederhana. Fokus utama metode prosedural ini adalah fungsi dan prosedur, dimana keduanya digunakan untuk memanipulasi data.

Lain halnya dengan pemrograman berorientasi objek (OOP), fungsi dan data bukan menjadi dua hal yang terpisah. Fungsi dan data menjadi satu kesatuan yang disebut sebagai objek aktif. Cara pandang OOP ini yaitu sebuah program merupakan "**serangkaian objek yang bekerjasama untuk menyelesaikan suatu masalah**".

Dengan kata lain, metode prosedural berfokus pada cara komputer menangani tugas, sedangkan metode OOP berfokus pada tugas yang kita kembangkan untuk dieksekusi komputer. Kedua jenis metode pemrograman tersebut dapat digunakan untuk menangani masalah yang sama, asalkan bahasa pemrograman yang digunakan mendukung metode-metode tersebut.

MODUL 2

ISTILAH-ISTILAH DALAM OBJECT-ORIENTED PROGRAMMING (OOP)

1. Class

Definisi class yaitu template untuk membuat objek. Class merupakan *prototipe* atau *blue prints* yang mendefinisikan variabel-variabel dan method-method secara umum. Objek merupakan hasil instansiasi dari suatu class. Proses pembentukan objek dari suatu kelas disebut sebagai *instantiation*. Objek disebut juga sebagai *instances*.

Dalam bahasa teoritis OOP, class merupakan kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh kelas burung adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari burung. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek.

2. Objek

Untuk mempermudah pemahaman, maka disini akan dijelaskan melalui analogi. Pada dasarnya semua benda yang ada di dunia nyata dapat dianggap sebagai objek. Misalnya rumah, mobil, sepeda, motor, gelas, komputer, meja, sepatu, dll. Setiap objek memiliki atribut sebagai status (*state*) dan tingkah laku (*behavior*).

Contoh objek : Motor. Maka atribut (*state*) nya adalah pedal, roda, jaluji, speedometer, warna, jumlah roda, dll. Sedangkan tingkah laku (*behavior*) nya adalah kecepatan menaik, kecepatan menurun, perpindahan gigi motor, dll.

Analogi pemrograman berorientasi objek sama dengan penggambaran pada dunia nyata seperti contoh di atas. Dalam OOP, *state* disimpan pada variabel dan tingkah laku disimpan pada *method*.

Dalam bahasa teoritis OOP, Objek berfungsi untuk membungkus data dan fungsi bersama menjadi satu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

3. Attributes

Atribut adalah data yang membedakan antara objek satu dengan yang lainnya. Dalam class, atribut sering disebut sebagai variabel. Atribut dibedakan menjadi dua jenis yaitu *Instance Variable* dan *Class Variable*.

- *Instance variable* adalah atribut untuk tiap objek dari kelas yang sama. Tiap objek mempunyai dan menyimpan nilai atributnya sendiri. Jadi, tiap objek dari class yang sama boleh mempunyai nilai yang sama atau berbeda.

- Class Variable adalah atribut untuk semua objek yang dibuat dari class yang sama. Semua objek mempunyai nilai atribut yang sama. Jadi semua objek dari class yang sama mempunyai hanya satu nilai yang value nya sama.

Type	Meaning
boolean	Represents true/false values
byte	8-bit integer
char	Character
double	Double-precision floating point
float	Single-precision floating point
int	Integer
long	Long integer
short	Short integer

Table 2-1 Java's Built-in Primitive Data Types

Sumber : *Herbert Schildt - A Beginner's Guide, 3rd Edition* (2005;37)

4. Behavior

Behavior/tingkah laku adalah hal-hal yang bisa dilakukan oleh objek dari suatu class. Behavior dapat digunakan untuk mengubah nilai atribut suatu objek, menerima informasi dari objek lain, dan mengirim informasi ke objek lain untuk melakukan suatu tugas (*task*).

Dalam class, behavior disebut juga sebagai *methods*. *Methods* sendiri adalah serangkaian statements dalam suatu class yang menghandle suatu *task* tertentu. Cara objek berkomunikasi dengan objek yang lain adalah dengan menggunakan *method*.

MODUL 3

KONSEP-KONSEP DALAM OBJECT-ORIENTED PROGRAMMING (OOP)

1. Enkapsulasi (*Encapsulation*)

Definisi enkapsulasi : Pembungkusan variabel dan method dalam sebuah obyek yang terlindungi serta menyediakan interface untuk mengakses variabel tersebut. Variabel dan method yang dimiliki oleh suatu objek, bisa ditentukan hak aksesnya. Dalam OOP, konsep enkapsulasi sebenarnya merupakan perluasan dari struktur dalam bahasa java.

Contoh: jam tangan. Dalam hal ini, penting sekali untuk mengetahui waktu, sedangkan cara jam mencatat waktu dengan baik antara jam bertenaga baterai atau bertenaga gerak tidaklah penting kita ketahui.

Dengan kata lain enkapsulasi berfungsi untuk memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam/dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses *interface* yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

2. Pewarisan (*Inheritance*)

Pewarisan merupakan pewarisan atribut dan method dari sebuah class ke class lainnya. Class yang mewarisi disebut superclass dan Class yang diwarisi disebut subclass. Subclass bisa berlaku sebagai superclass bagi class lainnya, disebut sebagai multilevel inheritance.

Contoh : terdapat burung dan burung merpati. Burung termasuk superclass. Burung merpati termasuk subclass. Hal ini dikarenakan burung merpati memiliki variabel dan method yang dimiliki oleh burung.

Prinsip dasar inheritance yaitu persamaan-persamaan yang dimiliki oleh beberapa kelas dapat digabungkan dalam sebuah class induk sehingga setiap kelas yang diturunkannya memuat hal-hal yang spesifik untuk kelas yang bersangkutan.

Keuntungan Pewarisan :

- Subclass menyediakan *state/behaviour* yang spesifik yang membedakan dengan superclass, sehingga memungkinkan programmer untuk menggunakan ulang source code dari superclass yang telah ada.
- Programmer dapat mendefinisikan superclass khusus yang bersifat generik, yang disebut abstract class (abstraksi), untuk mendefinisikan class dengan tingkah laku dan state secara umum.

3. Polimorfisme

Polimorfisme adalah kemampuan suatu obyek untuk mempunyai lebih dari satu bentuk. *Polimorfisme* tidak bergantung kepada pemanggilan subrutin. Metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesan tersebut dikirim. Contohnya, bila sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tunggal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

Catatan :

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah perusahaan yang memiliki manager, sekretaris, petugas administrasi, dan lainnya. Misal manager tersebut ingin memperoleh data dari bagian administrasi, maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bagian administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah, dapat dilakukan dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri.

MODUL 4

PENGENALAN NETBEANS IDE 7.4

1. Software Yang Diperlukan

Software yang dibutuhkan dalam melakukan praktikum Object Oriented Programming, sebagai berikut :

- Software Development Kit (J2SDK) (versi bebas sesuaikan dengan kebutuhan)
- Netbeans IDE 7.4 (versi bebas sesuaikan dengan kebutuhan)

2. Installasi Software

Untuk melakukan instalasi lakukan installasi **Software Development Kit (J2SDK)** terlebih dahulu, langkahnya sebagai berikut :

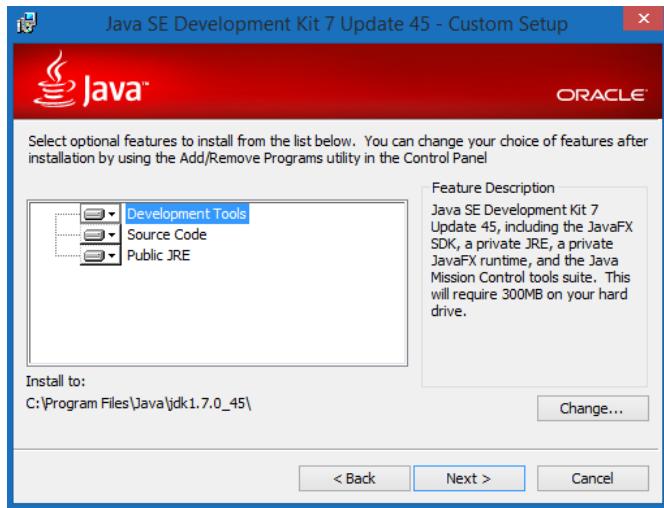
- 1) Double klik software **jdk-7u45-windows-i586.exe** didalam folder yang telah disediakan. Maka akan muncul tampilan seperti dibawah ini.



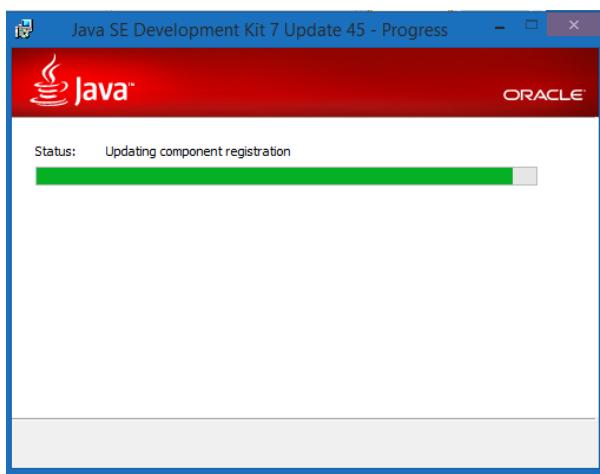
- 2) Kemudian akan muncul tampilan seperti dibawah ini yang harus diperhatikan Install to : **C:\Program Files\Java\jdk1.7.0_45** karena alamat itu nantinya akan dimasukkan dalam konfigurasi JAVA_HOME. klik **Next**

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



- 3) Tunggu sampai installasi selesai. Seperti tampilan dibawah ini.



- 4) Ketika muncul tampilan seperti dibawah ini maka installasi sudah selesai klik tombol **Close**

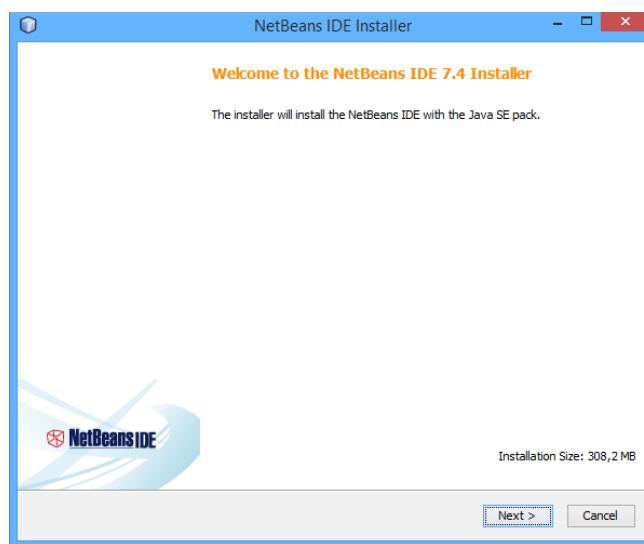
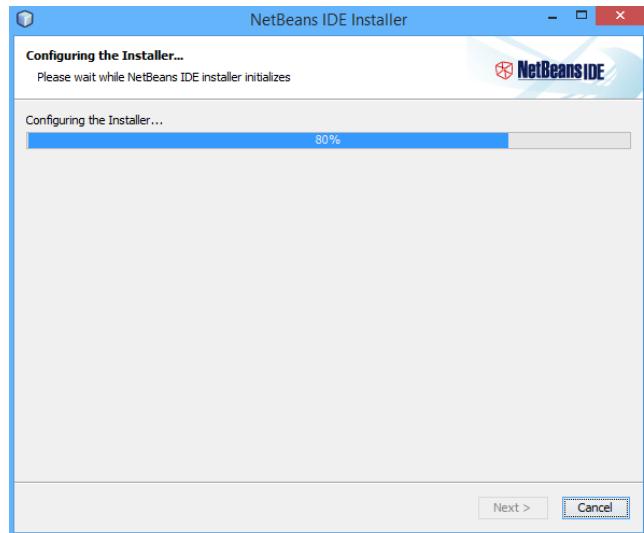


Sedangkan untuk installasi **Netbeans IDE 7.4** sebagai berikut :

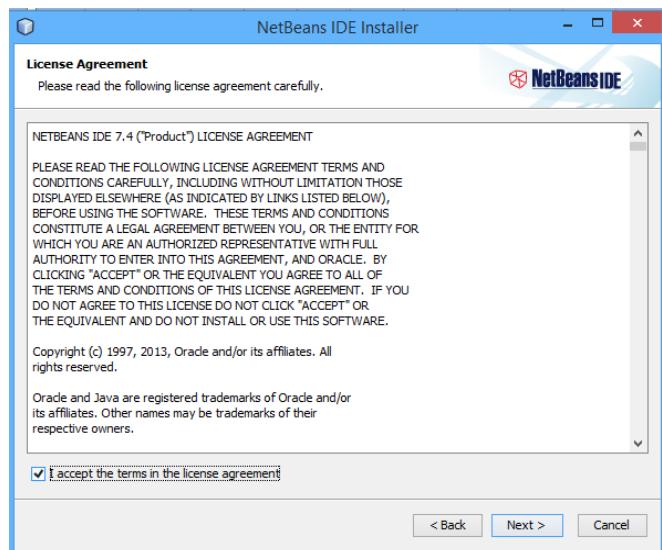
- 1) Double klik software **netbeans-7.4-javase-windows.exe** didalam folder yang telah disediakan.
Maka akan muncul tampilan seperti dibawah ini.

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

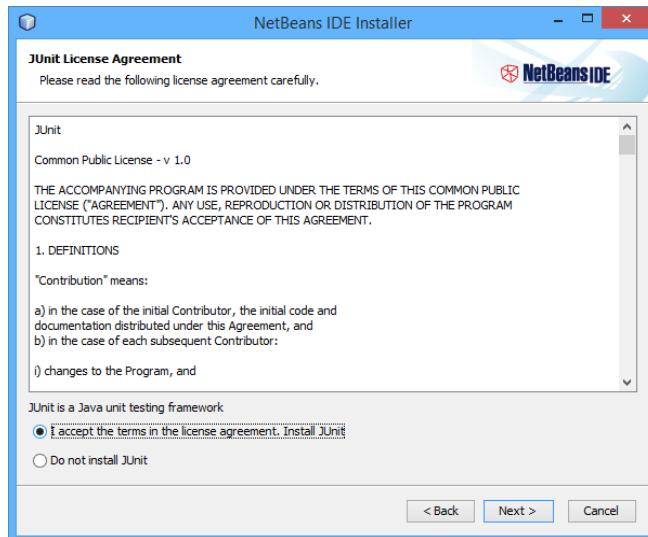


- 2) Klik **Next** untuk melanjutkan installasi. Ceklis **I accept the terms in the licence agreement**, klik **Next**

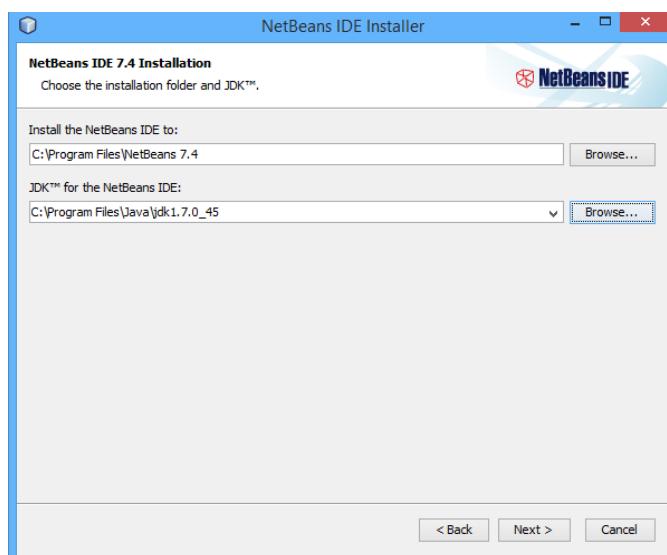


Object Oriented Programming (OOP)

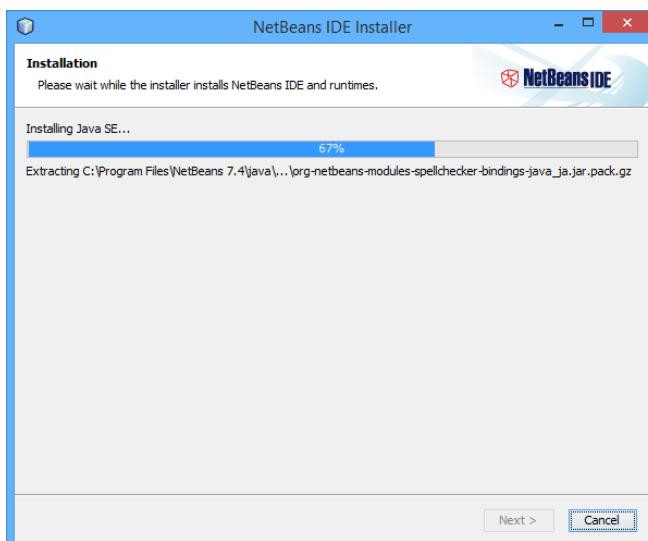
Guru : BEBEN SUTARA, S.Kom., M.T

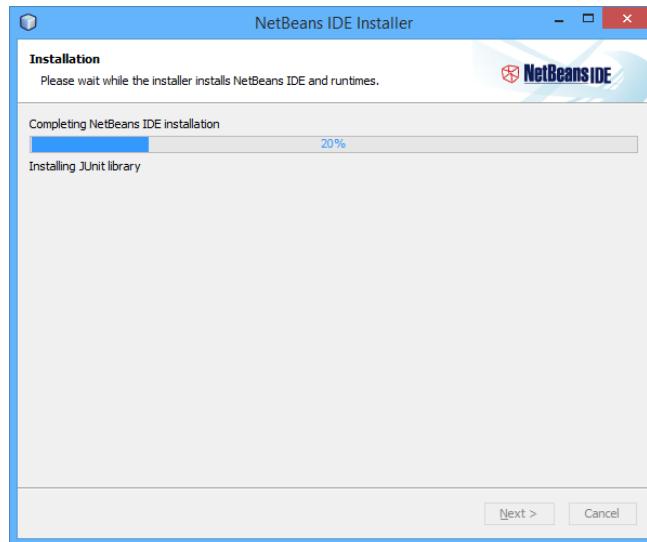


- 3) Penentuan alamat installasi karena berhubungan dengan Java maka akan muncul dua alamat installasi yaitu Netbean dan java yang sebelumnya sudah diinstall. Klik **Next**

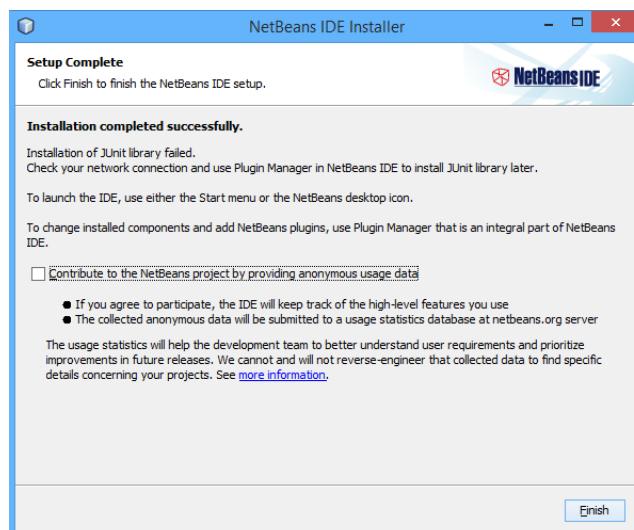


- 4) Akan muncul konfirmasi update hilangkan Cheklisnya, klik **Next**, akan muncul tampilan seperti dibawah tunggu sampai installasi selesai.





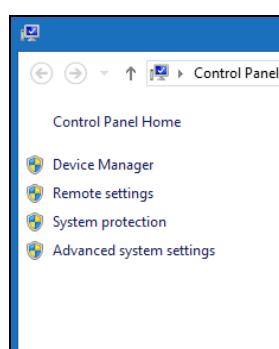
- 5) Tampilan dibawah merupakan akhir dari installasi Netbean. Klik **Finish**



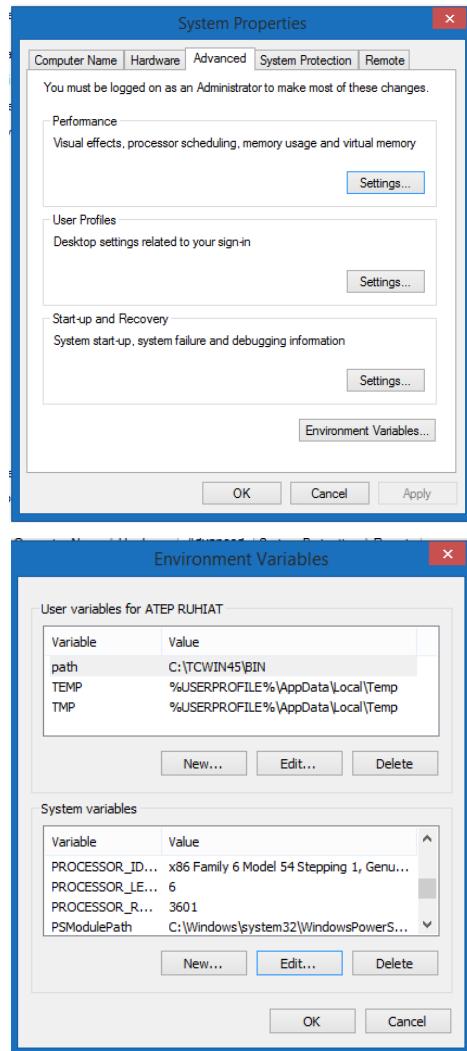
3. Konfigurasi JAVA_HOME, PATH, dan CLASSPATH

Setelah installasi selesai selanjutnya yaitu konfigurasi JAVA_HOME, PATH, dan CLASSPATH

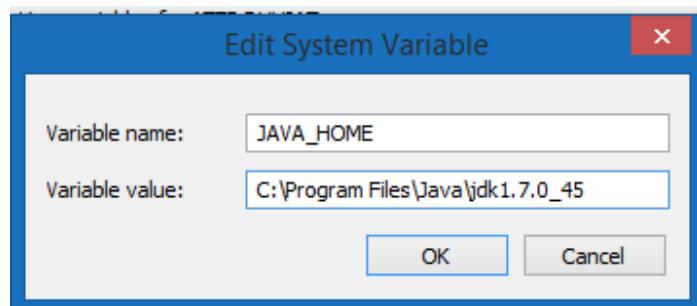
- 1) Klik kanan My Computer/This PC pilih **Advanced system settings**.



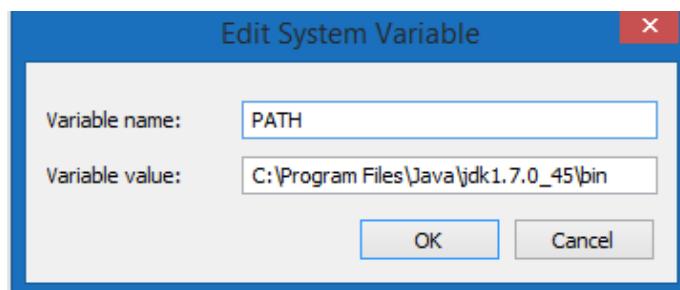
- 2) Akan muncul tampilan **System Properties** seperti dibawah ini pilih **Environment Variables**.



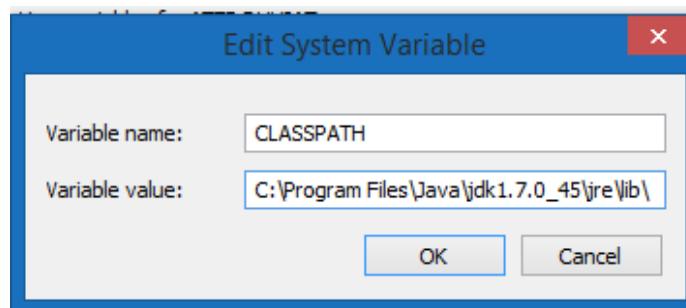
- 3) Buatlah konfigurasi baru pada **System variables** dengan mengklik tombol **New**. Untuk Variable name : **JAVA_HOME** dan Variable value : **C:\Program Files\Java\jdk1.7.0_45**. Klik **OK**.



- 4) Untuk konfigurasi PATH lihat terlebih dahulu pada **System variables** sudah ada atau belum. Jika belum maka ulangi langkah diatas, jika sudah maka tinggal double klik **PATH** kemudian lihat Variable value sudah terisi atau belum. Jika belum maka tinggal ketikkan alamat **C:\Program Files\Java\jdk1.7.0_51\bin**, jika sudah maka tinggal tambahkan dengan mengetik tanda **(;)** kemudian ketik alamatnya seperti ini ;**C:\Program Files\Java\jdk1.7.0_51\bin**
Pada setting PATH dimaksudkan agar file-file executable di **C:\Program Files\Java\jdk1.7.0_45\bin** bisa dijalankan dari seluruh direktori kerja.



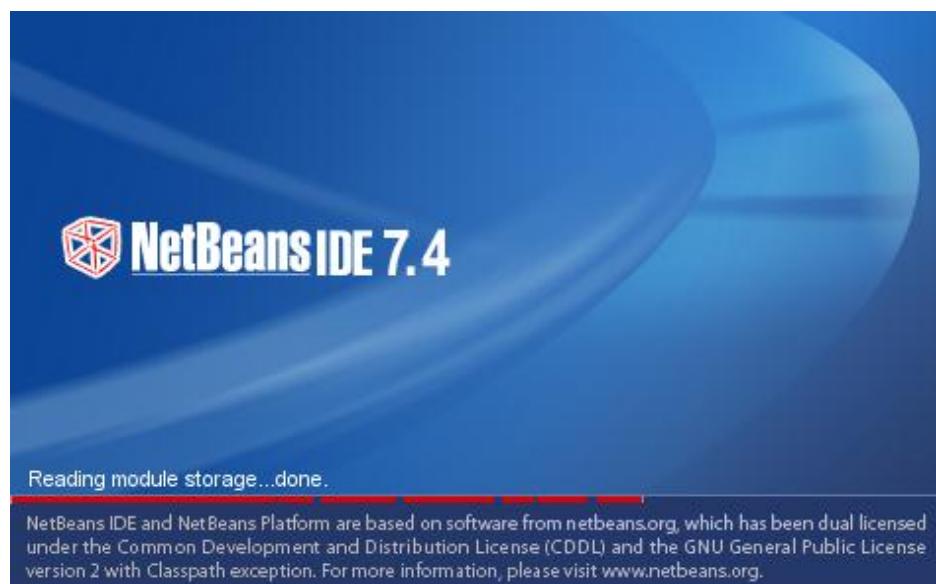
- 5) Untuk konfigurasi CLASSPATH juga hampir sama dengan yang diatas namun untuk value-nya
C:\Program Files\Java\jdk1.7.0_45\jre\lib
Pada setting CLASSPATH dimaksudkan agar class-class yang ada di **C:\Program Files\Java\jdk1.7.0_45\jre\lib** bisa diimport dari seluruh direktori kerja.



4. Membuat Project dan Class di Netbeans IDE 7.4

Untuk membuat project/paket dan class pada Netbeans IDE 7.4 caranya sebagai berikut :

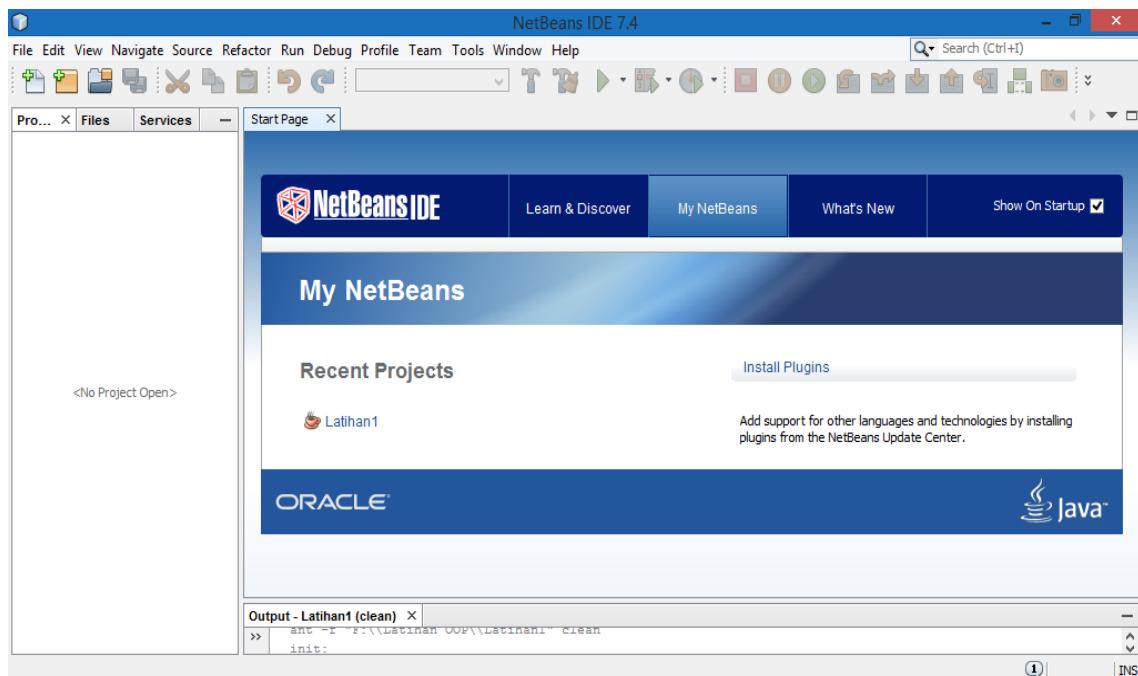
- 1) Double klik **NetBeans IDE 7.4.exe** baik itu pada taskbar, desktop, atau all programs. Maka akan muncul jendela pembuka seperti dibawah ini.



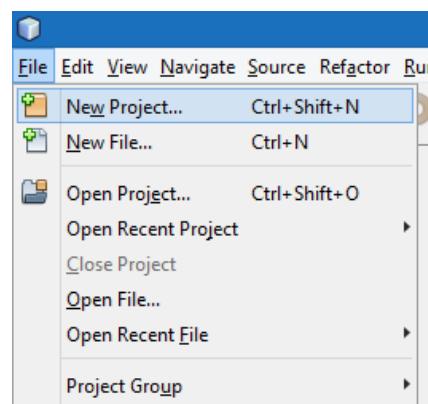
Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

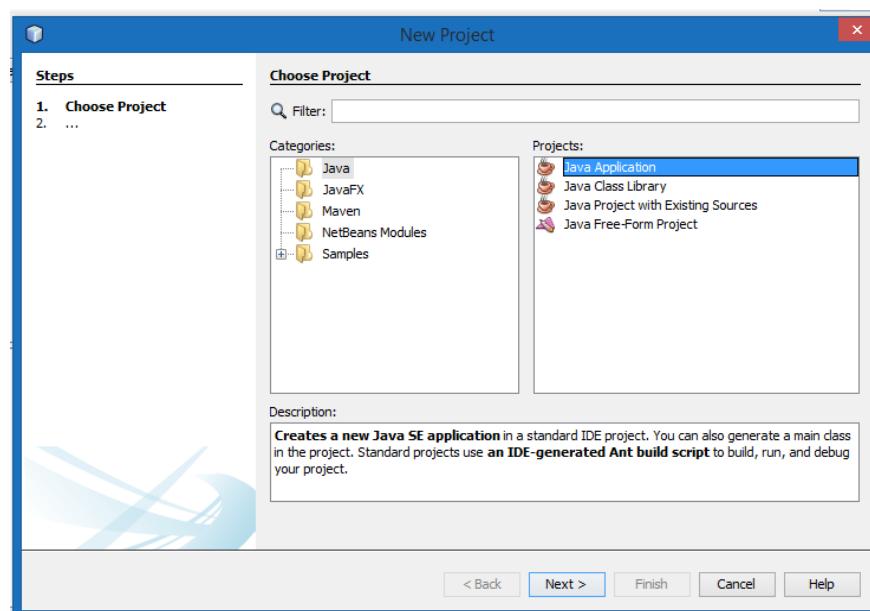
- 2) Akan muncul area kerja pad NetBeans IDE 7.4. Close Start Pagenya.



- 3) Untuk membuat Project pada Netbeans caranya pilih menu **File – New Project**. kemudian klik.



- 4) Pilih tipe file yang akan digunakan Categories **Java** untuk Projects **Java Application**, klik **Next**.

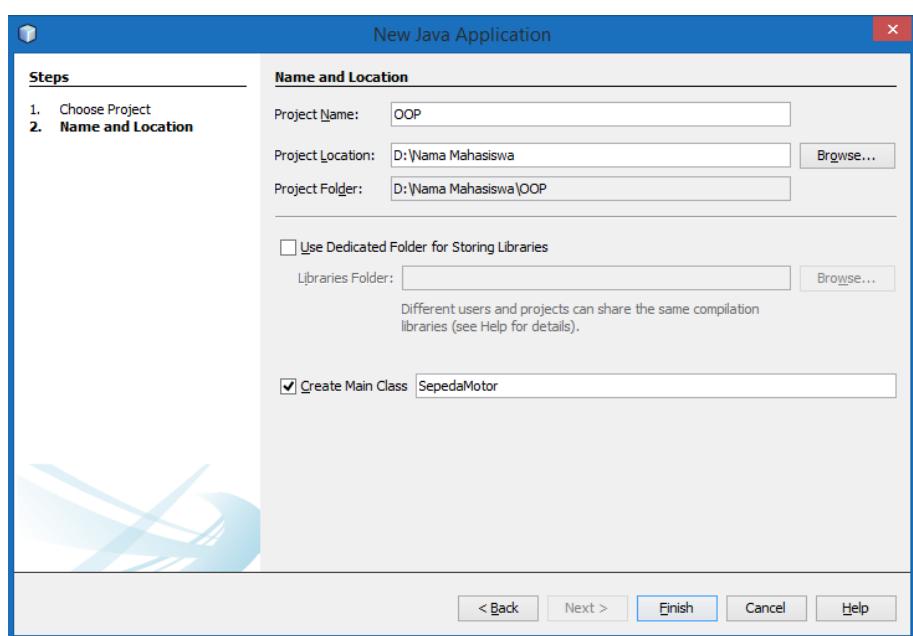


Object Oriented Programming (OOP)

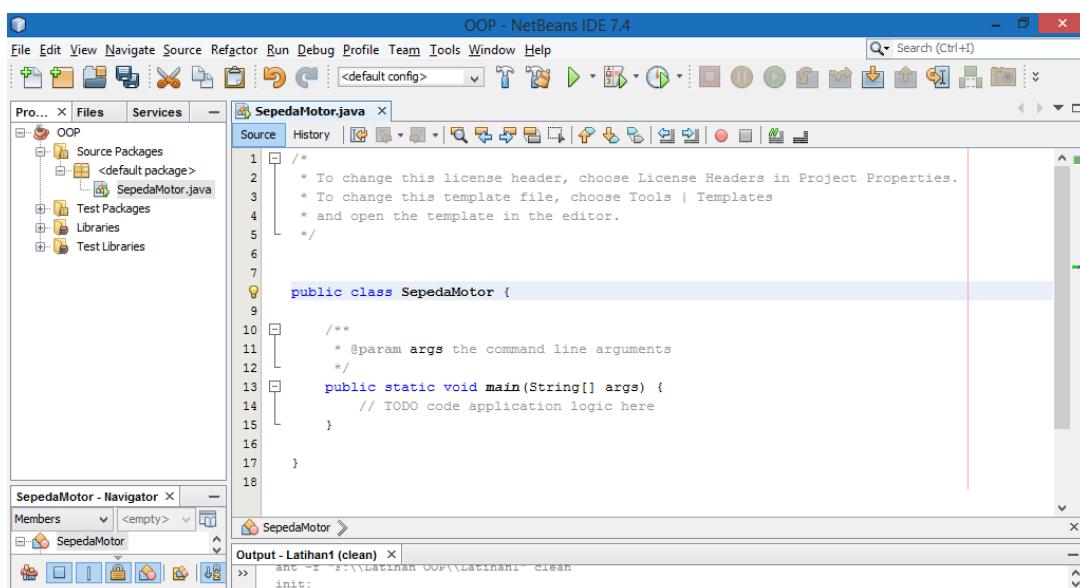
Guru : BEBEN SUTARA, S.Kom., M.T

- 5) Langkah selanjutnya yaitu nama dan lokasi.

Isian	Nilai	Keterangan
Project Name	OOP	Untuk memberikan nama project atau paket.
Project Location	D:\Nama Siswa	Alamat penyimpanan yang telah dibuat sebelumnya diberi nama masing-masing siswa dan kelas .
Project Folder	D:\Nama Siswa\OOP	Akan menghasilkan dalam folder tersebut yaitu folder OOP
Create Main Class	SepedaMotor	Sebagai inisiasi class awal super class yaitu SepedaMotor (tanpa space)



- 6) Klik **Finish**, akan menghasilkan tampilan sebagai berikut :



MODUL 5

KELAS DAN OBJEK

1. Class

Class merupakan cetak biru (blue print) dari objek atau dengan kata lain sebuah class menggambarkan ciri-ciri objek secara umum.

Sebagai contoh Yamaha MX, Suzuki Smash, Honda Supra Fit, dan Kawasaki Ninja merupakan objek dari class sepeda motor. Objek-objek tersebut mempunyai kesamaan atribut (merk, tipe, berat, kapasitas bensin, tipe mesin, warna, harga, dll) dan method untuk mengakses data pada atributnya (misal fungsi untuk menginput data merk, tipe, berat, dsb serta fungsi untuk mencetak data merk, tipe, berat, dsb)

Contoh :

```
class SepedaMotor{  
    private String merk, tipe;  
    private int tangki;  
    private long harga;  
    public void inputMerk (String merk){  
        this.merk = merk;  
    }  
    public String tampilMerk(){  
        return merk;  
    }  
}
```

2. Objek

Objek merupakan segala sesuatu yang ada di dunia ini, yaitu manusia, hewan, tumbuhan, rumah, kendaraan, dan lain sebagainya. Contoh objek yang telah disebutkan diatas merupakan contoh objek nyata pada kehidupan kita. Pada pemrograman berorientasi objek, kita akan belajar bagaimana membawa konsep objek dalam kehidupan nyata menjadi objek dalam dunia pemrograman.

Setiap objek dalam dunia nyata pasti memiliki 2 elemen penyusunnya yaitu keadaan (state) dan perilaku/sifat (behaviour).

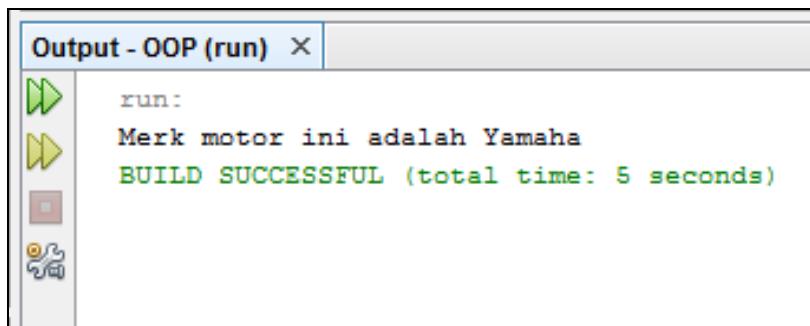
Sintak untuk membuat objek :

```
namaClass namaObjek = new namaClass ()
```

Class utama dari program :

```
public class Yamaha {  
    public static void main(String[] args) {  
        SepedaMotor motor = new SepedaMotor();  
        motor.inputMerk("Yamaha");  
        System.out.println("Merk motor ini adalah "+motor.tampilMerk());  
    }  
}
```

HASIL :



```
Output - OOP (run) X  
run:  
Merk motor ini adalah Yamaha  
BUILD SUCCESSFUL (total time: 5 seconds)
```

3. Variabel Static dan Variabel Instan

Variabel static adalah variabel yang didefinisikan/dideklarasikan dengan menggunakan keyword "static", contoh :

```
static int jumlah ;
```

Variabel-variabel yang dideklarasikan dengan tidak menggunakan keyword "static", maka variabel tersebut disebut dengan variabel instant.

Jika sebuah variabel merupakan variabel instant, maka masing-masing objek dari class tersebut akan memiliki variabel yang sama dengan variable instant di satu objek tidak akan berpengaruh pada variabel instant di objek yang berbeda.

Jika sebuah variabel merupakan variabel static (pada suatu class), maka variabel static tersebut adalah variabel yang sama di semua objek dari class tersebut. Sehingga perubahan nilai pada variabel static tersebut di suatu objek akan berpengaruh juga terhadap objek yang lainnya.

4. Input Data

Untuk menginputkan data dari keyboard ada 2 cara, yaitu :

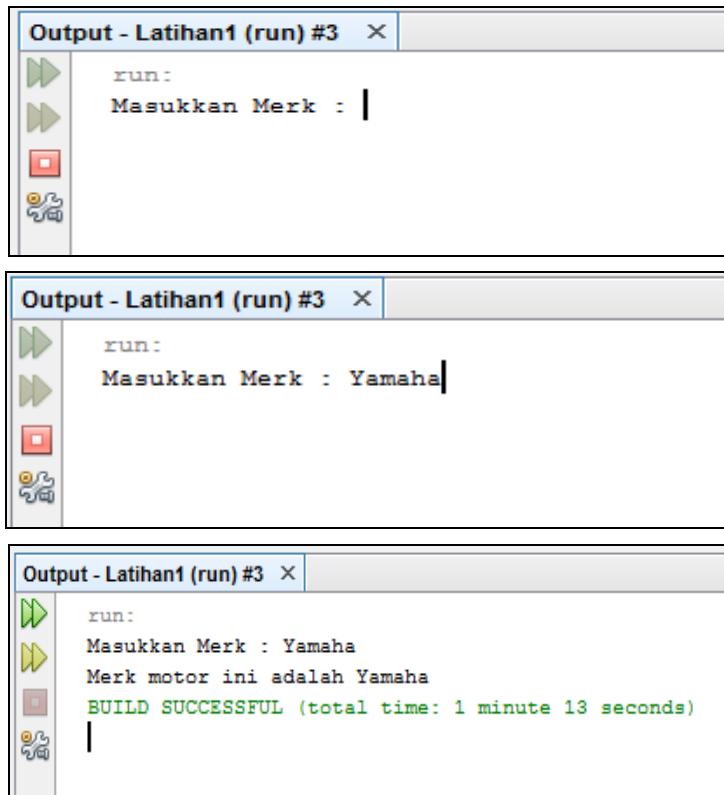
- Input dari mode console yaitu dengan memanfaatkan class **BufferedReader** dan **InputStreamReader**.

Untuk dapat mengakses class **BufferedReader** maka perlu mengimpor dari package **java.io.*** dan menambahkan statemen **throws IOException** pada header method main.

Contoh :

```
import java.io.*;  
  
public class InputMerk1 {  
    public static void main(String[] args) throws IOException{  
        BufferedReader input = new BufferedReader (  
            new InputStreamReader(System.in));  
        String merk;  
        System.out.print("Masukkan Merk : ");  
        merk = input.readLine();  
        System.out.println("Merk motor ini adalah " + merk);  
    }  
}
```

HASIL :



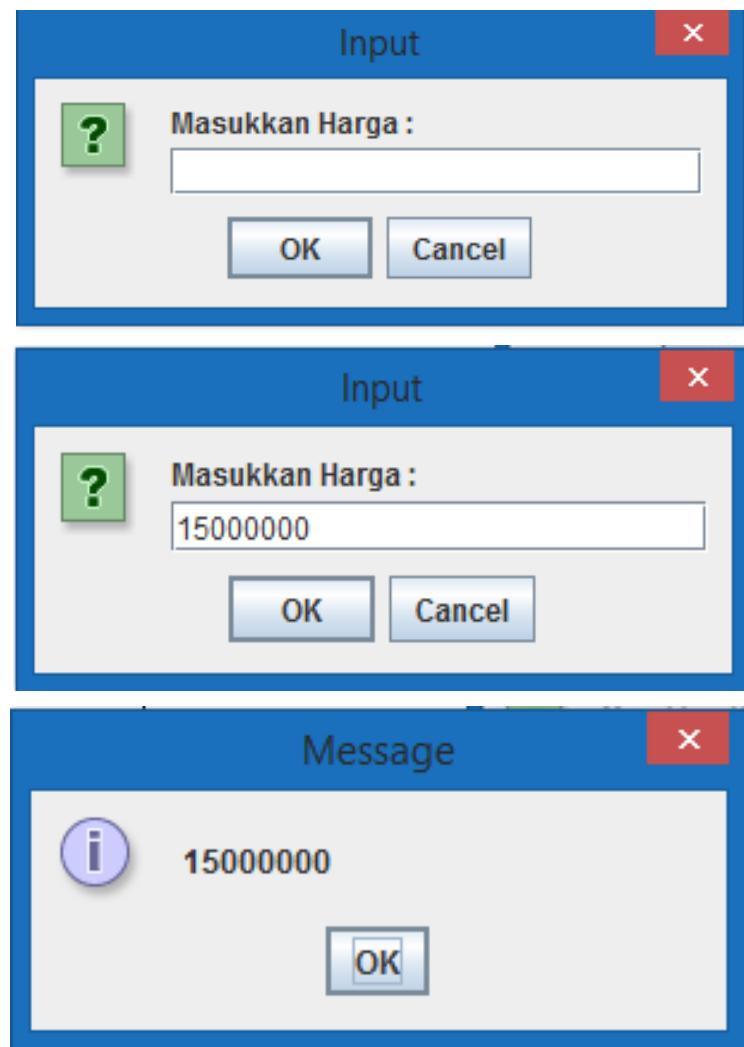
- Inputan dengan memanfaatkan class **JOptionPane**.

Untuk bisa menggunakan class **JOptionPane**, maka perlu mengimport dari package **javax.swing.*** dan gunakan method **showInputDialog()** yang terdapat pada class **JOptionPane**.

Contoh :

```
import javax.swing.*;  
  
public class InputMerk2 {  
    public static void main(String[] args) {  
        String input;  
        input = JOptionPane.showInputDialog("Masukkan Harga : ");  
        long harga = Integer.valueOf(input).intValue();  
        JOptionPane.showMessageDialog(null, harga);  
    }  
}
```

HASIL :



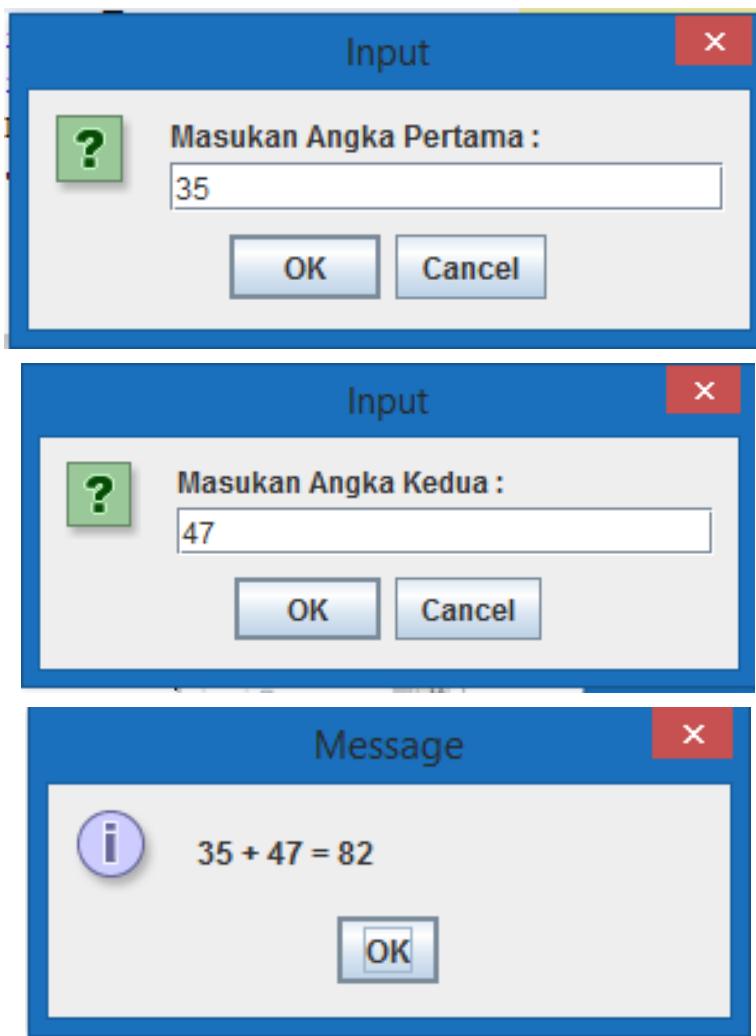
Catatan :

Wrapper	Conversion Method
Double	static double parseDouble(String str) throws NumberFormatException
Float	static float parseFloat(String str) throws NumberFormatException
Long	static long parseLong(String str) throws NumberFormatException
Integer	static int parseInt(String str) throws NumberFormatException
Short	static short parseShort(String str) throws NumberFormatException
Byte	static byte parseByte(String str) throws NumberFormatException

Sumber : *Herbert Schildt - A Beginner's Guide, 3rd Edition* (2005;396-397)

5. Latihan Satu

Buatlah program pertambahan sederhana seperti dibawah ini.



KODE PROGRAM :

6. Latihan Dua

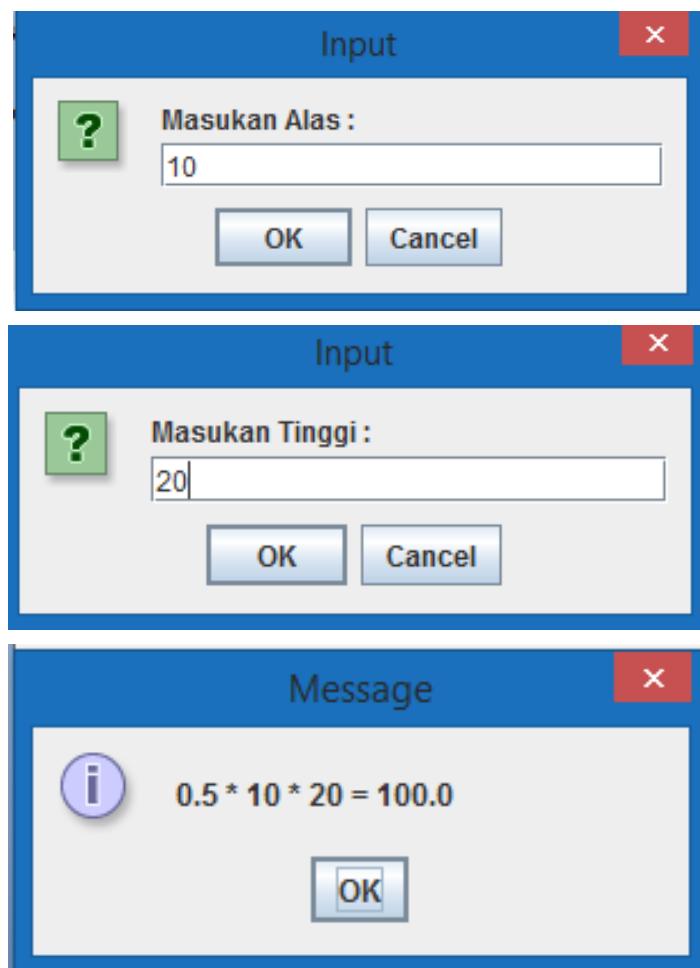
Buatlah kelas baru dengan nama segitiga kemudian lengkapi kode program berikut :

```
import javax.swing.*;
class segitiga {
String alas,tinggi;
public void input(){
    ....
    ....
}
public double hitung(){
    long a = Long.parseLong(alas);
    long t = Long.parseLong(tinggi);
    return (0.5*a*t);
}
```

```
public void hasil(){
    JOptionPane.showMessageDialog(null,"0.5 * " + .... + " * " + ....
+ " = " + .... );
}
}

class luas_segitiga{
public static void main(String[] args) {
    ....
    ....
    ....
    ....
}
}
```

HASIL :



MODUL 6

ENCAPSULATION

Salah satu keistimewaan java adalah pengkapsulan. Pengkapsulan adalah mengkombinasikan suatu struktur dengan fungsi dalam satu kerangka yaitu kelas(*class*).

Kelas akan menutup rapat baik data maupun kode. Akses item di dalam kelas dikendalikan. Pengendalian ini tidak hanya berupa data tetapi juga kode. Saat kelas akan digunakan, kelas harus sudah dideklarasikan. Yang penting, pemakai kelas mengetahui deskripsi kelas, tetapi bukan implementasinya. Bagi pemakai, detail internal kelas tidak penting. Konsep ini disebut penyembunyian informasi (*information hiding*).

Untuk menggunakan kelas, kita perlu mengetahui sesuatu tentangnya. Kita perlu mengetahui fungsi apa yang bisa digunakan dan data apa saja yang dapat diakses. Fungsi yang dapat digunakan dan data yang dapat diakses disebut antarmuka pemakai (*userinterface*). Antarmuka pemakai menceritakan bagaimana kelas berperilaku, bukan bagaimana kelas dibuat. Kita tidak perlu mengetahui implementasi kelas. Sekali kelas dibuat, kita bisa memakainya berulang-ulang. Bagi pandangan pemakai, kelas adalah kotak hitam dengan perilaku tertentu.

1. Kendali Akses terhadap Kelas

Tugas kelas adalah untuk menyembunyikan informasi yang tidak diperlukan oleh pemakai. Ada tiga macam pemakai kelas:

- kelas itu sendiri
- pemakai umum
- kelas turunan

Setiap macam pemakai mempunyai hak aksesnya masing-masing.

2. Hak akses

Kelas pada java menawarkan tiga hak akses untuk anggota kelas (baik anggota data maupun fungsi anggota), yaitu :

- Private

Tingkat akses ini berguna untuk memberikan hak akses data hanya kepada kelas yang bersangkutan saja. Artinya kelas-kelas turunan ataupun lingkungan luar di dalam program tidak diizinkan untuk mengakses data tersebut. Dalam java untuk menentukan data tersebut bersifat privasi, maka harus menggunakan kata kunci **private**. Secara default, jika tidak menuliskan tingkat akses dalam pendeklarasian data dalam sebuah kelas maka data-datanya akan dianggap data private.

- Public

Tingkat akses ini dapat diakses oleh fungsi anggota kelas itu sendiri, lingkungan luar dan fungsi anggota kelas turunan. Suatu kelas agar bisa diakses dari luar kelas, misalnya dalam fungsi main(), perlu mempunyai hak akses publik. Hak akses ini yang biasanya digunakan sebagai perantara antara

kelas dengan dunia luar. Dalam java untuk menentukan data tersebut bersifat umum, maka harus menggunakan kata kunci **public**.

- Protected

Tingkat akses ini digunakan untuk memberikan hak akses terhadap data dalam suatu kelas sehingga data tersebut dapat diakses oleh kelas turunannya, namun lingkungan luar di dalam program masih tetap tidak diberi hak untuk mengaksesnya. Dalam java untuk menentukan data tersebut bersifat diproteksi, maka harus menggunakan kata kunci **protected**.

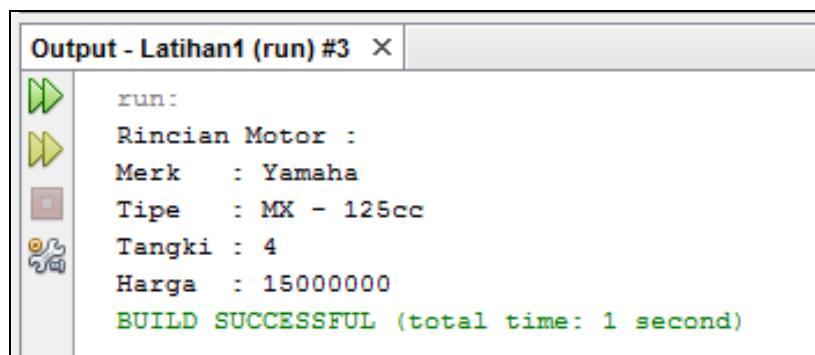
Contoh :

```
class SepedaMotor{  
    private String merk, tipe;  
    private int tangki;  
    private long harga;  
    public void inputMerk (String merk){  
        this.merk = merk;  
    }  
    public void inputTipe (String tipe){  
        this.tipe = tipe;  
    }  
    public void inputTangki(int tangki){  
        this.tangki = tangki;  
    }  
    public void inputHarga (long harga){  
        this.harga = harga;  
    }  
    public String tampilMerk(){  
        return merk;  
    }  
    public String tampilTipe(){  
        return tipe;  
    }  
    public int tampilTangki(){  
        return tangki;  
    }  
    public long tampilHarga(){
```

```
        return harga;
    }

    public static void main(String[] args) {
        SepedaMotor motor = new SepedaMotor();
        motor.inputMerk("Yamaha");
        motor.inputTipe("MX - 125cc");
        motor.inputTangki(4);
        motor.inputHarga(15000000);
        System.out.println("Rincian Motor :");
        System.out.println("Merk    : " + motor.tampilMerk());
        System.out.println("Tipe    : " + motor.tampilTipe());
        System.out.println("Tangki : " + motor.tampilTangki());
        System.out.println("Harga   : " + motor.tampilHarga());
    }
}
```

HASIL :

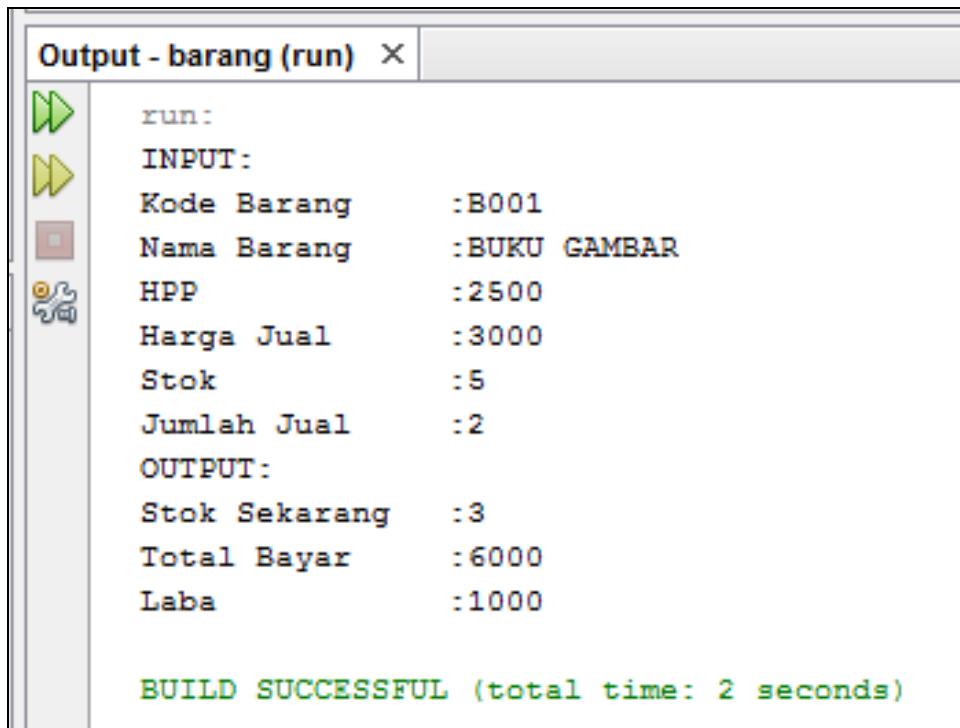


```
Output - Latihan1 (run) #3 ×
run:
Rincian Motor :
Merk    : Yamaha
Tipe    : MX - 125cc
Tangki : 4
Harga   : 15000000
BUILD SUCCESSFUL (total time: 1 second)
```

3. Latihan

Implementasikan encapsulation kedalam kelas barang, dengan jenis barang yang diambil adalah buku gambar. Yang mana memiliki Kode, Nama, HPP, Harga Jual, dan Stok. Ketika dilakukan transaksi penjualan terdapat atribut lain yaitu Jumlah Jual. Yang diminta pada kasus ini adalah memunculkan Stok Sekarang, Total Bayar, dan Laba. Proses yang dibutuhkan yaitu input, hitung stok, hitung total, hitung laba, dan tampil.

HASIL :



```
Output - barang (run) X
run:
INPUT:
Kode Barang      :B001
Nama Barang      :BUKU GAMBAR
HPP              :2500
Harga Jual       :3000
Stok              :5
Jumlah Jual      :2
OUTPUT:
Stok Sekarang    :3
Total Bayar       :6000
Laba              :1000

BUILD SUCCESSFUL (total time: 2 seconds)
```

KODE PROGRAM :

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

MODUL 7

CONSTRUCTOR DAN INHERITANCE

1. Konstruktor

Constructor adalah method yang secara otomatis dipanggil/dijalankan pada saat sebuah class diinisiasi. Jika dalam sebuah class tidak terdapat constructor maka secara otomatis Java akan memuatkan sebuah default constructor. Nama constructor harus sama dengan nama class dan tidak boleh memiliki tipe *return value*. Sama halnya dengan method, constructor dapat memiliki satu atau banyak parameter maupun tanpa parameter.

Constructor biasanya digunakan untuk memberi nilai awal dari atribut-atribut class tersebut.

Contoh :

```
class Login {  
    private String user, pass;  
    Login() {  
        this.user = "";  
        this.pass = "";  
    }  
}
```

2. Multiple Constructor

Java tidak membatasi jumlah constructor dalam satu class, sehingga memungkinkan sebuah class memiliki lebih dari satu constructor. Multiple constructor adalah adanya lebih dari satu constructor untuk sebuah class. Yang membedakan antara satu constructor dengan constructor lainnya adalah pada parameternya (nama constructornya sama)

Contoh :

```
class Login {  
    private String user, pass;  
    Login() {  
        this.user = "";  
        this.pass = "";  
    }  
    Login (String user, String pass) {  
        this.user = user;  
        this.pass = pass;  
    }  
}
```

```
public void inputUser(){
    this.user = user;
}
public void inputPass(){
    this.pass = pass;
}
}
```

3. Inheritance

Inheritance merupakan proses pewarisan data dan method dari suatu class yang telah ada kepada suatu class baru. Class yang mewariskan disebut dengan superclass / parent class / base class, sedangkan class yang mewarisi (class yang baru) disebut dengan subclass / child class / derived class. Subclass tidak dapat mewarisi anggota private dari superclass-nya.

Dengan inheritance, class yang baru (subclass) akan mirip dengan class yang lama (superclass) namun memiliki karakteristik yang baru. Dalam java, seubclass hanya bisa memiliki satu superclass (single inheritance) sedangkan superclass bisa memiliki satu subclass atau lebih.

Untuk menerapkan inheritance, gunakan statement “**extends**”

```
namaSubclass extends namaSuperclass
{
.... //definisi class
}
```

Keyword “super” digunakan oleh subclass untuk memanggil constructor atau method yang ada pada superclass-nya.

Contoh untuk memanggil constructor milik superclass-nya :

```
super()
super(parameter)
```

Contoh untuk memanggil method milik superclass-nya :

```
super.namaMethod(parameter)
```

Contoh inheritance :

```
class Pelajar {
private String nama;
private long tinggi;
private long berat;
```

```
public Pelajar (String nama, long tinggi, long berat){  
    this.nama = nama;  
    this.tinggi = tinggi;  
    this.berat = berat;  
}  
public String tampilPelajar(){  
    return ("Nama\t: " + nama + "\nTinggi\t: " + tinggi +  
"\nBerat\t: " + berat + "\n");  
}  
}  
  
class Siswa extends Pelajar {  
    private String nim, asalsekolah;  
    private long nilai;  
    public Siswa (String nama, long tinggi, long berat,  
        String nim, String asalsekolah, long nilai) {  
        super(nama,tinggi,berat);  
        this.nim = nim;  
        this.asalsekolah = asalsekolah;  
        this.nilai = nilai;  
    }  
    public String tampilSiswa(){  
        return (super.tampilPelajar()+"nim\t: " + nim + "\nAsal Sekolah  
:" + asalsekolah + "\nNilai\t:" + nilai);  
    }  
}  
  
class main {  
    public static void main(String[] args) {  
        Siswa ti = new Siswa ("Ahmad", 180,78,"1390192","SMK  
Informatika",90);  
        System.out.println(ti.tampilSiswa());  
    }  
}
```

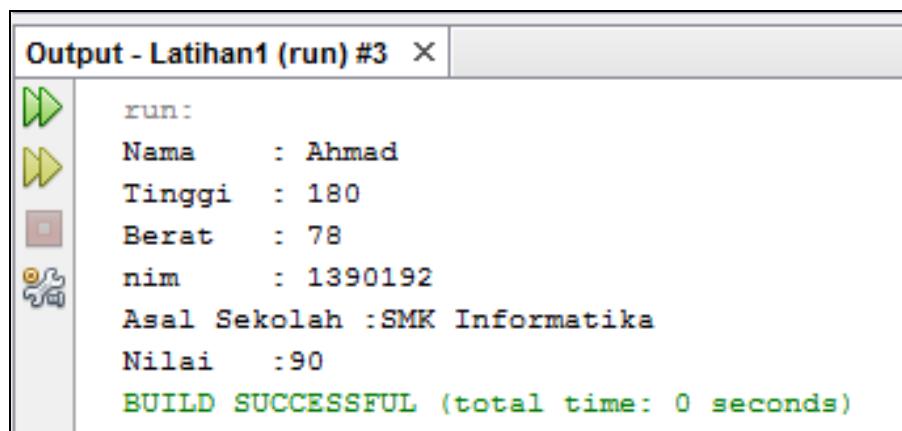
Catatan :

Escape Sequence	Description
\'	Single quote
\"	Double quote
\\\	Backslash
\r	Carriage return
\n	New line
\f	Form feed
\t	Horizontal tab
\b	Backspace
\ddd	Octal constant (where ddd is an octal constant)
\uxxxx	Hexadecimal constant (where xxxx is a hexadecimal constant)

Table 2-2 Character Escape Sequences

Sumber : *Herbert Schildt - A Beginner's Guide, 3rd Edition* (2005;37)

HASIL :



```
Output - Latihan1 (run) #3 ×
run:
Nama      : Ahmad
Tinggi   : 180
Berat    : 78
nim      : 1390192
Asal Sekolah :SMK Informatika
Nilai    :90
BUILD SUCCESSFUL (total time: 0 seconds)
```

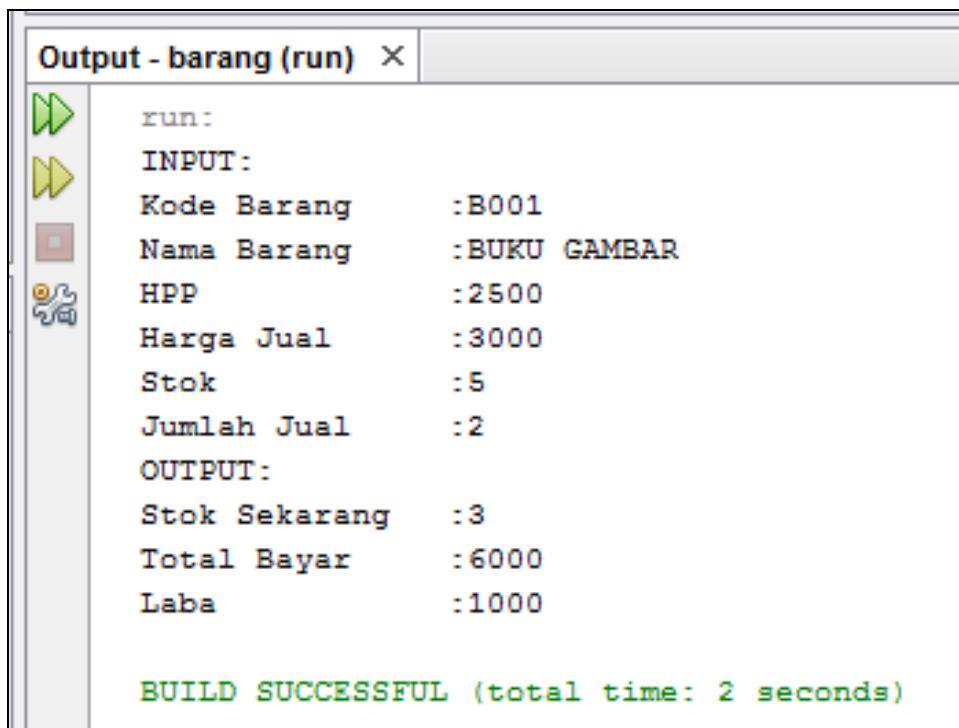
4. Latihan

Implementasikan inheritance kedalam kelas barang, dengan jenis barang yang diambil adalah masih buku gambar. Ketentuan sama namun untuk proses input dibuat dalam bentuk constructor . Proses yang dibutuhkan yaitu hitung stok, hitung total, hitung laba, dan tampil.

Keterangan :

- Barang (Super Class) = **Kode** dan **Nama**
- Buku Gambar (Sub Class) = **Kode**, **Nama**, HPP, Harga Jual, dan Stok

HASIL :



```
Output - barang (run) X
run:
INPUT:
Kode Barang      :B001
Nama Barang      :BUKU GAMBAR
HPP              :2500
Harga Jual       :3000
Stok             :5
Jumlah Jual      :2
OUTPUT:
Stok Sekarang    :3
Total Bayar      :6000
Laba             :1000

BUILD SUCCESSFUL (total time: 2 seconds)
```

KODE PROGRAM :

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

MODUL 8

POLYMORPHISME

1. Pengertian Polymorphisme

Salah satu konsep Pemrograman Berorientasi Objek adalah polymorfisme yaitu kemampuan beberapa objek bertipe sama bereaksi secara berbeda terhadap “pesan” yang sama.

Contoh :

```
class Manusia {  
    public String nama, hobby, pekerjaan;  
    public Manusia (String nm, String hobby, String kerja) {  
        this.nama= nm;  
        this.hobby = hobby;  
        this.pekerjaan = kerja;  
    }  
    public String tampilManusia (){  
        return ("Nama\t\t: " + nama + "\n"+  
                "Hobby\t\t: " + hobby + "\n"+  
                "Pekerjaan\t: " + pekerjaan + "\n");  
    }  
}  
  
class programmer extends Manusia {  
    public programmer(String nm, String hobby, String kerja) {  
        super(nm, hobby, kerja);  
    }  
}  
  
class polisi extends Manusia {  
    public polisi(String nm, String hobby, String kerja) {  
        super(nm,hobby,kerja);  
    }  
}
```

```
class kegiatan {  
    public static void main(String args[]) {  
        Manusia[] informasi= new Manusia[2];  
        informasi[0]=new programmer("Programmer", "Futsal", "Mengetik  
        Sintak");  
        informasi[1]=new polisi("Polisi", "Fusal", "Operasi keliling");  
        for (int i=0; i<2; i++) {  
            System.out.println("-----");  
            System.out.println(informasi[i].tampilManusia());  
        }  
    }  
}
```

HASIL :

```
Output - manusia (run) X  
run:  
-----  
Nama : Programmer  
Hobby : Futsal  
Pekerjaan : Mengetik Sintak  
-----  
Nama : Polisi  
Hobby : Fusal  
Pekerjaan : Operasi keliling  
  
BUILD SUCCESSFUL (total time: 1 second)
```

2. Latihan

Implementasikan konsep polymorphism ke dalam kelas barang, dengan jenis barang yang diambil adalah buku gambar dan pensil. Ketentuan sama namun untuk proses input dibuat dalam bentuk constructor . Proses yang dibutuhkan yaitu hitung stok, hitung total, hitung laba, dan tampil. Ubahlah latihan sebelumnya menjadi berkonsep polymorphism.

Keterangan :

- Barang (Super Class) = **Kode** dan **Nama**
 - Buku Gambar dan Pensil 2B (Sub Class) = **Kode**, **Nama**, HPP, Harga Jual, dan Stok

HASIL :

```
Kode : B001
Nama : BUKU GAMBAR
HPP : 2500
Harga Jual : 3000
Stok : 5
Jumlah Jual : 2
OUTPUT:
Stok Sekarang : 3
Total Bayar : 6000
Laba : 1000
-----
INPUT:
Kode : P002
Nama : PENSIL 2B
HPP : 1500
Harga Jual : 1750
Stok : 10
Jumlah Jual : 1
OUTPUT:
Stok Sekarang : 9
Total Bayar : 1750
Laba : 250
BUILD SUCCESSFUL (total time: 1 second)
```

KODE PROGRAM :

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

MODUL 9

DATABASE

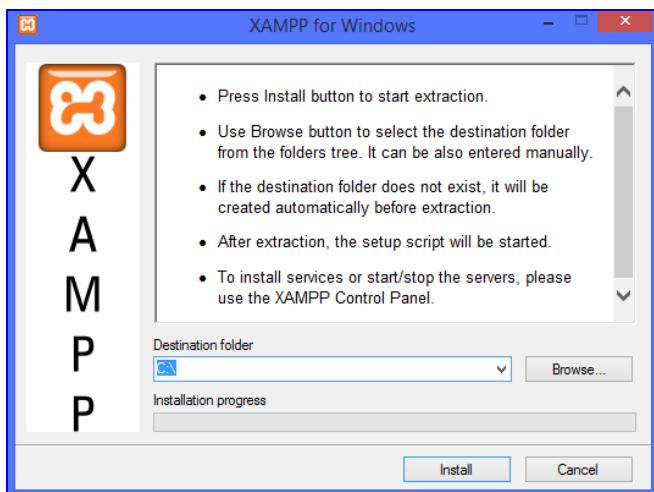
Bericara pemrograman maka tidak akan lepas dari yang namanya database. Dua hal ini tidak dapat dipisahkan dalam keilmuan pemrograman karena saling berhubungan, database sebagai tempat menyimpan data sedangkan pemrograman sebagai media untuk memanipulasi data tersebut. Bagaimana database dapat kita akses dengan menggunakan bahasa pemrograman dan dapat memanipulasinya baik itu menyisipkan data, mengubah data, menghapus data, dan menampilkan data. Hal ini sudah menjadi satu kesatuan sehingga keduanya perlu dipelajari. Dalam praktikum ini kita akan mencoba mengintegrasikan MySQL ke dalam pemrograman Java.

Software yang dibutukan dalam hal ini adalah :

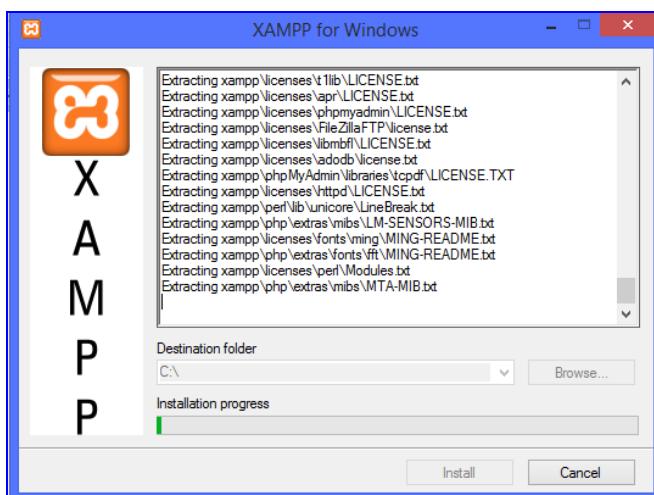
1. Installasi Xampp

Untuk melakuan instalasi xampp, langkahnya sebagai berikut :

- 1) Double klik software **xampp-win32-1.7.3.exe** didalam folder yang telah disediakan. Maka akan muncul tampilan seperti dibawah ini.



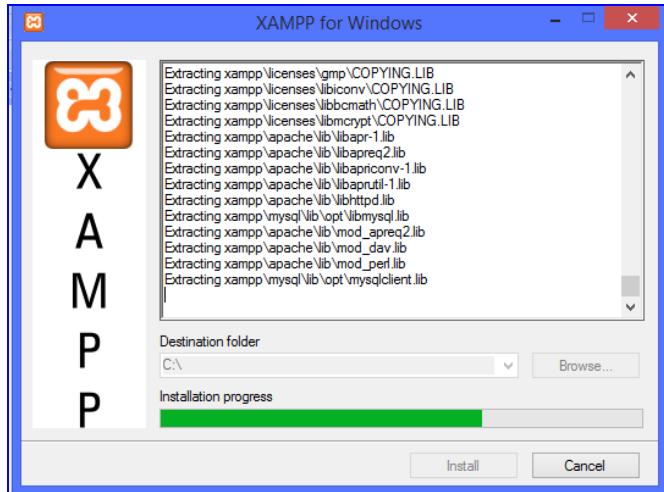
- 2) Tentukan lokasi penginstallan namun defaultnya di drive **C:** silahkan klik **Install**.



Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- 3) Tunggu sampai beberapa saat sehingga proses selesai, installasi xampp memerlukan waktu yang cukup lama.



- 4) Ketika proses selesai maka akan muncul tampilan **cmd.exe** dan sesuaikan perintahnya.

The image contains two stacked screenshots of a cmd.exe window. Both windows have the title "C:\Windows\system32\cmd.exe". The top window displays the initial setup configuration:

```
#####
# XAMPP 1.7.3 - Setup
#
# Copyright 2009 Carsten Wiedmann <FreeBSD License>
#
# Authors: Carsten Wiedmann <carsten_sttgt@gmx.de>
#          Kay Uogelgesang <kuo@apachefriends.org>
#####
Should I add shortcuts to the startmenu/desktop? <y/n>: y
```

The bottom window shows the continuation of the setup process:

```
#####
# XAMPP 1.7.3 - Setup
#
# Copyright 2009 Carsten Wiedmann <FreeBSD License>
#
# Authors: Carsten Wiedmann <carsten_sttgt@gmx.de>
#          Kay Uogelgesang <kuo@apachefriends.org>
#####
Should I locate the XAMPP paths correctly?
Should I proceed? <y/x=exit setup>: y
```

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

The image consists of three vertically stacked screenshots of a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe".

Screenshot 1: Displays the initial XAMPP setup screen with copyright information and a question: "Should I make a portable XAMPP without drive letters?". It also includes a note about drive letters and services.

```
#####
# XAMPP 1.7.3 - Setup
#
# Copyright 2009 Carsten Wiedmann <FreeBSD License>
#
# Authors: Carsten Wiedmann <carsten_sttigt@gmx.de>
#          Kay Vogelgesang <kvo@apachefriends.org>
#####
Should I make a portable XAMPP without drive letters?
NOTE: - You should use drive letters, if you want use services.
      - With USB sticks you must not use drive letters.
Your choice? <y/n>: n
```

Screenshot 2: Shows the setup process relocating various components like Apache, MySQL, and PHP. It ends with a message: "XAMPP is ready to use." and a prompt to press Return to continue.

```
relocating XAMPP...
relocate XAMPP base package
relocate Apache
relocate FileZilla FTP Server
relocate Mercury
relocate MySQL
relocate OpenSSL
relocate Perl
relocate PHP
relocate phpMyAdmin
relocate Sendmail
relocate Webalizer
relocate XAMPP Demopage
relocating XAMPP successful.

XAMPP is ready to use.

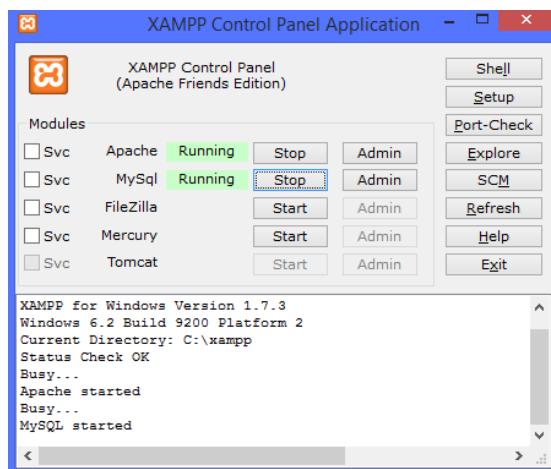
Press <Return> to continue: _
```

Screenshot 3: Displays the final XAMPP setup screen with copyright information and a menu of options. The user is prompted to choose an option from 1 to 7 or x for Exit.

```
#####
# XAMPP 1.7.3 - Setup
#
# Copyright 2009 Carsten Wiedmann <FreeBSD License>
#
# Authors: Carsten Wiedmann <carsten_sttigt@gmx.de>
#          Kay Vogelgesang <kvo@apachefriends.org>
#####
1. start XAMPP
2. relocate XAMPP
   (current path: C:\xampp)
3. disable HTTPS (SSL)
4. disable Server Side Includes (SSI)
5. enable IPv4 only (current: IPv4/6 <auto>)
6. disable mod_perl
7. disable Apache::ASP
x Exit

Please choose <1-7/x>: 1
```

- 5) Untuk menjalankan xampp, terlebih dahulu harus mengaktifkan **Apache** dan **MySQL**, caranya double klik **XAMPP Control Panel**. Klik **Start** untuk mengaktifkan keduanya.



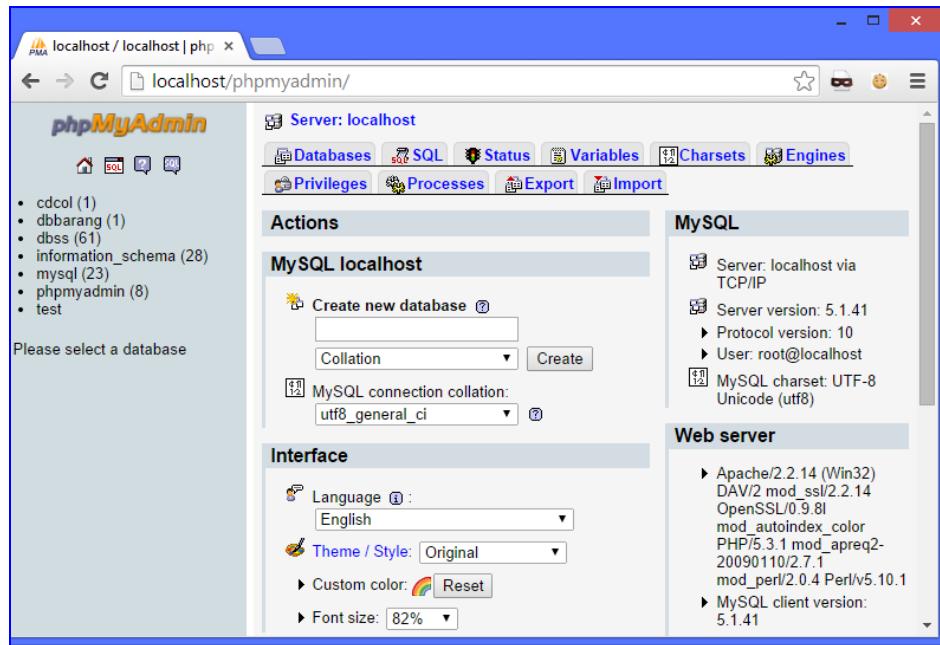
- 6) Untuk mencoba silahkan buka browser yang tersedia, silahkan masukkan alamat **localhost**, maka akan tampilan seperti dibawah ini.



2. Membuat Database

Untuk membuat database dengan menggunakan **PhpMyAdmin**, caranya sebagai berikut :

- 1) Buka browser, ketik <http://localhost/phpmyadmin/> , maka akan muncul tampilan seperti dibawah ini.



- 2) Pada kotak **Create new database** silahkan ketik nama databasenya misalkan **db_oop**. Kemudian klik **Create**.

A screenshot of the 'Create new database' dialog box. It shows the 'Create new database' field with 'db_oop' typed into it. Below it is a 'Collation' dropdown set to 'utf8_general_ci'. At the bottom right is a 'Create' button.

3. Membuat Tabel

Setelah selesai membuat database selanjutnya yaitu membuat table sebagai tempat penyimpanan datanya. Caranya sebagai berikut :

- 1) Didalam database **db_oop** yang telah dibuat tadi, buatlah satu buah tabel dengan nama tabelnya **barang** dan jumlah fielnya **4**. Seperti tampilan dibawah ini, lalu klik **Go**.

A screenshot of the 'Create new table on database db_oop' dialog box. It has two main input fields: 'Name:' with 'barang' typed into it, and 'Number of fields:' with '4' typed into it. At the bottom right is a 'Go' button.

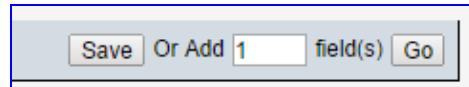
- 2) Kemudian masukan nama fieldnya yaitu **kode_barang**, **nama_barang**, **jumlah**, dan **harga** serta sesuaikan tipe datanya sehingga seperti tampilan dibawah ini. Sebagai **primarykey** adalah **kode_barang**.

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

Field	Type	Length/Values ¹	Null	Index	A_I
kode_barang	VARCHAR	10	<input type="checkbox"/>	PRIMARY	<input type="checkbox"/>
nama_barang	VARCHAR	50	<input type="checkbox"/>	---	<input type="checkbox"/>
jumlah	INT		<input type="checkbox"/>	---	<input type="checkbox"/>
harga	INT		<input type="checkbox"/>	---	<input type="checkbox"/>

- 3) Kemudian klik tombol **Save**.



- 4) Maka hasil akhirnya akan seperti dibawah ini.

Table 'db_oop`.`barang' has been created.

Field	Type	Collation	Attributes	Null	Default	Extra	Action
kode_barang	varchar(10)	latin1_swedish_ci		No	None		
nama_barang	varchar(50)	latin1_swedish_ci		No	None		
jumlah	int(11)			No	None		
harga	int(11)			No	None		

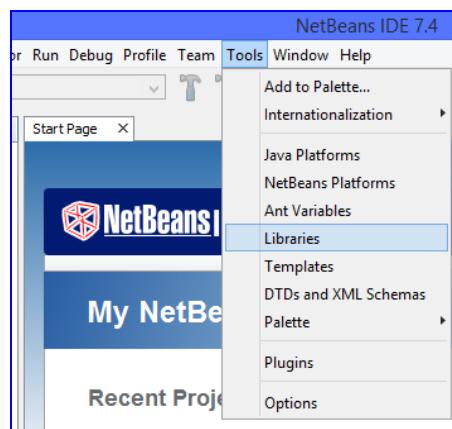
MODUL 10

INTEGRASI DATABASE DENGAN JAVA

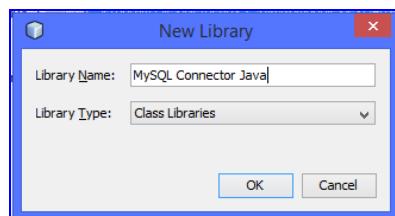
1. Membuat Library

Sebelum memulai membuat project java, harus membuat konektor terlebih dahulu untuk menghubungkan antara java dengan MySQL harus menggunakan mysql connector java dalam praktikum ini yang digunakan **mysql-connector-java-5.0.8-bin.jar**. Caranya sebagai berikut :

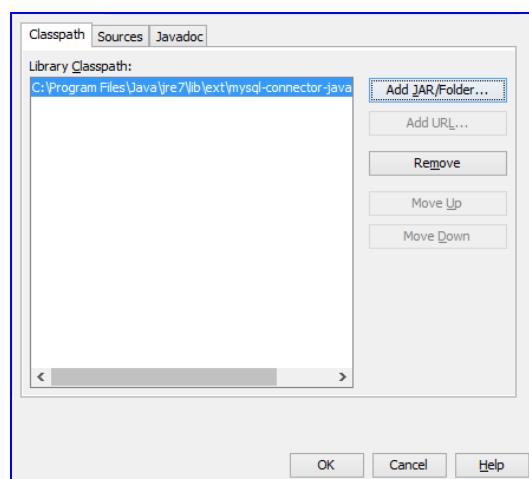
- 1) **Copy**-kan **mysql-connector-java-5.0.8-bin.jar** ke alamat **C:\Program Files\Java\jre7\lib\ext** kemudian **Paste**.
- 2) Buka **Netbean 7.4**, setelah muncul area kerja pilih menu **Tools – Libraries**.



- 3) Kemudian buatlah **Library** baru, dengan mengklik tombol **New Library....** Lalu beri nama seperti tampilan dibawah ini.



- 4) Klik tombol **Add JAR/Folder**. Tambahkan konektor yang tadi sudah dicopykan ke alamat **C:\Program Files\Java\jre7\lib\ext\mysql-connector-java-5.0.8-bin.jar**.

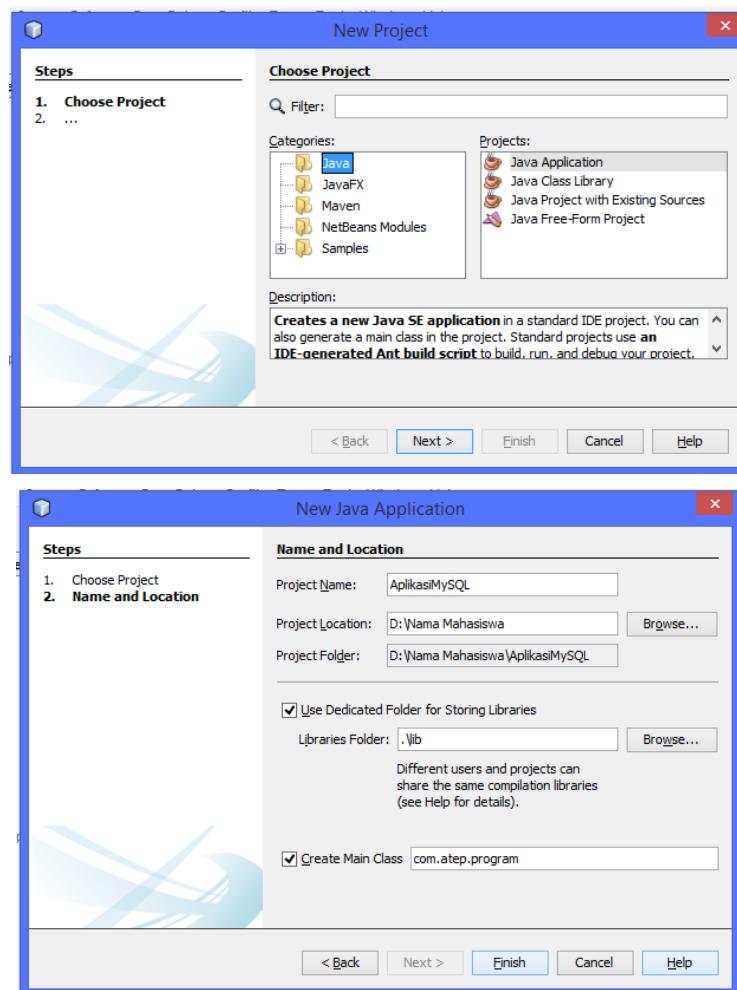


- 5) Lalu klik **Ok**. Maka library untuk konektor MySQL sudah dibuat.

2. Mendaftarkan Driver

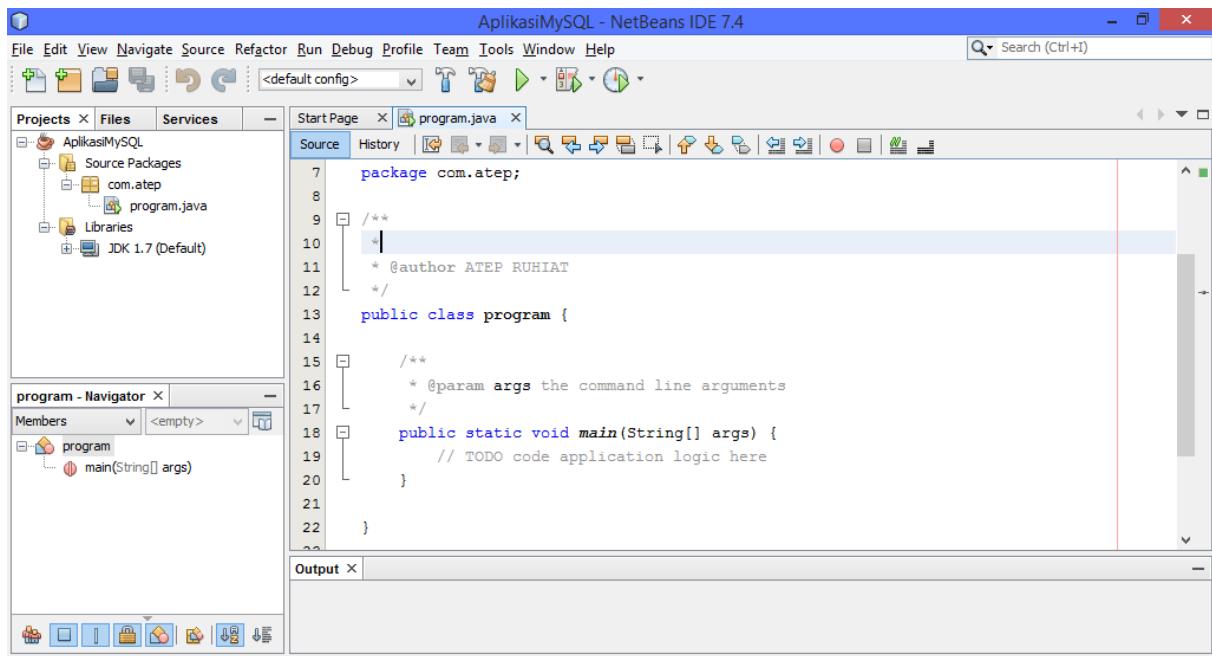
Setelah library konektor dibuat, selanjutnya tinggal mendaftarkan driver tersebut agar dapat digunakan. Caranya sebagai berikut :

- 1) Buatlah project baru dengan ketentuan sebagai berikut

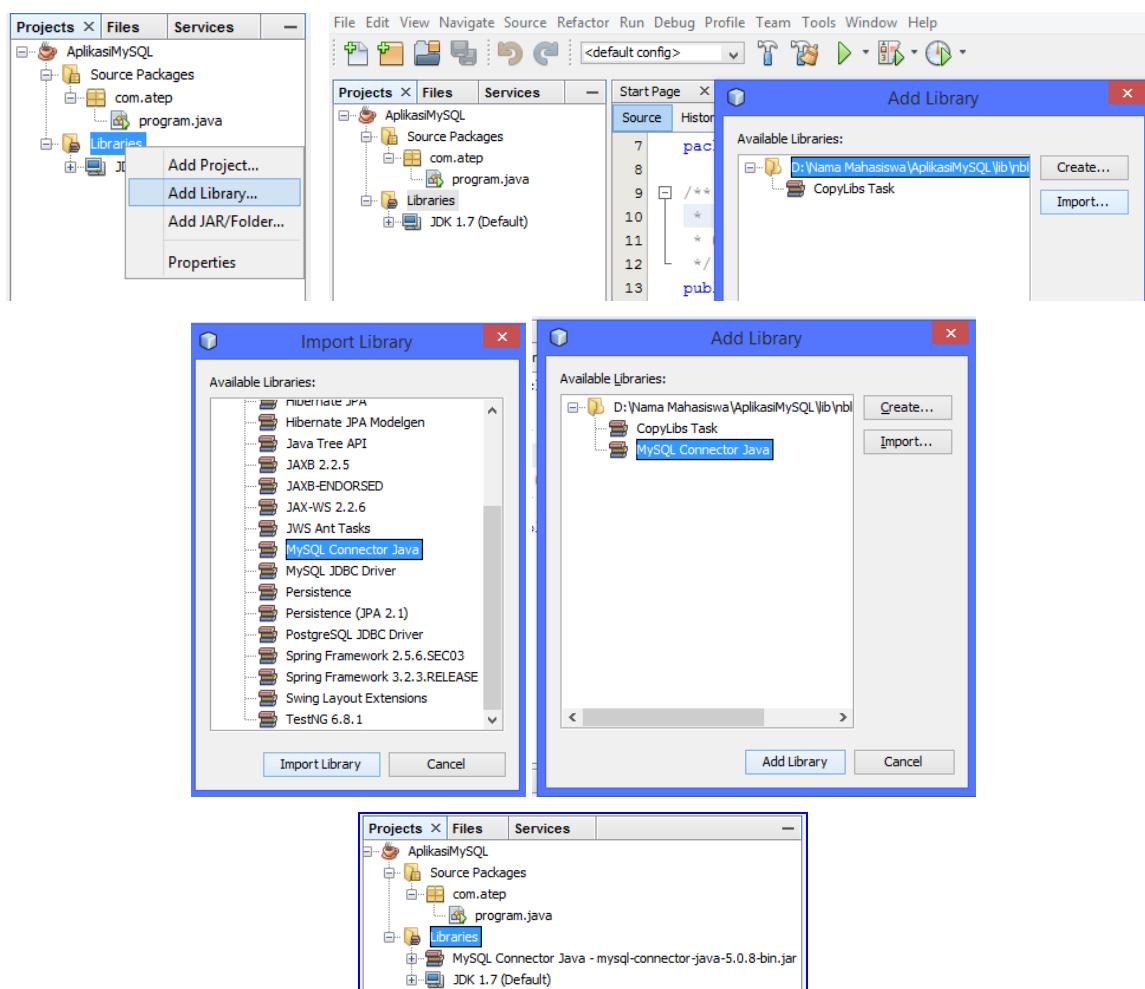


Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



- 2) Kemudian tambahkan library **MySQL Connector Java**, dengan melakukan klik kanan pada **Libraries** pilih **Add library...** pada project **AplikasiMySQL**.



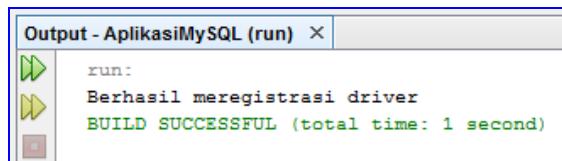
- 3) Kemudian ketikkan kode program seperti dibawah ini

```
package com.atep;
```

```
import com.mysql.jdbc.Driver;
import java.sql.DriverManager;
import java.sql.SQLException;

public class program {
    public static void main(String[] args) {
        try {
            Driver driver = new Driver();
            DriverManager.registerDriver(driver);
            System.out.println("Berhasil meregistrasi driver");
        }
        catch(SQLException ex){
            System.out.println("Gagal meregistrasi driver");
            System.out.println("Pesan : "+ex.getMessage());
        }
    }
}
```

- 4) Coba jalankan program dengan menekan kombinasi **Shift+F6**.



3. Membuat Connection

Untuk membuat koneksi java ke MySQL caranya tinggal menambahkan kode program seperti dibawah ini.

```
package com.atep;
import com.mysql.jdbc.Driver;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class program {
    public static void main(String[] args) {
        try {
            Driver driver = new Driver();
            DriverManager.registerDriver(driver);
            System.out.println("Berhasil meregistrasi driver");
        }
    }
}
```

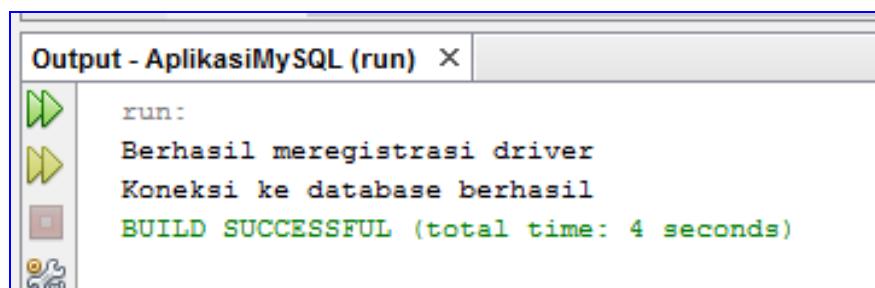
Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
    }
    catch(SQLException ex){
        System.out.println("Gagal meregistrasi driver");
        System.out.println("Pesan : "+ex.getMessage());
    }
    try {
        String url="jdbc:mysql://localhost:3306/db_oop";
        String user="root";
        String pass="";
        Connection koneksi = DriverManager.getConnection(url, user, pass);
        System.out.println("Koneksi ke database berhasil");
    }
    catch(SQLException ex){
        System.out.println("Koneksi ke database gagal");
        System.out.println("Pesan : "+ex.getMessage());
    }
}

}
```

HASIL :



The screenshot shows an IDE's output window titled "Output - AplikasiMySQL (run)". The window contains the following text:
run:
Berhasil meregistrasi driver
Koneksi ke database berhasil
BUILD SUCCESSFUL (total time: 4 seconds)

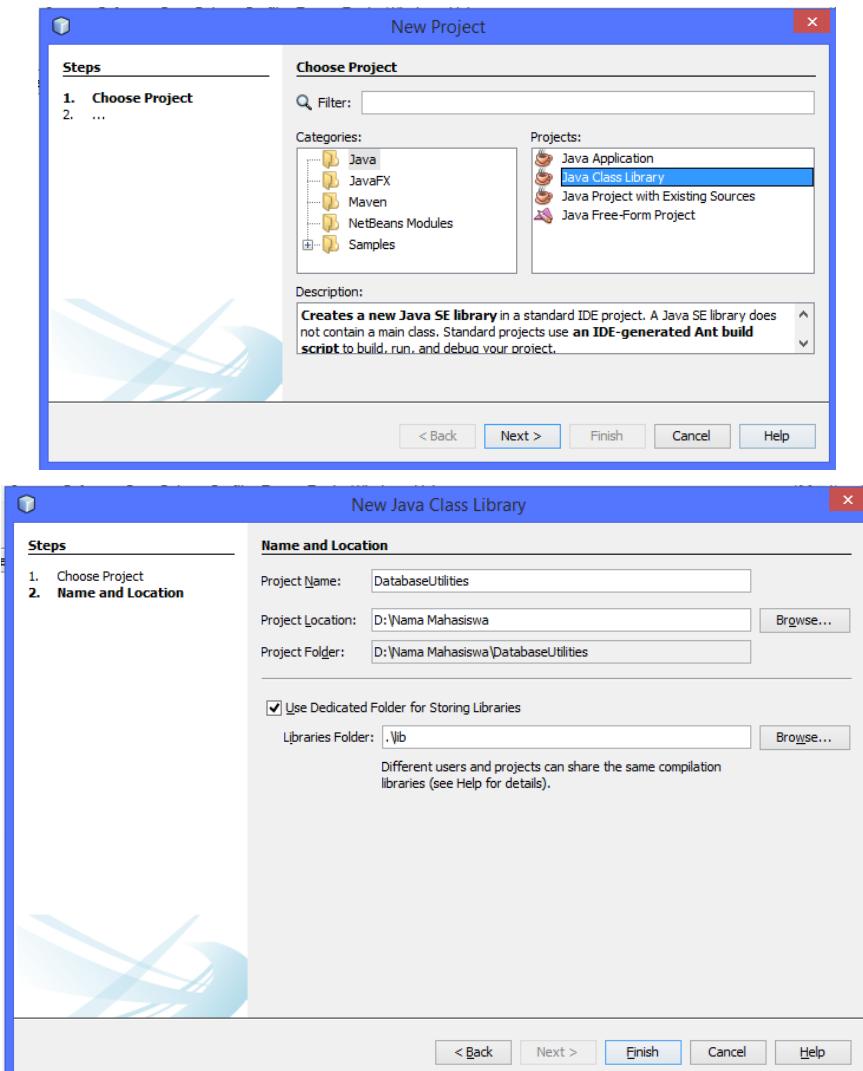
MODUL 11

STATEMENT

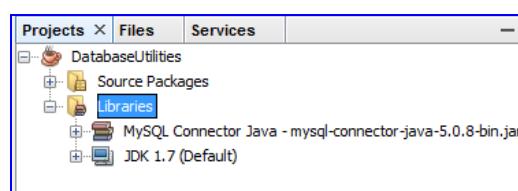
1. Membuat Database Utilities

Sebelum memanipulasi data sebaiknya membuat project Database Utilities sehingga nanti kita tidak perlu lagi membuat register dan membuat koneksi lagi yang berulang. Caranya sebagai berikut :

- 1) Buatlah project baru namun untuk tipe projectnya pilih **Java Class Library**.



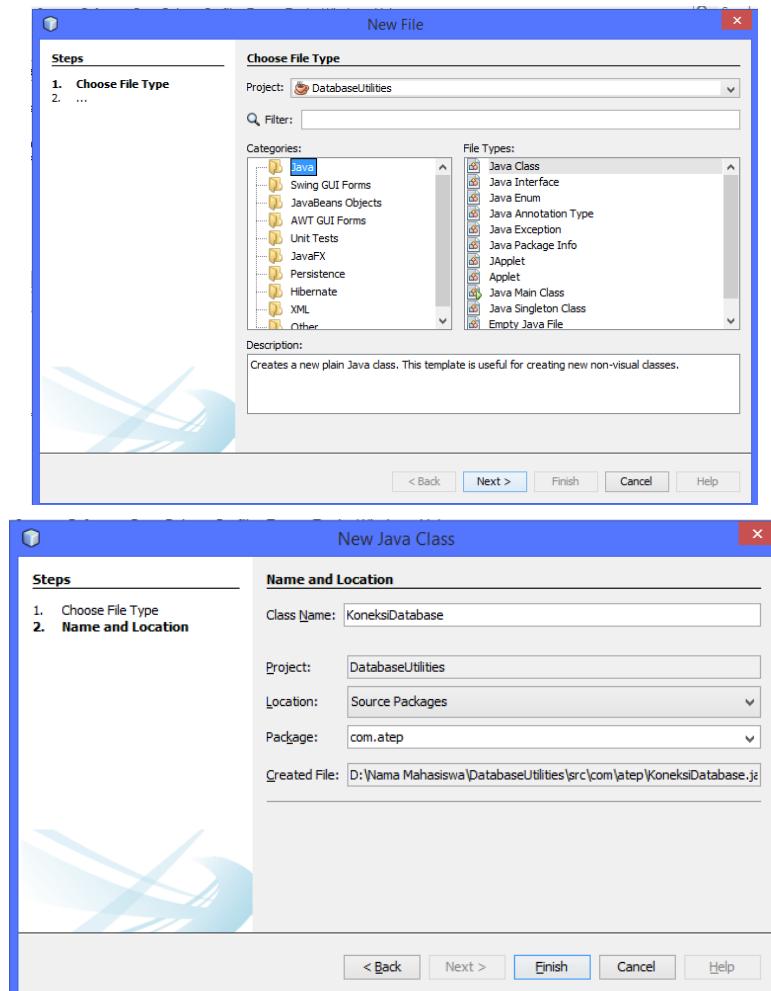
- 2) Kemudian tambahkan library **MySQL Connector Java**, dengan melakukan klik kanan pada **Libraries** pilih **Add library...** pada project **DatabaseUtilities**.



- 3) Lalu buat file baru (**new file**).

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



- 4) Ketikkan kode program seperti dibawah ini.

```
package com.atep;

import com.mysql.jdbc.Driver;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class KoneksiDatabase {
private static Connection koneksi;
public static Connection getKoneksi(){
    if (koneksi == null){
        try{
            Driver driver = new Driver();
            DriverManager.registerDriver(driver);
            String url="jdbc:mysql://localhost:3306/db_oop";
            String user="root";
            String pass ="";
        }
    }
}
```

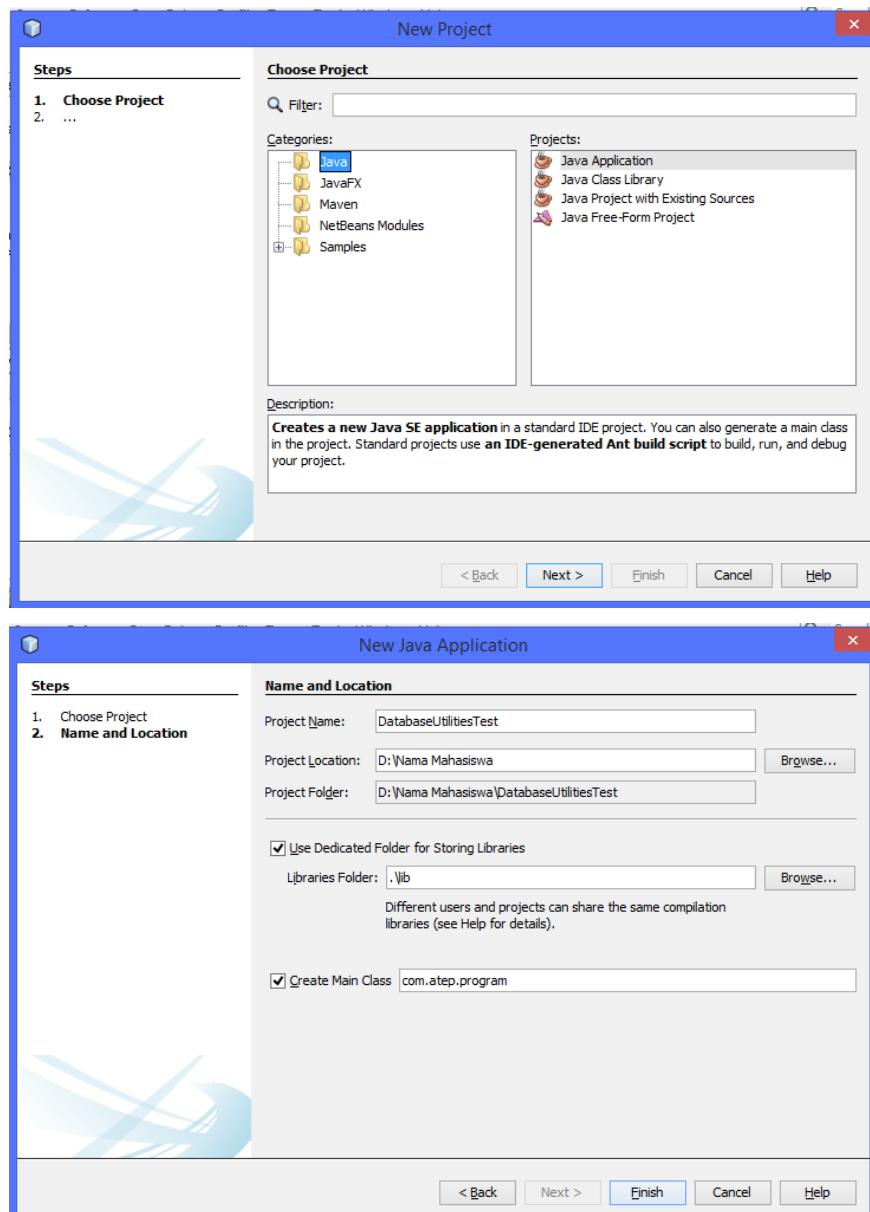
Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
koneksi = DriverManager.getConnection(url, user, pass);
System.out.println("Koneksi berhasil");
}catch(SQLException ex){
    System.out.println("Koneksi gagal");
    System.out.println("Pesan : "+ex.getMessage());
}

}
return koneksi;
}
}
```

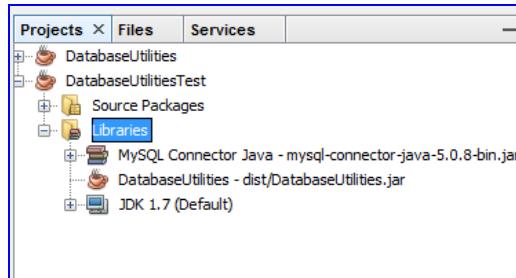
- 5) Untuk mengecek koneksi maka buatlah project baru, dengan nama **DatabaseUtilitiesTest**.



Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- 6) Kemudian tambahkan library **MySQL Connector Java**, dengan melakukan klik kanan pada **Libraries** pilih **Add library...** pada project **DatabaseUtilitiesTest**.
- 7) Lalu tambahkan juga project **DatabaseUtilities** sebagai project sumber, dengan cara klik kanan pada **Libraries** pilih **Add project...** pada project **DatabaseUtilitiesTest**
- 8) Sehingga akan seperti tampilan dibawah ini.



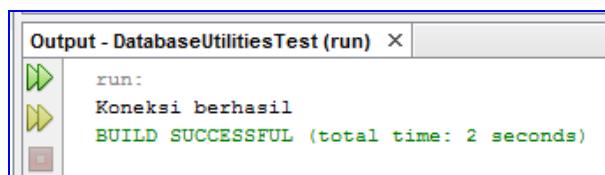
- 9) Kemudian ketik kode program seperti dibawah ini.

```
package com.atep;

public class program {

    public static void main(String[] args) {
        KoneksiDatabase.getKoneksi();
    }
}
```

HASIL :



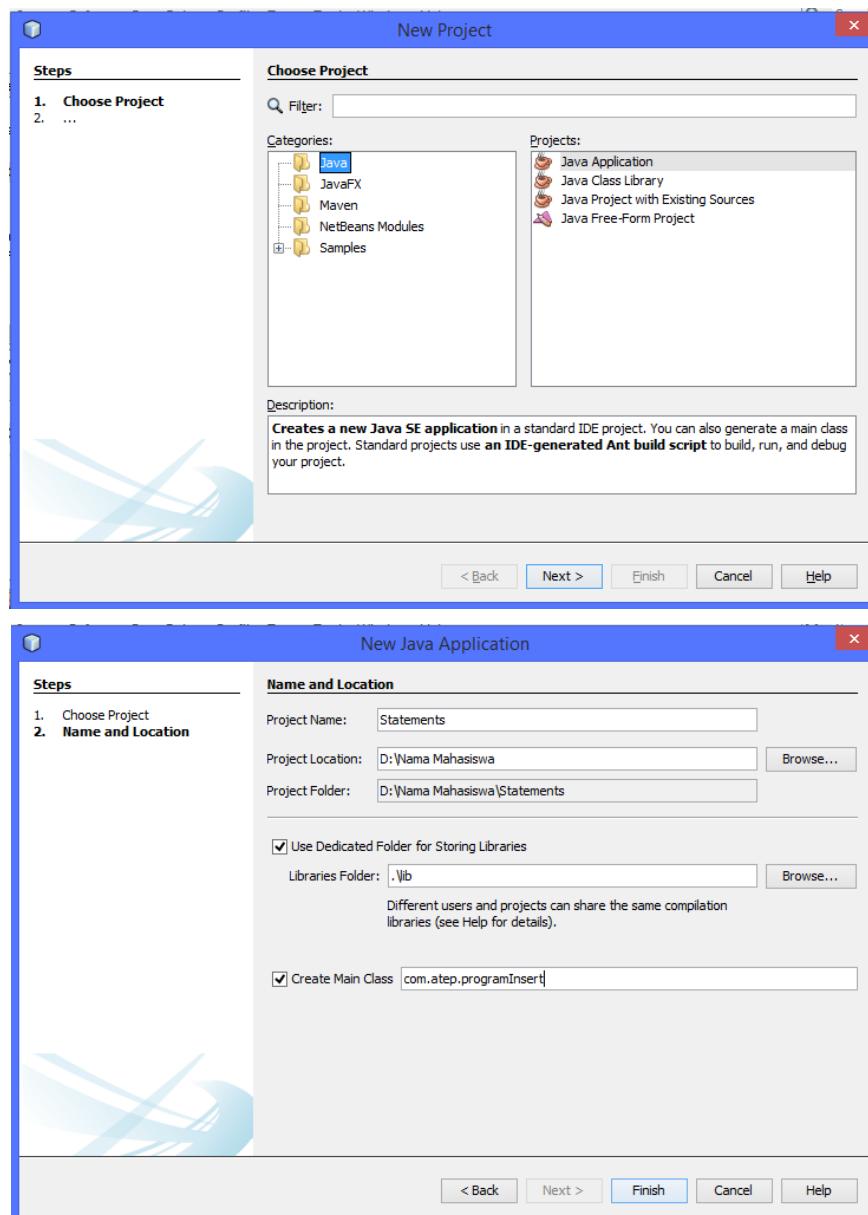
2. Statement Untuk Menyimpan Data Ke MySQL

Statement adalah perintah untuk memanipulasi data misalnya insert, update, delete, dan select. Untuk membuat statement yang pertama yaitu untuk menyimpan data, aranya sebagai berikut :

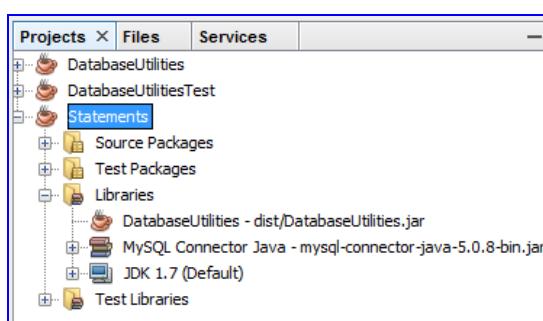
- 1) Untuk mengecek koneksi maka buatlah project baru, dengan nama **Statement**.

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



- 2) Kemudian tambahkan library **MySQL Connector Java**, dengan melakukan klik kanan pada **Libraries** pilih **Add library...** pada project **Statements**.
- 3) Lalu tambahkan juga project **DatabaseUtilities** sebagai project sumber, dengan cara klik kanan pada Libraries pilih **Add project...** pada project **Statements**.
- 4) Sehingga akan seperti tampilan dibawah ini.



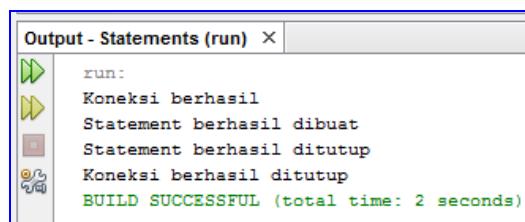
Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- 5) Ketikkan kode program dibawah ini.

```
package com.atep;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
public class programInsert {
    public static void main(String[] args) {
        Connection koneksi = KoneksiDatabase.getKoneksi();
        Statement statement = null;
        try {
            statement = koneksi.createStatement();
            System.out.println("Statement berhasil dibuat");
            String url= "INSERT INTO barang
(kode_barang,nama_barang,jumlah,harga)VALUES('B001','BUKU GAMBAR',10,3000)";
            statement.executeUpdate(url);
        }catch(SQLException ex){
            System.out.println("Statement gagal dibuat");
            System.out.println("Pesan : "+ex.getMessage());
        }finally{
            if (statement != null){
                try{
                    statement.close();
                    System.out.println("Statement berhasil ditutup");
                }catch(SQLException ex){
                    System.out.println("Statement gagal ditutup");
                    System.out.println("Pesan : "+ex.getMessage());
                }
            }
        }
    }
}
```

- 6) Coba jalankan program dengan menekan kombinasi **Shift+F6**.



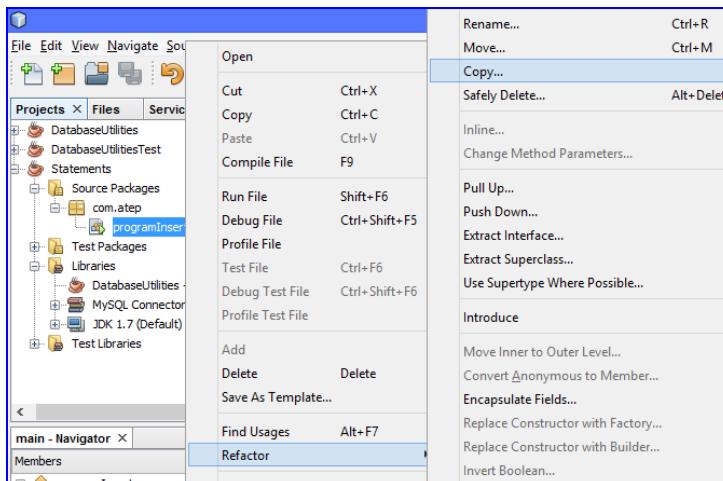
- 7) Kalau dilihat didalam database maka data akan tersimpan.

	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B001	BUKU GAMBAR	10	3000

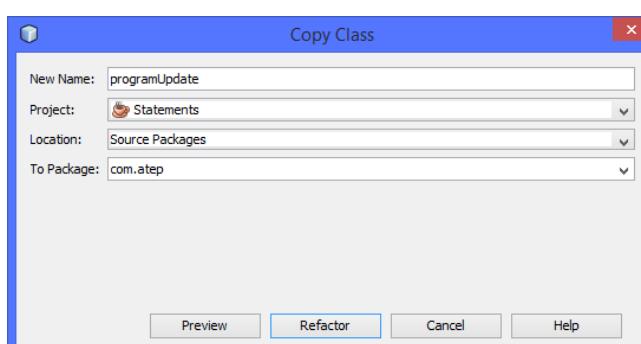
3. Statement Untuk Mengubah Data Ke MySQL

Dalam membuat statement ubah data caranya hampir sama dengan statement untuk menyimpan data. Untuk lebih jelasnya caranya sebagai berikut :

- 1) Copykan class **programInsert.java**, caranya klik kanan class tersebut lalu pilih **Refactor – Copy**.



- 2) Pada kotak **New Name** isi nama classnya dengan nama **programUpdate** lalu klik tombol **Refactor**.

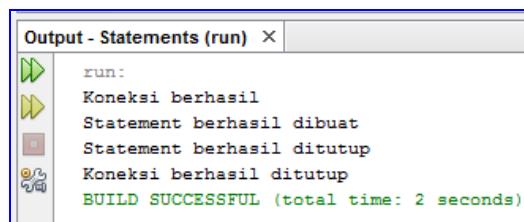


- 3) Perbaiki kode program sehingga sesuai dengan

```
package com.atep;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
public class programUpdate {
    public static void main(String[] args) {
        Connection koneksi = KoneksiDatabase.getKoneksi();
```

```
Statement statement = null;
try {
    statement = koneksi.createStatement();
    System.out.println("Statement berhasil dibuat");
    String url= "UPDATE barang SET nama_barang='PENSIL
2B',jumlah=20,harga=2500 WHERE kode_barang='B001'";
    statement.executeUpdate(url);
}catch(SQLException ex){
    System.out.println("Statement gagal dibuat");
    System.out.println("Pesan : "+ex.getMessage());
}finally{
    if (statement != null){
        try{
            statement.close();
            System.out.println("Statement berhasil ditutup");
        }catch(SQLException ex){
            System.out.println("Statement gagal ditutup");
            System.out.println("Pesan : "+ex.getMessage());
        }
    }
}
}
```

- 4) Coba jalankan program dengan menekan kombinasi **Shift+F6**.



- 5) Silahkan cek dalam database untuk melihat perubahan data yang sudah dilakukan.

The screenshot shows the MySQL Workbench interface with a table named 'barang'. The table has columns: kode_barang, nama_barang, jumlah, and harga. One row is visible with the values: B001, PENSIL 2B, 20, and 2500. Below the table, there are buttons for 'Check All / Uncheck All With selected' and a search bar for 'Show : 30 row(s) starting from record # 0'. There are also dropdown menus for 'in horizontal' and 'mode and repeat headers after 100 cells'.

	kode_barang	nama_barang	jumlah	harga
	B001	PENSIL 2B	20	2500

4. Statement Untuk Menghapus Data Ke MySQL

Dalam membuat statement hapus data caranya hampir sama dengan statement untuk menyimpan dan mengubah data. Untuk lebih jelasnya caranya sebagai berikut :

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- 1) Sebelum memulai membuat program untuk menghapus data terlebih dahulu lihat data yang ada pada database. Misalkan terdapat dua buah data seperti tampilan dibawah ini.

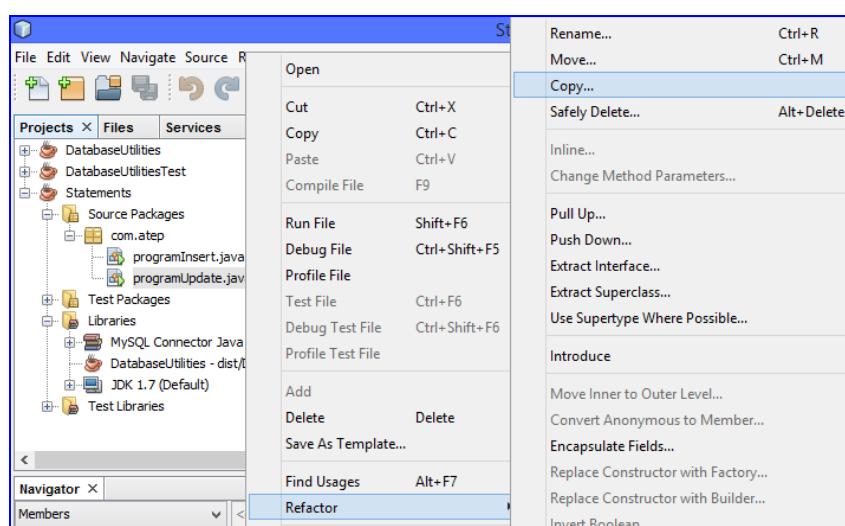
	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B001	PENSIL 2B	20	2500
<input type="checkbox"/>	B002	BUKU GAMBAR	10	3000

Check All / Uncheck All With selected:

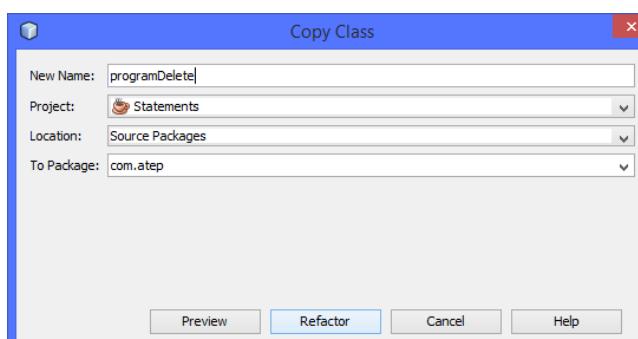
Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after

- 2) Lalu copykan class **programUpdate.java**, caranya klik kanan class tersebut lalu pilih **Refactor – Copy**.



- 3) Pada kotak **New Name** isi nama classnya dengan nama **programDelete** lalu klik tombol **Refactor**.

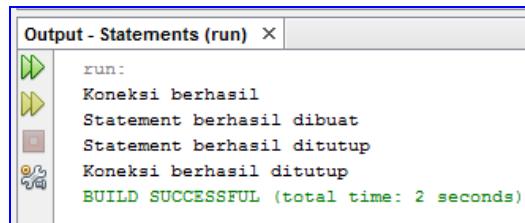


- 4) Perbaiki kode program sehingga sesuai dengan

```
package com.atep;  
  
import java.sql.Connection;  
import java.sql.SQLException;  
import java.sql.Statement;  
  
public class programDelete {  
    public static void main(String[] args) {  
        Connection koneksi = KoneksiDatabase.getKoneksi();  
        Statement statement = null;
```

```
try {
    statement =koneksi.createStatement();
    System.out.println("Statement berhasil dibuat");
    String url="DELETE FROM barang WHERE kode_barang='B001'";
    statement.executeUpdate(url);
}catch(SQLException ex){
    System.out.println("Statement gagal dibuat");
    System.out.println("Pesan : "+ex.getMessage());
}finally{
    if (statement != null){
        try{
            statement.close();
            System.out.println("Statement berhasil ditutup");
        }catch(SQLException ex){
            System.out.println("Statement gagal ditutup");
            System.out.println("Pesan : "+ex.getMessage());
        }
    }
}
}
```

- 5) Coba jalankan program dengan menekan kombinasi **Shift+F6**.



- 6) Silahkan cek dalam database, maka data yang kode barangnya = B001 akan dihapus seperti tampilan dibawah ini.

	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="View"/>	B002	BUKU GAMBAR	10	3000

Check All / Uncheck All With selected:

Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after

5. Membuat Database Service

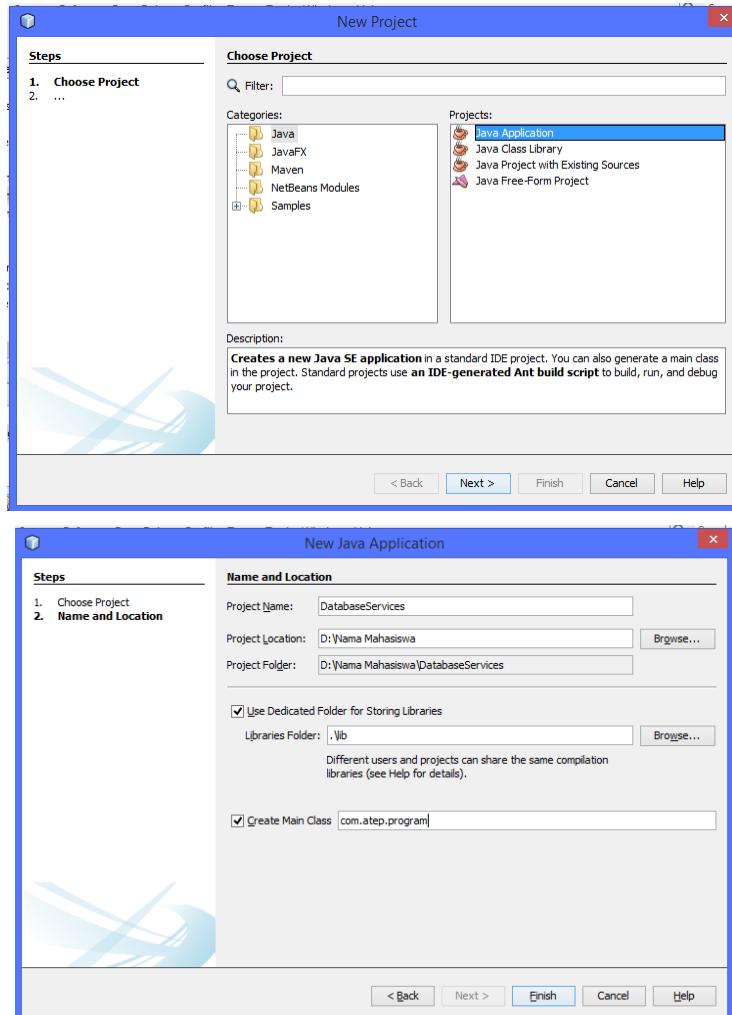
Yang sebelumnya telah dipraktekan untuk pembuatan statement insert, update, dan delete itu adalah bersifat static maksudnya kita dapat menginput, merubah, dan menghapus data dalam statement

Object Oriented Programming (OOP)

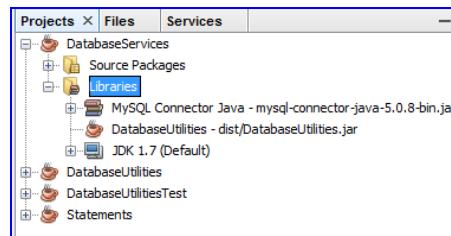
Guru : BEBEN SUTARA, S.Kom., M.T

sehingga ketika kita ingin misalkan menginput data maka harus merubah struktur kodennya. Sekarang kita coba akan membuat dinamisnya dengan cara datanya akan dijadikan parameter sehingga kita dapat menginput data dengan parameter tersebut. Caranya sebagai berikut :

- 1) Buat project baru, sesuaikan seperti tampilan dibawah ini.



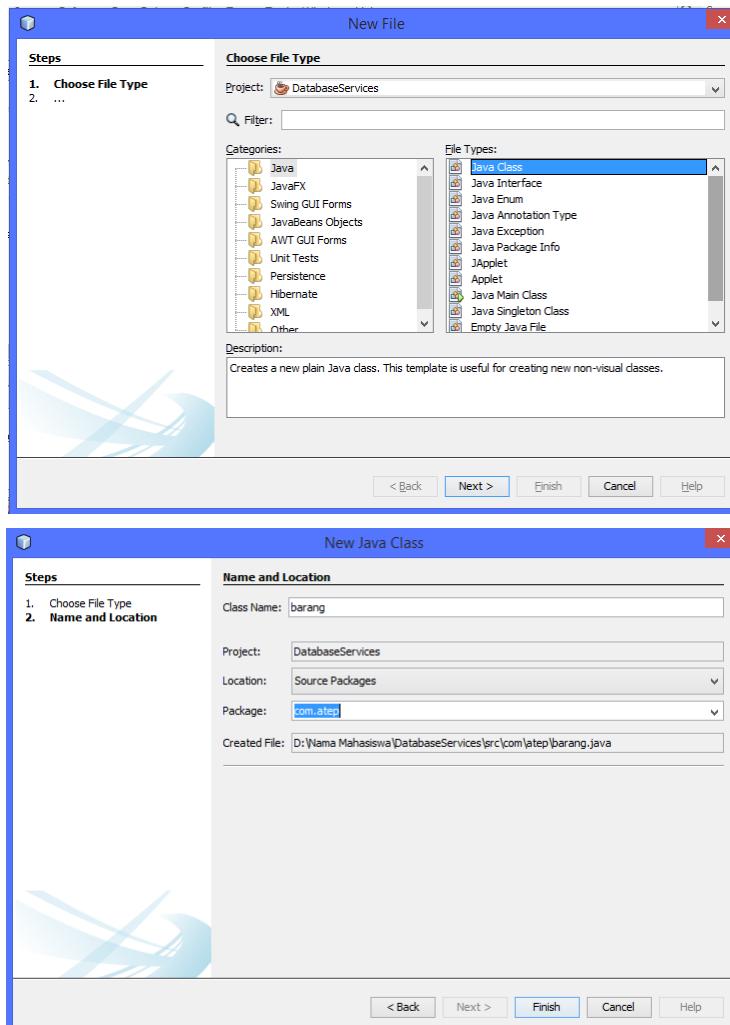
- 2) Pada **Libraries** project **DatabaseServices** tambahkan project **DatabaseUtilities** dan **MySQL Connector Java**. Seperti tampilan dibawah ini.



- 3) Buatlah kelas baru sesuaikan dengan table yang kita buat, misalkan **barang.java**

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



- 4) Buatlah kode program seperti tampilan dibawah ini.

```
package com.atep;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
public class barang {
    private Connection koneksi;
    public barang() {
        koneksi = KoneksiDatabase.getKoneksi();
    }
    public void insert(String kd, String nm, int jml, long hrg){
        Statement statement = null;
        try {
            statement =koneksi.createStatement();
            System.out.println("Statement berhasil dibuat");
        }
```

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
String url= "INSERT INTO barang
(kode_barang,nama_barang,jumlah,harga)VALUES('"+kd+"','"+nm+"','"+jml+"','"+hrg+"')";
statement.executeUpdate(url);
}catch(SQLException ex){
    System.out.println("Statement gagal dibuat");
    System.out.println("Pesan : "+ex.getMessage());
}finally{
    if (statement != null){
        try{
            statement.close();
            System.out.println("Statement berhasil ditutup");
        }catch(SQLException ex){
            System.out.println("Statement gagal ditutup");
            System.out.println("Pesan : "+ex.getMessage());
        }
    }
}
}

public void update(String kd, String nm, int jml, long hrg){
    Statement statement = null;
    try {
        statement =koneksi.createStatement();
        System.out.println("Statement berhasil dibuat");
        String url="UPDATE barang
nama_barang='"+nm+"',jumlah='"+jml+"',harga='"+hrg+"' WHERE kode_barang='"+kd+"'";
        statement.executeUpdate(url);
    }catch(SQLException ex){
        System.out.println("Statement gagal dibuat");
        System.out.println("Pesan : "+ex.getMessage());
    }finally{
        if (statement != null){
            try{
                statement.close();
                System.out.println("Statement berhasil ditutup");
            }catch(SQLException ex){
                System.out.println("Statement gagal ditutup");
                System.out.println("Pesan : "+ex.getMessage());
            }
        }
    }
}
```

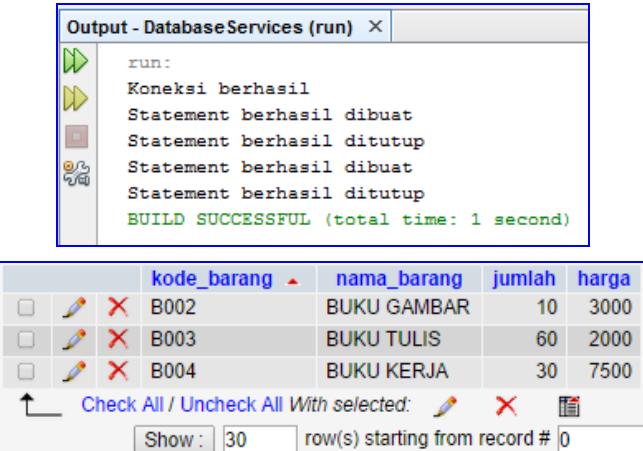
```
        }
    }

    public void delete(String kd){
        Statement statement = null;
        try {
            statement = koneksi.createStatement();
            System.out.println("Statement berhasil dibuat");
            String url="DELETE FROM barang WHERE kode_barang='"+kd+"'";
            statement.executeUpdate(url);
        }catch(SQLException ex){
            System.out.println("Statement gagal dibuat");
            System.out.println("Pesan : "+ex.getMessage());
        }finally{
            if (statement != null){
                try{
                    statement.close();
                    System.out.println("Statement berhasil ditutup");
                }catch(SQLException ex){
                    System.out.println("Statement gagal ditutup");
                    System.out.println("Pesan : "+ex.getMessage());
                }
            }
        }
    }
}
```

- 5) Untuk mencoba program, silahkan ketikkan kode program dibawah ini pada class program.

- a) Perintah **insert (Shift+F6)**

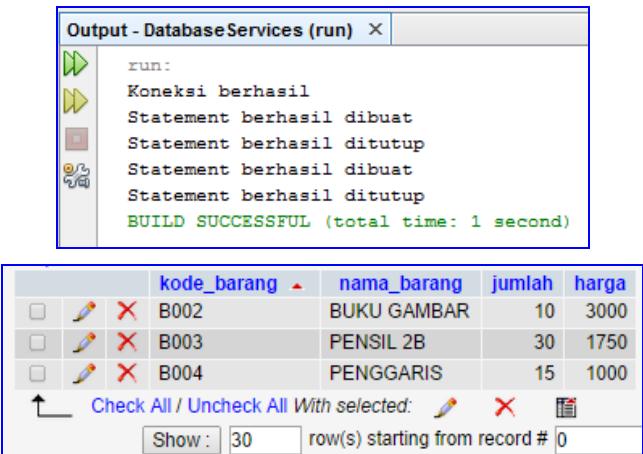
```
package com.atep;
public class program {
    public static void main(String[] args) {
        barang buku = new barang();
        buku.insert("B003", "BUKU TULIS", 60, 2000);
        buku.insert("B004", "BUKU KERJA", 30, 7500);
    }
}
```



	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B002	BUKU GAMBAR	10	3000
<input type="checkbox"/>	B003	BUKU TULIS	60	2000
<input type="checkbox"/>	B004	BUKU KERJA	30	7500

b) Perintah **Update (Shift+F6)**

```
package com.atep;
public class program {
    public static void main(String[] args) {
        barang buku = new barang();
        buku.update("B003", "PENSIL 2B", 30, 1750);
        buku.update("B004", "PENGgaris", 15, 1000);
    }
}
```



	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B002	BUKU GAMBAR	10	3000
<input type="checkbox"/>	B003	PENSIL 2B	30	1750
<input type="checkbox"/>	B004	PENGgaris	15	1000

c) Perintah **Delete (Shift+F6)**

```
package com.atep;
public class program {
    public static void main(String[] args) {
        barang buku = new barang();
        buku.delete("B002");
    }
}
```

The screenshot shows two windows. The top window is titled "Output - DatabaseServices (run)" and displays the following text:
run:
Koneksi berhasil
Statement berhasil dibuat
Statement berhasil ditutup
BUILD SUCCESSFUL (total time: 2 seconds)
The bottom window is a table view with the following data:

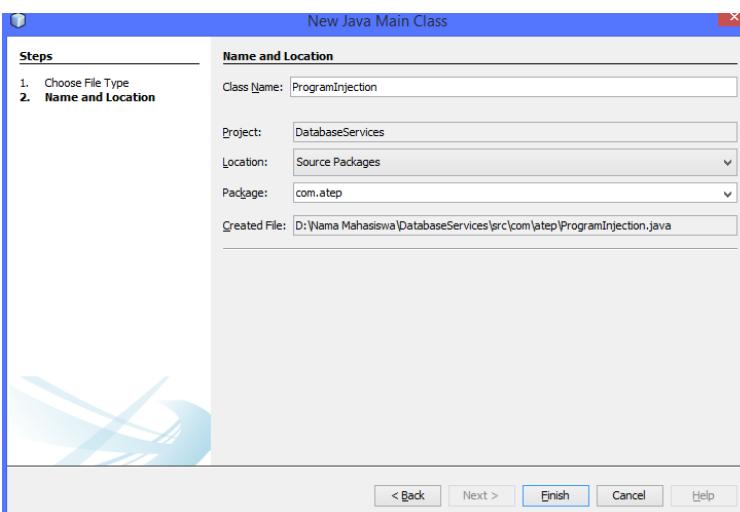
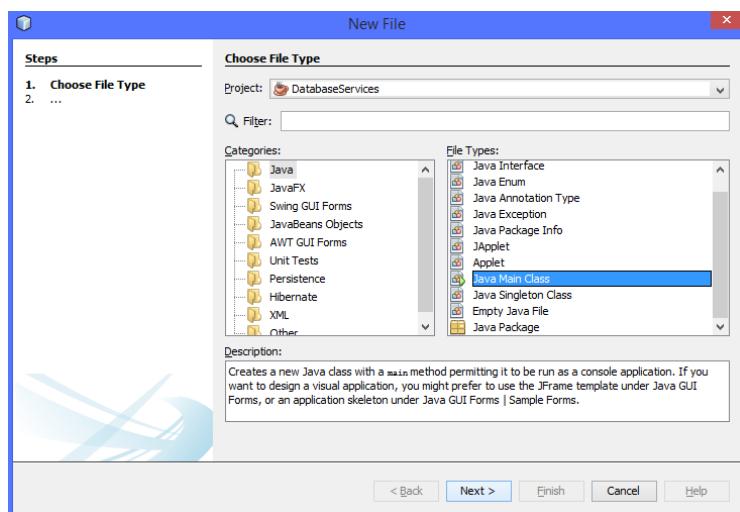
	kode_barang	nama_barang	jumlah	harga
□	B003	PENSIL 2B	30	1750
□	B004	PENGGARIS	15	1000

Below the table are buttons for "Check All / Uncheck All", "With selected:", and "Show : 30". A message at the bottom says "row(s) starting from record # 0".

6. SQLInjection di Statement

Salah satu kekurangan fitur statement adalah dia tidak tahan terhadap SQLInjection. SQLInjection adalah kode program atau perintah yang dimasukan oleh pihak yang tidak bertanggung jawab dengan maksud merusak atau menghapus data yang telah ada pada database kita. Sebagai contoh ikuti langkah berikut :

- 1) Buat file baru, seperti dibawah ini pilih **Java - Java Main Class**.



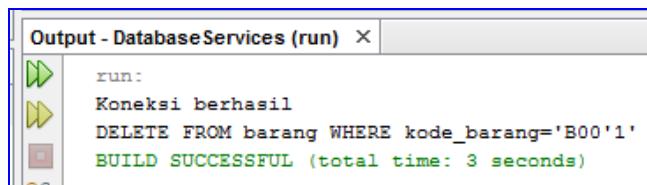
Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- 2) Coba ketikkan kode program seperti dibawah ini

```
package com.atep;
public class ProgramInjection {
    public static void main(String[] args) {
        barang barang = new barang();
        String kd="B00'1";
        String sql = "DELETE FROM barang WHERE kode_barang='"+kd+"'";
        System.out.println(sql);
    }
}
```

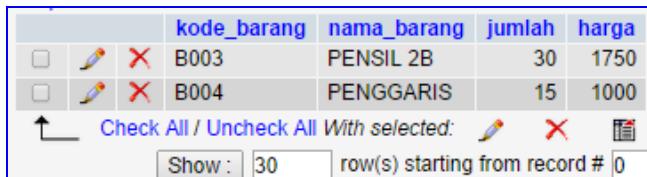
- 3) Lalu tekan kombinasi **Shift+F6**, maka akan hasilnya seperti tampilan dibawah ini.



```
Output - DatabaseServices (run) ×
run:
Koneksi berhasil
DELETE FROM barang WHERE kode_barang='B00'1'
BUILD SUCCESSFUL (total time: 3 seconds)
```

Ket : hasil tersebut menunjukkan bawah tanda kutip satu ('') sangat berpengaruh dalam statement.

- 4) Coba cek database dan lihat data yang ada pada database.



	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B003	PENSIL 2B	30	1750
<input type="checkbox"/>	B004	PENG GARIS	15	1000

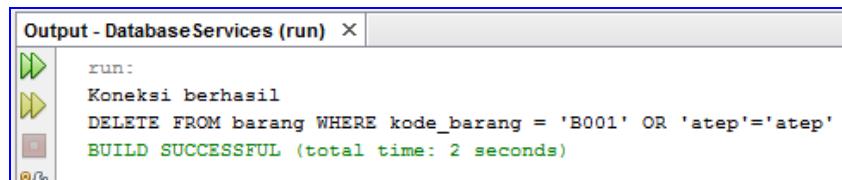
Check All / Uncheck All With selected:

Show : 30 row(s) starting from record # 0

- 5) Misalkan terdapat dua buah data, kemudian coba ubah kode programnya menjadi seperti dibawah ini.

```
package com.atep;
public class ProgramInjection {
    public static void main(String[] args) {
        barang barang = new barang();
        String kd="B001 OR 'atep'='atep";
        String sql = "DELETE FROM barang WHERE kode_barang='"+kd+"'";
        System.out.println(sql);
    }
}
```

- 6) Lalu tekan kombinasi **Shift+F6**, maka akan hasilnya seperti tampilan dibawah ini.



```
Output - DatabaseServices (run) ×
run:
Koneksi berhasil
DELETE FROM barang WHERE kode_barang = 'B001' OR 'atep'='atep'
BUILD SUCCESSFUL (total time: 2 seconds)
```

Ket : parameter kode_barang='B001' menunjukkan false atau data tidak valid karena datanya tidak ada sedangkan 'atep'='atep' menunjukkan true atau valid. Sehingga dengan operator OR maka

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

dapat diartikan bahwa kode SQL untuk menghapus data itu dapat dijalankan atau bisa dikatakan benar.

- 7) Untuk membuktikannya coba ubah kode programnya menjadi seperti dibawah ini.

```
package com.atep;
public class ProgramInjection {
    public static void main(String[] args) {
        barang barang = new barang();
        String kd="B001' OR 'atep='atep";
        barang.delete(kd);
        String sql = "DELETE FROM barang WHERE kode_barang='"+kd+"'";
        System.out.println(sql);
    }
}
```

- 8) Lalu tekan kombinasi **Shift+F6**, maka akan hasilnya seperti tampilan dibawah ini.

```
Output - DatabaseServices (run)
run:
Koneksi berhasil
Statement berhasil dibuat
Statement berhasil ditutup
DELETE FROM barang WHERE kode_barang = 'B001' OR 'atep='atep'
BUILD SUCCESSFUL (total time: 2 seconds)
```

- 9) Kemudian lihat data didalam databasenya.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	kode_barang	varchar(10)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	nama_barang	varchar(50)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	jumlah	int(11)			No	None		
<input type="checkbox"/>	harga	int(11)			No	None		

Check All / Uncheck All With selected:

Print view Relation view Propose table structure At End of Table At Beginning of Table After Go

MODUL 12

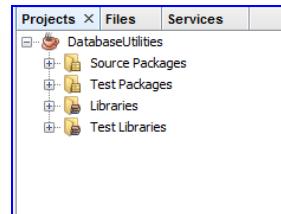
PREPARE STATEMENT

Selain menggunakan statement di java ada class prepare statement yang fungisnya sama dengan statement yaitu digunakan untuk memanipulasi data di database. Perbedaanya kalau statement tidak kebal terhadap SQLInjection sedangkan prepare statement mempunyai fasilitas untuk menangani SQLInjection. Jadi penggunaan prepare statement sangat aman ketika kita membuat query dengan banyak parameter.

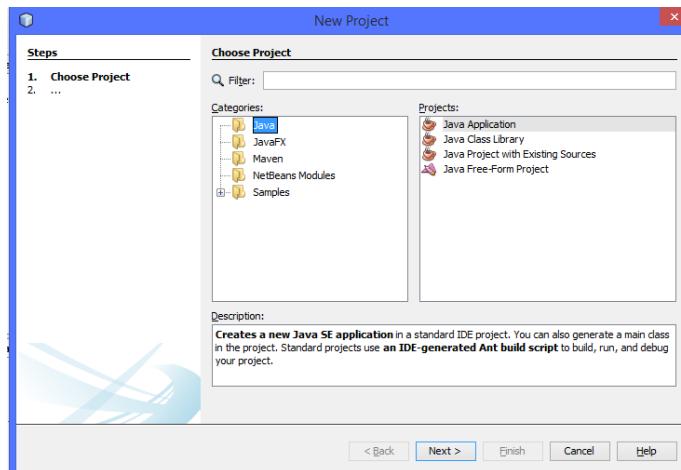
1. Membuat Prepare Statement

Cara untuk membuatnya prepare statemen hampir mirip dengan pembuatan statement, untuk lebih jelas caranya sebagai berikut.

- 1) Dalam praktikum ini kita memerlukan project yang telah dibuat sebelumnya yaitu **DatabaseUtilities**.

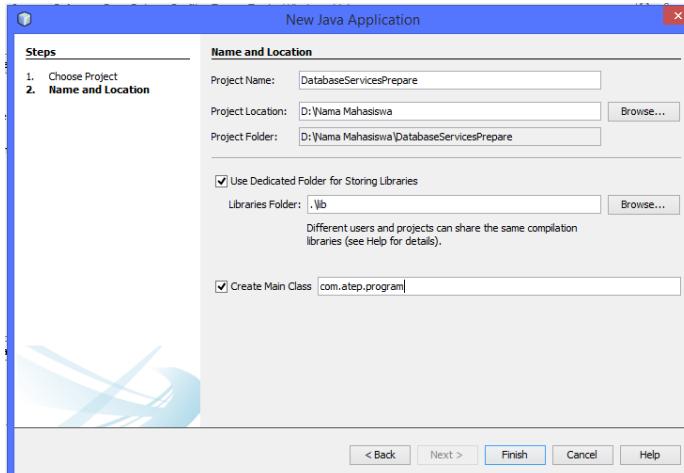


- 2) Buatlah project baru namun untuk tipe projectnya pilih **Java Java Application**.

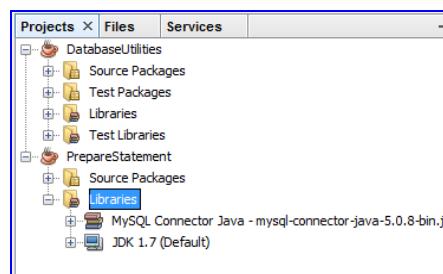


Object Oriented Programming (OOP)

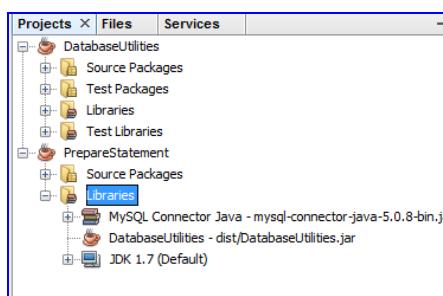
Guru : BEBEN SUTARA, S.Kom., M.T



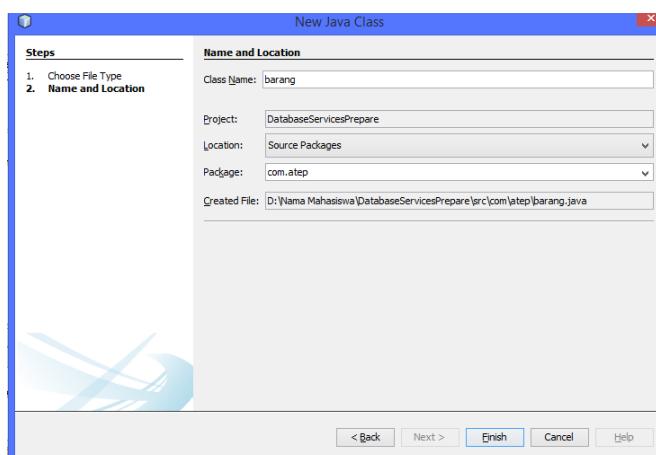
- 3) Kemudian tambahkan library MySQL Connector Java, dengan melakukan klik kanan pada **Libraries** pilih **Add library...** pada project **PreparedStatement**.



- 4) Kemudian tambahkan project **DatabaseUtilities**, dengan melakukan klik kanan pada **Libraries** pilih **Add project...** pada project **PreparedStatement**.



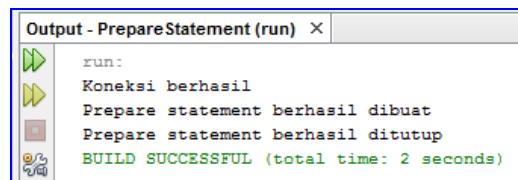
- 5) Pada project **DatabaseServicesUtilities** buatlah class baru (**New file – Java – Java Class**).



- 6) Kemudian ketikkan kode program seperti dibawah ini

```
package com.atep;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.SQLException;
public class program {
    public static void main(String[] args) {
        Connection koneksi = KoneksiDatabase.getKoneksi();
        PreparedStatement prepare = null;
        try{
            String sql="SELECT * FROM barang";
            prepare = koneksi.prepareStatement(sql);
            System.out.println("Prepare statement berhasil dibuat");
        }catch(SQLException ex){
            System.out.println("Prepare statement gagal dibuat");
            System.out.println("Pesan : "+ex.getMessage());
        }finally{
            if (prepare != null){
                try{
                    prepare.close();
                    System.out.println("Prepare statement berhasil ditutup");
                }catch(SQLException ex){
                    System.out.println("Pesan : "+ex.getMessage());
                }
            }
        }
    }
}
```

- 7) Coba jalankan program.



The screenshot shows the 'Output' window from an IDE during the execution of the 'PrepareStatement' run. The window title is 'Output - PrepareStatement (run)'. The output text is:
run:
Koneksi berhasil
Prepare statement berhasil dibuat
Prepare statement berhasil ditutup
BUILD SUCCESSFUL (total time: 2 seconds)

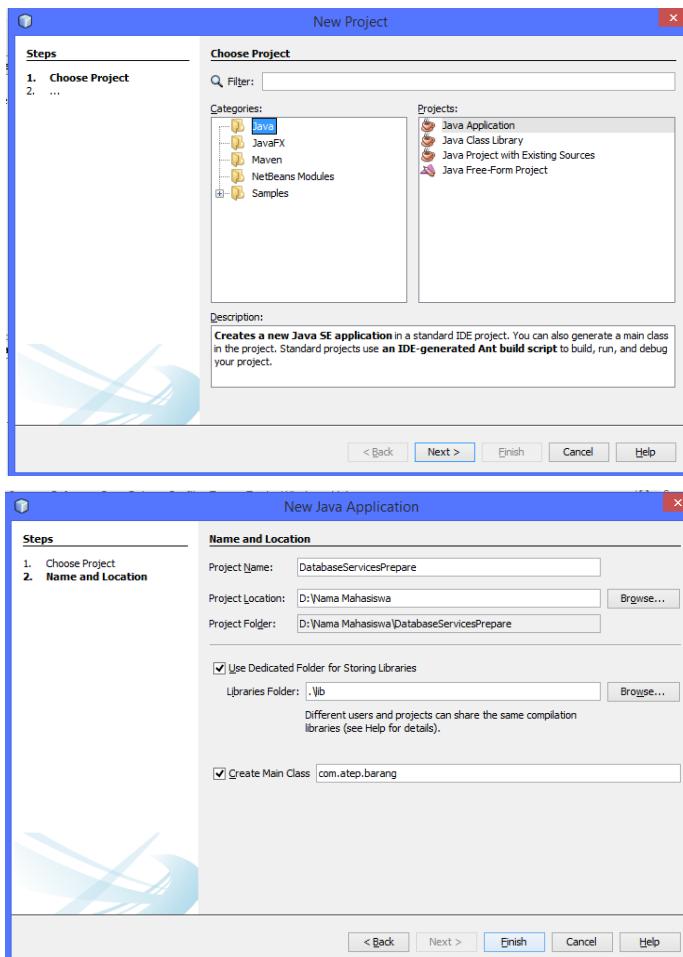
2. Membuat Database Service Prepare

Silahkan terlebih dahulu buatlah class **DatabaseServicePrepare**. Untuk lebih jelas caranya sebagai berikut :

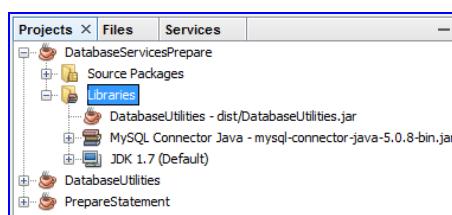
- 1) Buatlah project baru namun untuk tipe projectnya pilih **Java - Java Application**.

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



- 8) Kemudian tambahkan library **MySQL Connector Java**, dengan melakukan klik kanan pada **Libraries** pilih **Add library...** pada project **DatabaseServicesPrepare**.
- 9) Lalu tambahkan juga project **DatabaseUtilities**, dengan melakukan klik kanan pada **Libraries** pilih **Add project...** pada project **DatabaseServicesPrepare**.
- 10) Hasilnya akan seperti dibawah ini.



- 11) Ketikkan kode program seperti dibawah ini pada class barang.java

```
package com.atep;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.SQLException;  
  
public class barang {  
    private Connection koneksi;  
    public barang() {
```

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
koneksi = KoneksiDatabase.getKoneksi();
}

public void insert(String kd, String nm, int jml, long hrg){
    PreparedStatement prepare = null;
    try{
        String             sql="INSERT           INTO      barang
(kode_barang,nama_barang,jumlah,harga)VALUES(?, ?, ?, ?)";
        prepare = koneksi.prepareStatement(sql);
        System.out.println("Prepare statement berhasil dibuat");
        prepare.setString(1, kd);
        prepare.setString(2, nm);
        prepare.setInt(3, jml);
        prepare.setLong(4, hrg);
        prepare.executeUpdate();
    }catch(SQLException ex){
        System.out.println("Prepare statement gagal dibuat");
        System.out.println("Pesan : "+ex.getMessage());
    }finally{
        if (prepare != null){
            try{
                prepare.close();
                System.out.println("Prepare statement berhasil ditutup");
            }catch(SQLException ex){
                System.out.println("Pesan : "+ex.getMessage());
            }
        }
    }
}

public void update(String kd, String nm, int jml, long hrg){
    PreparedStatement prepare = null;
    try{
        String sql="UPDATE barang SET nama_barang=?,jumlah=?,harga=? WHERE
kode_barang=?";
        prepare = koneksi.prepareStatement(sql);
        System.out.println("Prepare statement berhasil dibuat");
        prepare.setString(1, nm);
        prepare.setInt(2, jml);
        prepare.setLong(3, hrg);
        prepare.setString(4, kd);
        prepare.executeUpdate();
    }
```

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
        }catch(SQLException ex){
            System.out.println("Prepare statement gagal dibuat");
            System.out.println("Pesan : "+ex.getMessage());
        }finally{
            if (prepare != null){
                try{
                    prepare.close();
                    System.out.println("Prepare statement berhasil ditutup");
                }catch(SQLException ex){
                    System.out.println("Pesan : "+ex.getMessage());
                }
            }
        }
    }

public void delete(String kd){
    PreparedStatement prepare = null;
    try{
        String sql="DELETE FROM barang WHERE Kode_barang=?";
        prepare = koneksi.prepareStatement(sql);
        System.out.println("Prepare statement berhasil dibuat");
        prepare.setString(1, kd);
        prepare.executeUpdate();
    }catch(SQLException ex){
        System.out.println("Prepare statement gagal dibuat");
        System.out.println("Pesan : "+ex.getMessage());
    }finally{
        if (prepare != null){
            try{
                prepare.close();
                System.out.println("Prepare statement berhasil ditutup");
            }catch(SQLException ex){
                System.out.println("Pesan : "+ex.getMessage());
            }
        }
    }
}
```

- 12) Lalu untuk mencobanya silahkan buka class **program.java** pada project **DatabaseServicesPrepare**. Kemudian ketikkan kode program dibawah ini.

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- a) **Insert** data (Shift+F6)

```
package com.atep;
public class program {
    public static void main(String[] args) {
        barang pensil = new barang();
        pensil.insert("P001", "PENSIL 2B", 20, 2000);
        pensil.insert("P002", "SPIDOL", 50, 1500);
        barang buku = new barang();
        buku.insert("B001", "BUKU TULIS", 30, 2500);
        buku.insert("B002", "BUKU GAMBAR", 40, 3000);
    }
}
```

HASIL :

	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B001	BUKU TULIS	30	2500
<input type="checkbox"/>	B002	BUKU GAMBAR	40	3000
<input type="checkbox"/>	P001	PENSIL 2B	20	2000
<input type="checkbox"/>	P002	SPIDOL	50	1500

- b) **Update** data (Shift+F6)

```
package com.atep;
public class program {
    public static void main(String[] args) {
        barang pensil = new barang();
        pensil.update("P001", "Pensil 2B", 20, 2000);
        pensil.update("P002", "Spidol", 50, 1500);
        barang buku = new barang();
        buku.update("B001", "Buku Tulis", 30, 2500);
        buku.update("B002", "Buku Gambar", 40, 3000);
    }
}
```

HASIL :

	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B001	Buku Tulis	30	2500
<input type="checkbox"/>	B002	Buku Gambar	40	3000
<input type="checkbox"/>	P001	Pensil 2B	20	2000
<input type="checkbox"/>	P002	Spidol	50	1500

- c) **Delete** data (Shift+F6)

```
package com.atep;
public class program {
```

```
public static void main(String[] args) {  
    barang pensil = new barang();  
    pensil.delete("P001");  
    barang buku = new barang();  
    buku.delete("B002");  
}  
}
```

HASIL :

	kode_barang	nama_barang	jumlah	harga
□	B001	Buku Tulis	30	2500
□	P002	Spidol	50	1500

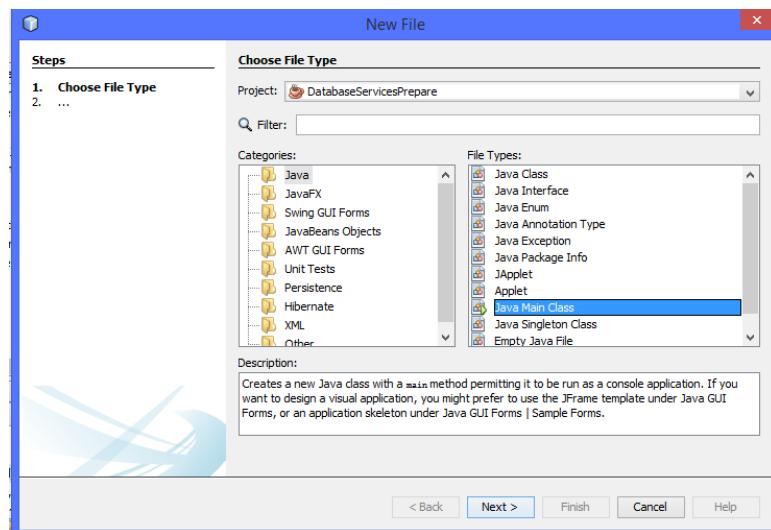
Check All / Uncheck All With selected:

Show : 30 row(s) starting from record # 0

3. SQLInjection di Prepare Statement

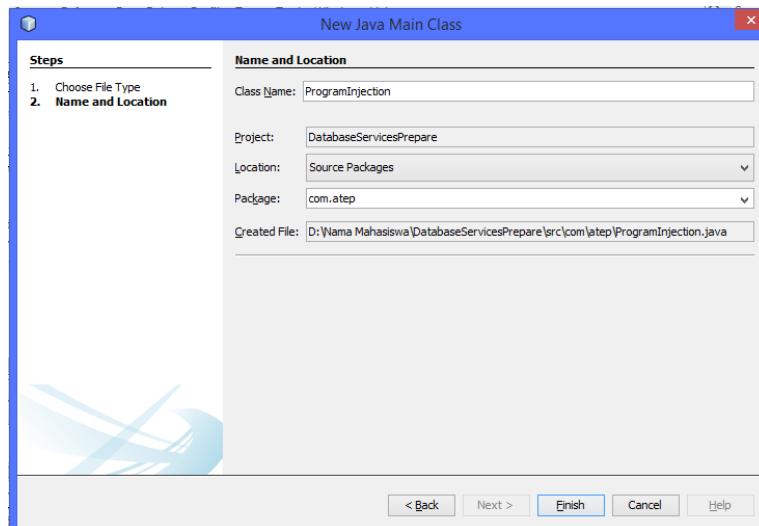
Untuk membuktikan bahwa penggunaan prepare statement tidak akan diserang orang SQLInjection maka kita coba lakukan hal yang sama seperti contoh sebelumnya. Sebagai contoh ikuti langkah berikut :

- 1) Buat file baru pada project **DatabaseServicesPrepare**, seperti dibawah ini pilih **Java – Java Main Class**.



Object Oriented Programming (OOP)

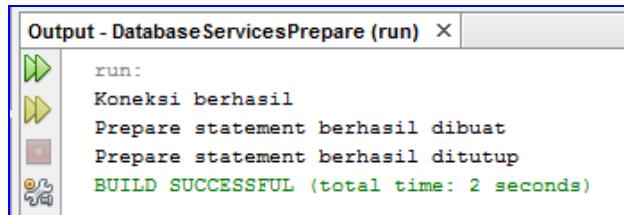
Guru : BEBEN SUTARA, S.Kom., M.T



- 2) Ketikkan kode program seperti dibawah ini.

```
package com.atep;
public class ProgramInjection {
    public static void main(String[] args) {
        barang barang = new barang();
        barang.delete("B001 OR 'atep'='atep");
    }
}
```

- 3) Coba jalankan program.



- 4) Kemudian cek pada database.

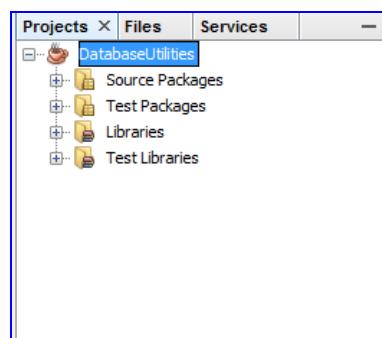
	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B001	Buku Tulis	30	2500
<input type="checkbox"/>	P002	Spidol	50	1500

Ket : terlihat data tidak hilang, walapun perintah tersebut sama seperti perintah sebelumnya dan ketika dijalankan berhasil tapi data yang ada pada database tetap aman.

MODUL 13

DATA ACCES OBJECT

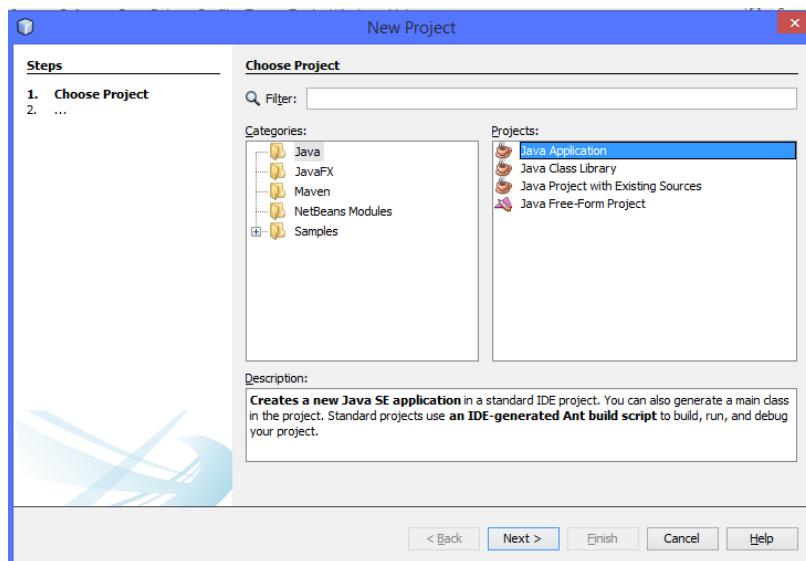
Setelah kita memahami dan mempraktekan integrasi antara oop dengan database MySQL, kita sekarang akan mencoba kembali mengulang dari awal langkah tersebut dengan cara yang berbeda namun untuk kode program tetap sama. Database dan table yang digunakan tetap **db_oop** dan **barang**. Dan Project yang digunakan masih menggunakan **DatabaseUtilities**.



1. Membuat Kelas Entity

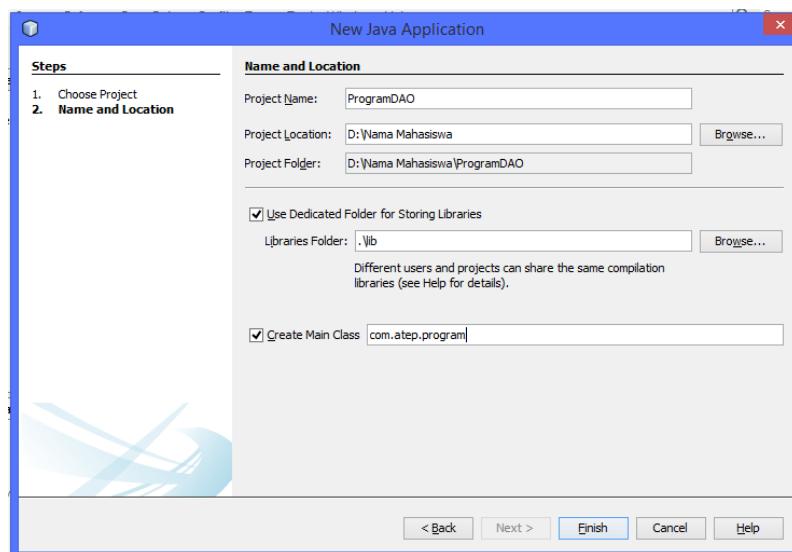
Sekarang kita akan membuat kelas entity dimana kelas ini harus sama dengan nama table yaitu barang. Caranya sebagai berikut :

- 1) Tambahkan project baru

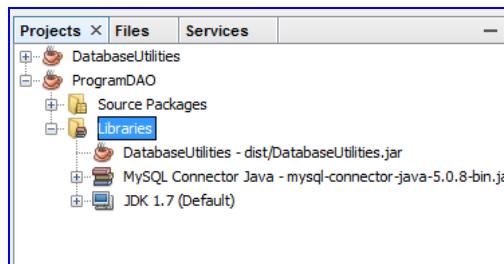


Object Oriented Programming (OOP)

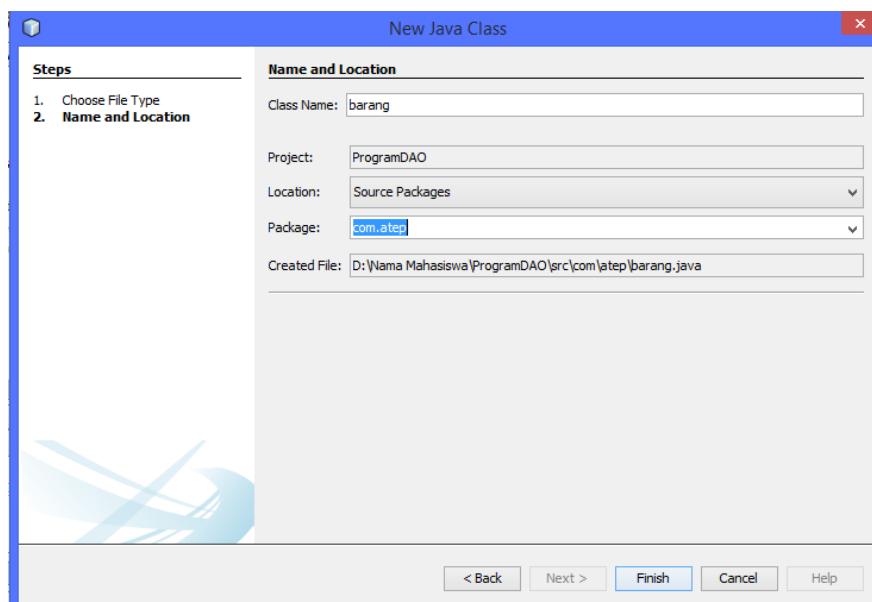
Guru : BEBEN SUTARA, S.Kom., M.T



- 2) Pada **Libraries** project **ProgramDAO** tambahkan project **DatabaseUtilities** dan **MySQL Connector Java**. Seperti tampilan dibawah ini.



- 3) Kemudian pada project **ProgramDAO** tambahkan file Java Class dan beri nama **barang**



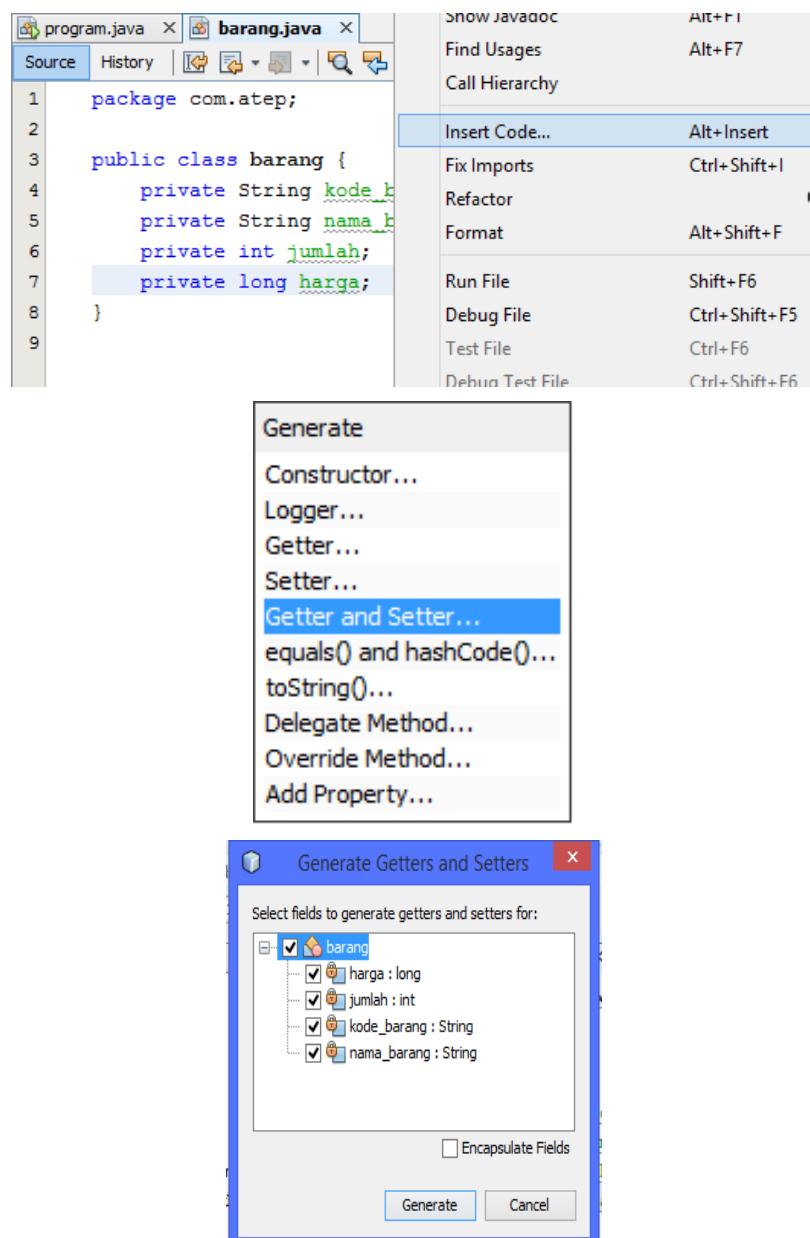
Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- 4) Kemudian ketiklah kode program dibawah ini.

```
package com.atep;
public class barang {
    private String kode_barang;
    private String nama_barang;
    private int jumlah;
    private long harga;
}
```

- 5) Kemudian pada area kode program klik kanan pilih **Insert Code** dan pilih **Getter and Setter**.



Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- 6) Hasilnya akan seperti dibawah ini.

```
package com.atep;
public class barang {
    private String kode_barang;
    private String nama_barang;
    private int jumlah;
    private long harga;

    public String getKode_barang() {
        return kode_barang;
    }
    public void setKode_barang(String kode_barang) {
        this.kode_barang = kode_barang;
    }

    public String getNama_barang() {
        return nama_barang;
    }
    public void setNama_barang(String nama_barang) {
        this.nama_barang = nama_barang;
    }

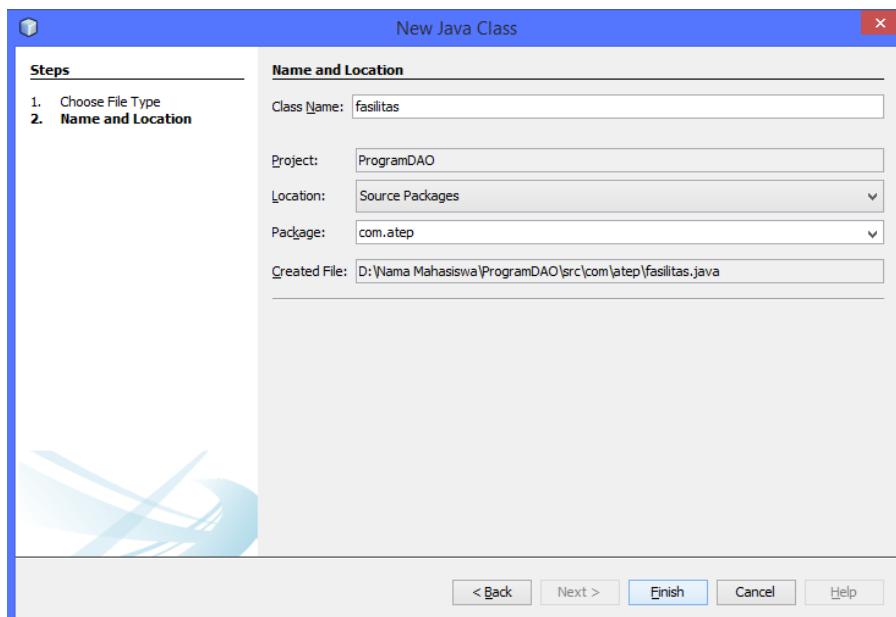
    public int getJumlah() {
        return jumlah;
    }
    public void setJumlah(int jumlah) {
        this.jumlah = jumlah;
    }

    public long getHarga() {
        return harga;
    }
    public void setHarga(long harga) {
        this.harga = harga;
    }
}
```

2. Membuat Kelas Fasilitas

Kelas fasilitas ini digunakan untuk menyimpan method-method untuk memanipulasi data misalkan insert, update, delete, dan selectAll. Untuk membuatnya ikuti langkah berikut :

- 1) Buatlah class baru yaitu **fasilitas**.



- 2) Lalu ketikkan kode program dibawah ini.

```
package com.atep;

import com.mysql.jdbc.PreparedStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class fasilitas {
    private Connection koneksi;
    public fasilitas() {
        koneksi = KoneksiDatabase.getKoneksi();
    }
    public void insert(barang barang){
        PreparedStatement prepare = null;
        try {
```

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
String          sql          ="INSERT      INTO      barang
(kode_barang,nama_barang,jumlah,harga)VALUES(?, ?, ?, ?)";
    prepare = (PreparedStatement) koneksi.prepareStatement(sql);
    prepare.setString(1, barang.getKode_barang());
    prepare.setString(2, barang.getNama_barang());
    prepare.setInt(3, barang.getJumlah());
    prepare.setLong(4, barang.getHarga());
    prepare.executeUpdate();
    System.out.println("Prepare statement berhasil dibuat");
}catch(SQLException ex){
    System.out.println("Prepare statement gagal dibuat");
    System.out.println("Pesanan : " + ex.getMessage());
}finally{
    if (prepare != null){
        try{
            prepare.close();
            System.out.println("Prepare statemen berhasil ditutup");
        }catch(SQLException ex){
            System.out.println("Prepare statemen gagal ditutup");
            System.out.println("Pesanan : " + ex.getMessage());
        }
    }
}

public void update(barang barang){
    PreparedStatement prepare = null;
    try {
        String sql = "UPDATE barang SET nama_barang=?,jumlah=?,harga=? WHERE
kode_barang=?";
        prepare = (PreparedStatement) koneksi.prepareStatement(sql);
        prepare.setString(1, barang.getNama_barang());
        prepare.setInt(2, barang.getJumlah());
        prepare.setLong(3, barang.getHarga());
        prepare.setString(4, barang.getKode_barang());
        prepare.executeUpdate();
    }
```

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
        System.out.println("Prepare statement berhasil dibuat");
    }catch(SQLException ex){
        System.out.println("Prepare statement gagal dibuat");
        System.out.println("Pesan : " + ex.getMessage());
    }finally{
        if (prepare != null){
            try{
                prepare.close();
                System.out.println("Prepare statemen berhasil ditutup");
            }catch(SQLException ex){
                System.out.println("Prepare statemen gagal ditutup");
                System.out.println("Pesan : " + ex.getMessage());
            }
        }
    }

}

public void delete(String kode_barang){
    PreparedStatement prepare = null;
    try {
        String sql ="DELETE FROM barang WHERE kode_barang=?";
        prepare = (PreparedStatement) koneksi.prepareStatement(sql);
        prepare.setString(1, kode_barang);
        prepare.executeUpdate();
        System.out.println("Prepare statement berhasil dibuat");
    }catch(SQLException ex){
        System.out.println("Prepare statement gagal dibuat");
        System.out.println("Pesan : " + ex.getMessage());
    }finally{
        if (prepare != null){
            try{
                prepare.close();
                System.out.println("Prepare statemen berhasil ditutup");
            }catch(SQLException ex){
                System.out.println("Prepare statemen gagal ditutup");
                System.out.println("Pesan : " + ex.getMessage());
            }
        }
    }
}
```

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
        }
    }
}

public List<barang> selectAll(){
    PreparedStatement prepare = null;
    ResultSet result = null;
    List<barang> list = new ArrayList<>();
    try {
        String sql ="SELECT * FROM barang";
        prepare = (PreparedStatement) koneksi.prepareStatement(sql);
        result = prepare.executeQuery();
        while (result.next()){
            barang barang = new barang();
            barang.setNama_barang(result.getString("nama_barang"));
            barang.setJumlah(result.getInt("jumlah"));
            barang.setHarga(result.getLong("harga"));
            list.add(barang);
        }
        System.out.println("Prepare statement berhasil dibuat");
        return list;
    }catch(SQLException ex){
        System.out.println("Prepare statement gagal dibuat");
        System.out.println("Pesan : " + ex.getMessage());
        return list;
    }finally{
        if (prepare != null){
            try{
                prepare.close();
                System.out.println("Prepare statemen berhasil ditutup");
            }catch(SQLException ex){
                System.out.println("Prepare statemen gagal ditutup");
                System.out.println("Pesan : " + ex.getMessage());
            }
        }
        if (result != null){
```

```
        try{
            result.close();
            System.out.println("Resultset berhasil ditutup");
        }catch(SQLException ex){
            System.out.println("Resultset gagal ditutup");
            System.out.println("Peseran : " + ex.getMessage());
        }
    }
}
```

3. Menggunakan Service

Setelah kerangka class nya kita buat, selanjutnya kita akan coba hasilnya, ikutilah langkah berikut :

- 1) Cek dulu data dalam database. Misalkan data sebelumnya seperti dibawah ini.

	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B001	Buku Tulis	30	2500
<input type="checkbox"/>	P002	Spidol	50	1500

Check All / Uncheck All With selected:
Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after

- 2) Buka class program.java, ketikkan kode program dibawah ini.

- a) **Insert**

```
package com.atep;
public class program {
    public static void main(String[] args) {
        fasilitas perintah = new fasilitas();
        barang atk = new barang();
        atk.setKode_barang("B002");
        atk.setNama_barang("Buku Gambar");
        atk.setJumlah(12);
        atk.setHarga(3000);
        perintah.insert(atk);
    }
}
```

HASIL :

	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B001	Buku Tulis	30	2500
<input type="checkbox"/>	B002	Buku Gambar	12	3000
<input type="checkbox"/>	P002	Spidol	50	1500

Check All / Uncheck All With selected:
 Show : 30 row(s) starting from record # 0

b) **Update**

```
package com.atep;
public class program {
    public static void main(String[] args) {
        fasilitas perintah = new fasilitas();
        barang atk = new barang();
        atk.setKode_barang("B002");
        atk.setNama_barang("BUKU GAMBAR");
        atk.setJumlah(12);
        atk.setHarga(3000);
        perintah.update(atk);
    }
}
```

HASIL :

	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B001	Buku Tulis	30	2500
<input type="checkbox"/>	B002	BUKU GAMBAR	12	3000
<input type="checkbox"/>	P002	Spidol	50	1500

Check All / Uncheck All With selected:
 Show : 30 row(s) starting from record # 0

c) **Delete**

```
package com.atep;
public class program {
    public static void main(String[] args) {
        fasilitas perintah = new fasilitas();
        perintah.delete("B002");
    }
}
```

HASIL :

	kode_barang	nama_barang	jumlah	harga
<input type="checkbox"/>	B001	Buku Tulis	30	2500
<input type="checkbox"/>	P002	Spidol	50	1500

Check All / Uncheck All With selected:

Show : 30 row(s) starting from record # 0

d) **SelectAll**

```
package com.atep;
import java.util.List;
public class program {
    public static void main(String[] args) {
        fasilitas perintah = new fasilitas();
        List<barang> list = perintah.selectAll();
        for(barang barang : list){
            System.out.println(barang.getKode_barang());
            System.out.println(barang.getNama_barang());
            System.out.println(barang.getJumlah());
            System.out.println(barang.getHarga());
            System.out.println("-----");
        }
    }
}
```

HASIL :

```
Output - ProgramDAO (run) ×
run:
Koneksi berhasil
Prepare statement berhasil dibuat
Prepare statemen berhasil ditutup
Resultset berhasil ditutup
P002
Spidol
50
1500
-----
B001
Buku Tulis
30
2500
```

MODUL 14

GUI

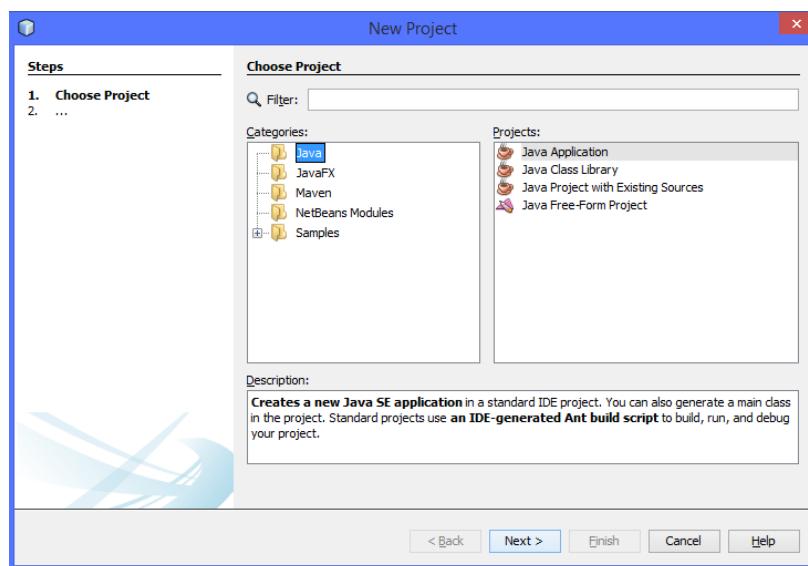
Pada modul ini akan dibahas tentang GUI (*Graphic User Interface*), dimana kita akan membuat aplikasi yang mempunyai antar muka sehingga lebih menarik untuk dilihat dan digunakan. Untuk praktikum pada modul ini masih menggunakan database dan table yang sama dengan praktikum sebelumnya dan menggunakan project **DatabaseUtilities** dan **ProgramDAO**.



1. Membuat Project Aplikasi

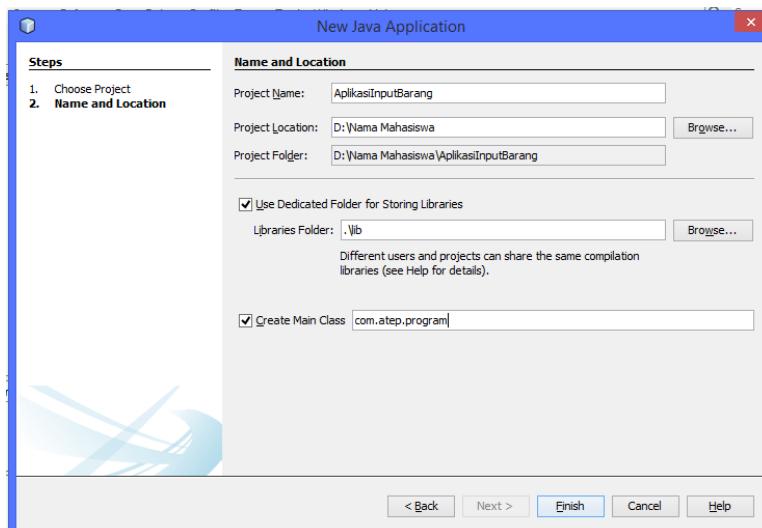
Untuk memulai praktikum membuat aplikasi berbasis GUI maka ikutilah langkah berikut :

- 1) Buatlah project baru.

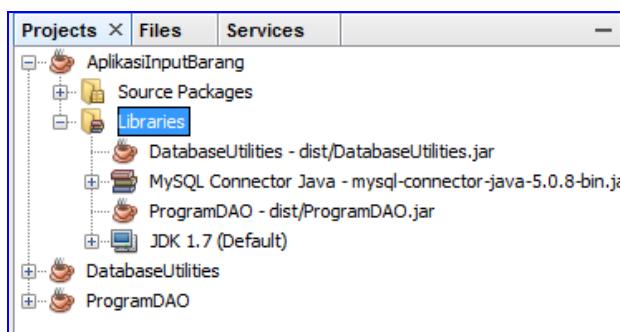


Object Oriented Programming (OOP)

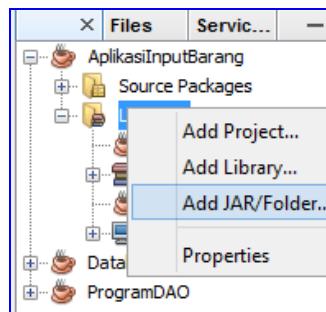
Guru : BEBEN SUTARA, S.Kom., M.T



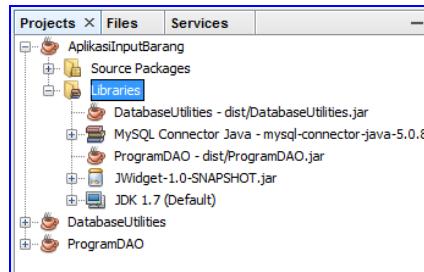
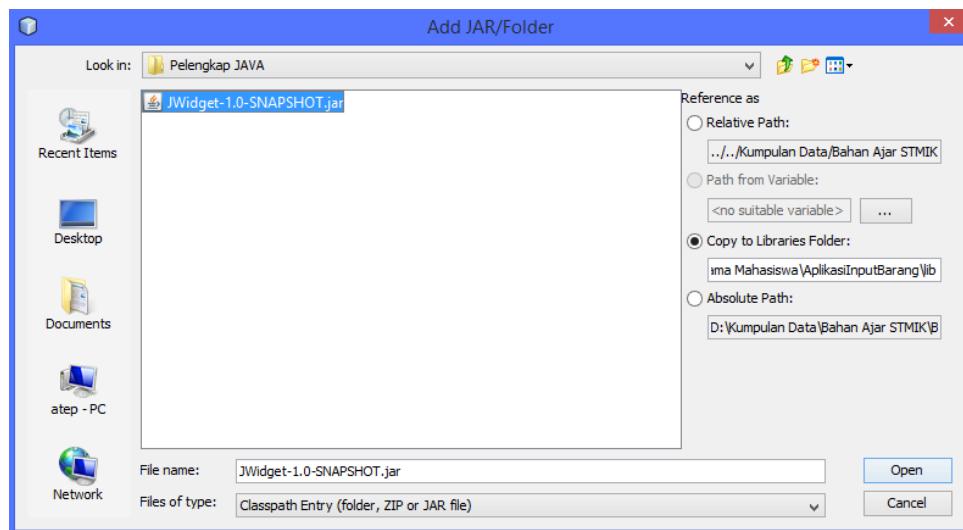
- 2) Jangan lupa tambahkan pada **Libraries** project **AplikasiInputBarang** yaitu project **DatabaseUtilities** dan **ProgramDAO** serta **MySQL Connector Java**.



- 3) Tambahkan juga pada **Libraries** file **JWidget**. Caranya sama namun yang dipilih **Add JAR/Folder..**



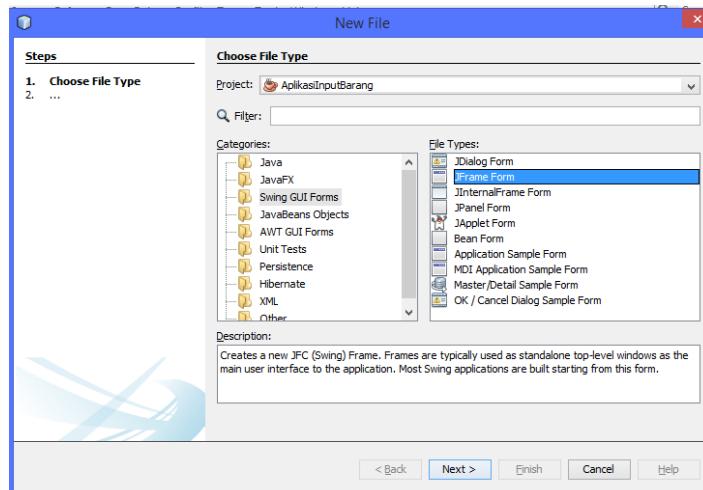
- 4) Cari dimana menyimpan file tersebut kemudian pilih dan klik **Open**.



2. Membuat Tampilan Aplikasi

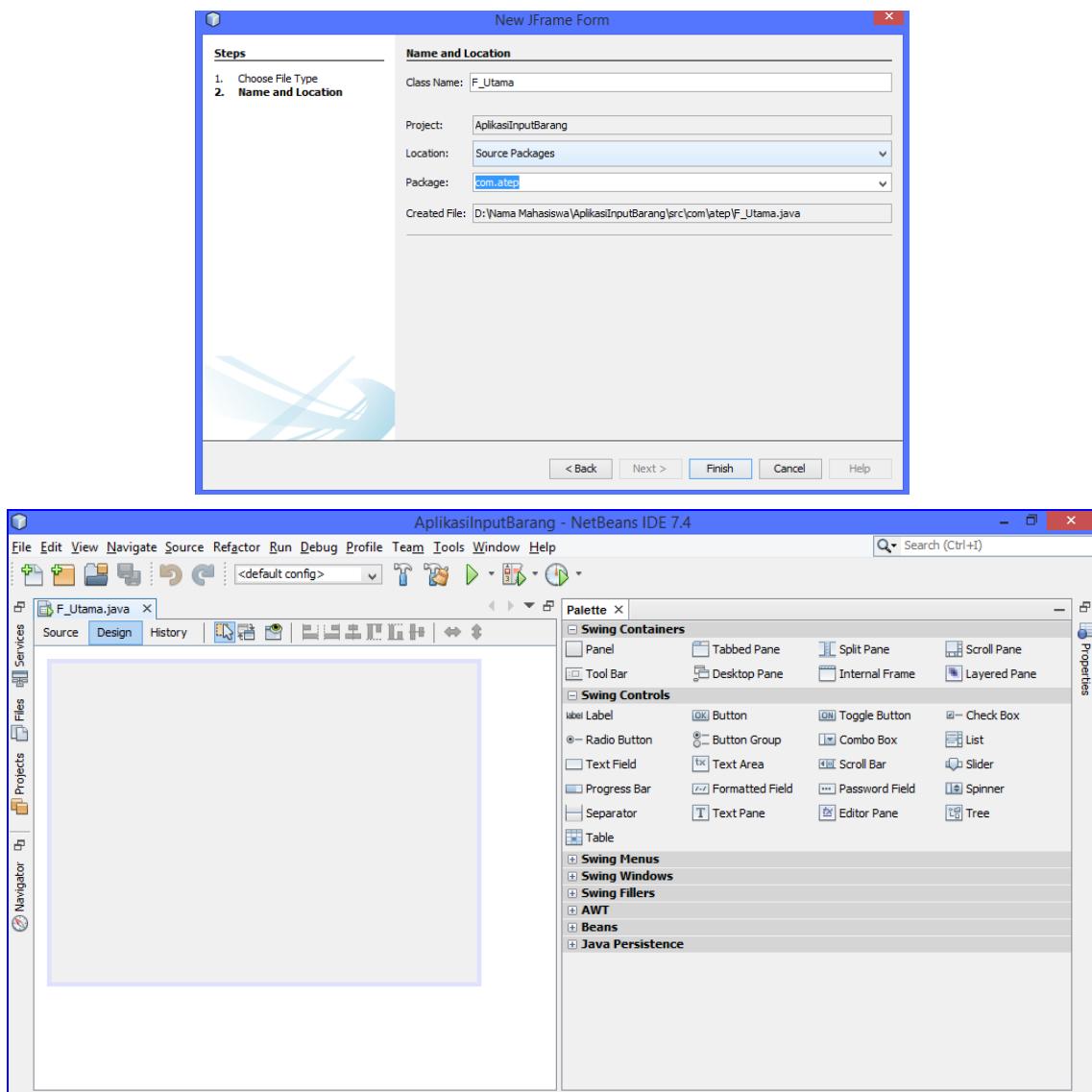
Sekarang kita akan memulai membuat tampilan form, caranya sebagai berikut :

- 1) Buatlah file baru, pilih **Swing GUI Forms – JFrameForm**.



Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



- 2) Silahkan tambahkan komponen yaitu **Label**, **ScrollPane**, dan **Button**. Seperti tampilan berikut



Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- 3) Ubahkan properties/code beberapa komponen, ketentuannya sebagai berikut :

No	Component	Properties/Code	Value
1	JLabel1	text	INPUT DATA BARANG
		font	Tahoma 18 Bold
2	JScrollPane1	Variable Name	sp_daftarbarang
		text	Tambah Data Baru
3	JButton1	Variable Name	btn_tambah
		text	Ubah Data Terseleksi
4	JButton2	Variable Name	btn_ubah
		text	Hapus Data Terseleksi
5	JButton3	Variable Name	btn_hapus
		text	Keluar Dari Aplikasi
6	JButton4	Variable Name	btn_keluar
		text	

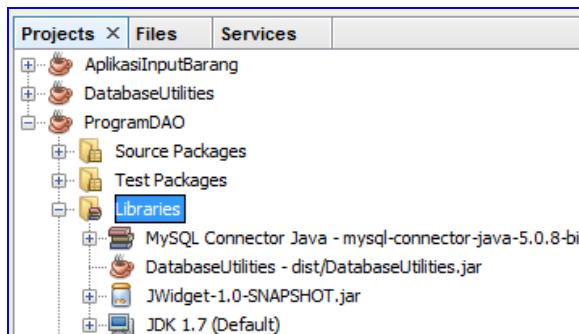
- 4) Sehingga tampilannya menjadi seperti dibawah ini.



3. Membuat Menampilkan Data dari MySQL ke Aplikasi

Sekarang kita akan mencoba menampilkan data yang ada pada database MySQL kedalam tampilan aplikasi yang telah kita buat. Caranya sebagai berikut :

- 1) Pada project **ProgramDAO** tambahkan **Libraries** file **JWidget**.



- 2) Buka class **barang.java** yang ada pada project **ProgramDAO** lalu tambahkan kode program seperti dibawah ini.

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
package com.atep;
import com.stripbandunk.jwidget.annotation.TableColumn;
public class barang {
    @TableColumn(number=1, name ="KODE BARANG")
    private String kode_barang;
    @TableColumn(number=2, name ="NAMA BARANG")
    private String nama_barang;
    @TableColumn(number=3, name ="JUMLAH")
    private int jumlah;
    @TableColumn(number=1, name ="HARGA")
    private long harga;

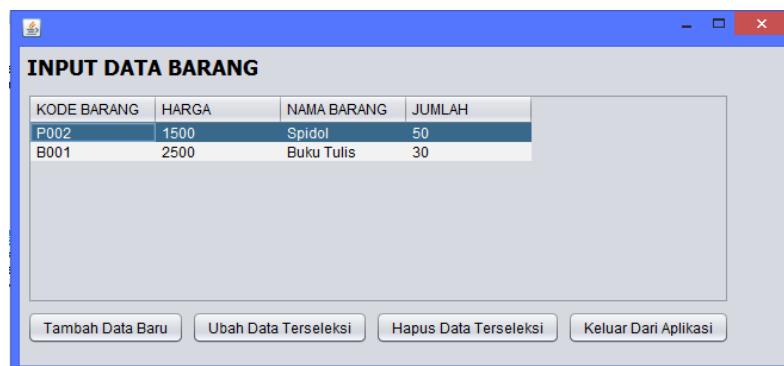
    public String getKode_barang() {
        return kode_barang;
    }
    public void setKode_barang(String kode_barang) {
        this.kode_barang = kode_barang;
    }
    public String getNama_barang() {
        return nama_barang;
    }
    public void setNama_barang(String nama_barang) {
        this.nama_barang = nama_barang;
    }
    public int getJumlah() {
        return jumlah;
    }
    public void setJumlah(int jumlah) {
        this.jumlah = jumlah;
    }
    public long getHarga() {
        return harga;
    }
    public void setHarga(long harga) {
        this.harga = harga;
    }
}
```

```
}
```

- 3) Buka class **F_Utama.java** pada project **AplikasiInputBarang**. Buka source-nya cari class yang dimaksud kemudian tambahkan kode seperti dibawah ini.

```
public class F_Utama extends javax.swing.JFrame {  
    private DynamicTableModel<barang> tableModel;  
    private JDynamicTable jDynamicTable;  
    public F_Utama() {  
        initComponents();  
        tableModel = new DynamicTableModel<>(barang.class);  
        jDynamicTable = new JDYNAMICTable(tableModel);  
        sp_daftarbarang.setViewportView(jDynamicTable);  
        reload();  
    }  
    private fasilitas perintah = new fasilitas();  
    private void reload() {  
        List<barang> list = perintah.selectAll();  
        for (barang barang : list){  
            tableModel.add(barang);  
        }  
    }  
}
```

- 4) Untuk melihat hasilnya tekan kombinasi **Shift+F6**



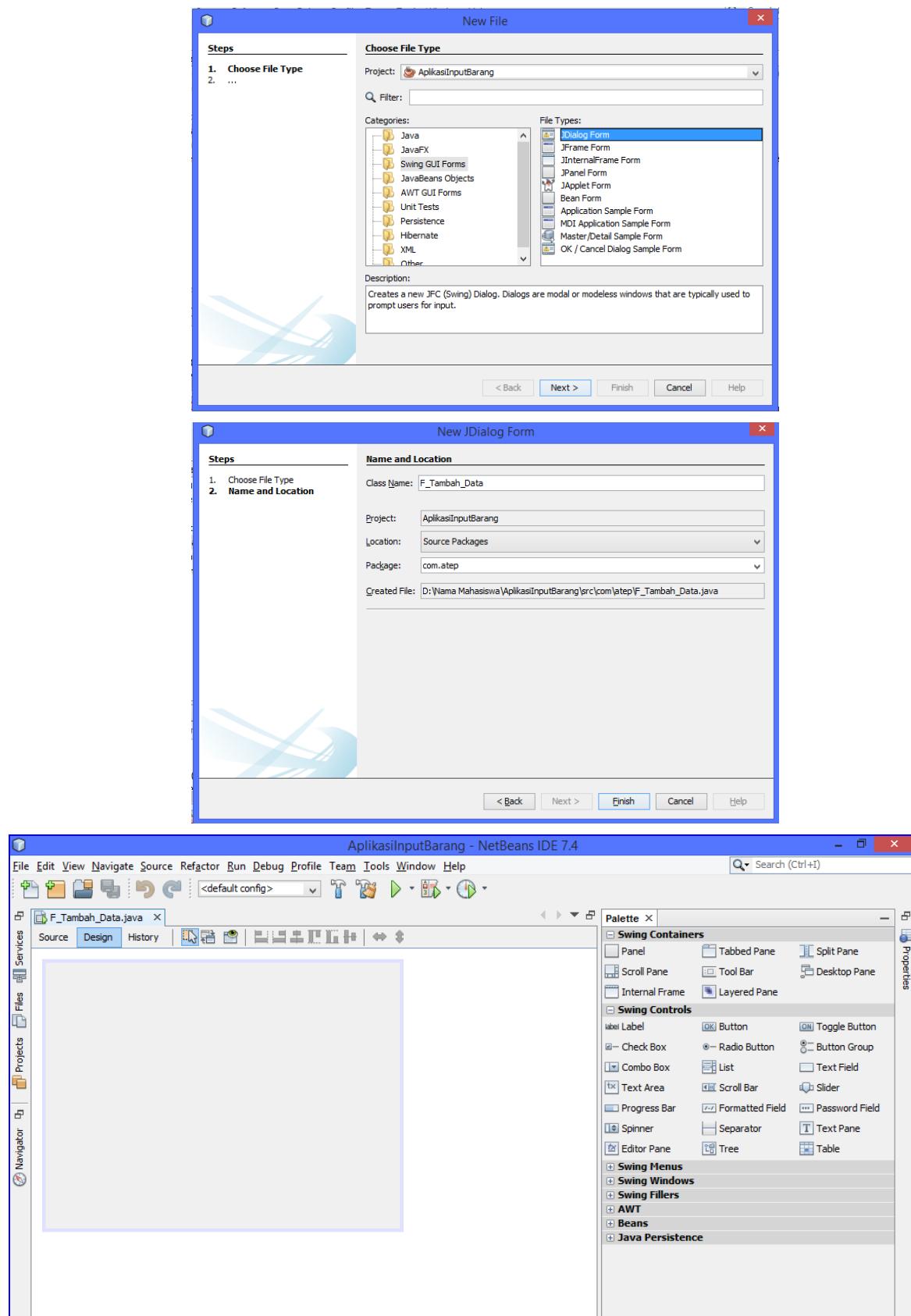
4. Menyimpan Data Dari Aplikasi ke MySQL

Untuk membuat fasilitas tambah data baru dalam hal ini yaitu simpan data dapat mengikuti langkah berikut :

- 1) Buatlah file baru seperti dibawah ini.

Object Oriented Programming (OOP)

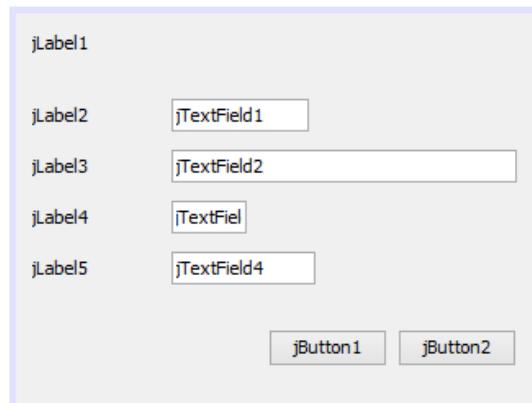
Guru : BEBEN SUTARA, S.Kom., M.T



Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- 2) Tambahkan beberapa komponen yaitu **Label**, **Text Field**, dan **Button**. Seperti tampilan dibawah ini.



- 3) Kemudian atur properties/codenya, sesuaikan dengan ketentuan dibawah ini.

No	Component	Properties/Code	Value
1	JLabel1	text	INPUT DATA BARANG
		font	Tahoma 18 Bold
2	JLabel2	text	KODE BARANG
3	JLabel3	text	NAMA BARANG
4	JLabel4	text	JUMLAH
5	JLabel5	text	HARGA
6	JTextField1	text	<kosongkan>
		Variable Name	txt_kode
7	JTextField2	text	<kosongkan>
		Variable Name	txt_nama
8	JTextField3	text	<kosongkan>
		Variable Name	txt_jumlah
9	JTextField4	text	<kosongkan>
		Variable Name	txt_harga
10	JButton1	text	SIMPAN DATA
		Variable Name	btn_simpan
11	JButton2	text	BATAL
		Variable Name	btn_batal

The image shows the final state of the Java Swing application. The window title is 'INPUT DATA BARANG'. Inside, there are four pairs of labels and text fields. The labels are 'KODE BARANG', 'NAMA BARANG', 'JUMLAH', and 'HARGA'. Each label is positioned above its respective text field. At the bottom right of the window are two buttons labeled 'SIMPAN DATA' and 'BATAL'.

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- 4) Selanjutnya pada class **F_Tambah_Data.java** buka source nya. Cari class **F_Tambah_Data** kemudian tambahkan kode program dibawah ini.

```
public class F_Tambah_Data extends javax.swing.JDialog {  
    public F_Tambah_Data(java.awt.Frame parent, boolean modal) {  
        super(parent, modal);  
        initComponents();  
    }  
    public void tambahBarang () {  
        setVisible(true);  
        setLocationRelativeTo(getParent());  
    }  
}
```

- 5) Kembali ke tampilan design, double klik tombol **SIMPAN DATA**, ketikkan kode program dibawah ini.

```
private void btn_simpanActionPerformed(java.awt.event.ActionEvent evt) {  
    barang barang = new barang();  
    barang.setKode_barang(txt_kode.getText());  
    barang.setNama_barang(txt_nama.getText());  
    barang.setJumlah(Integer.parseInt(txt_jumlah.getText()));  
    barang.setHarga(Long.parseLong(txt_harga.getText()));  
    fasilitas perintah = new fasilitas();  
    perintah.insert(barang);  
    dispose();  
}
```

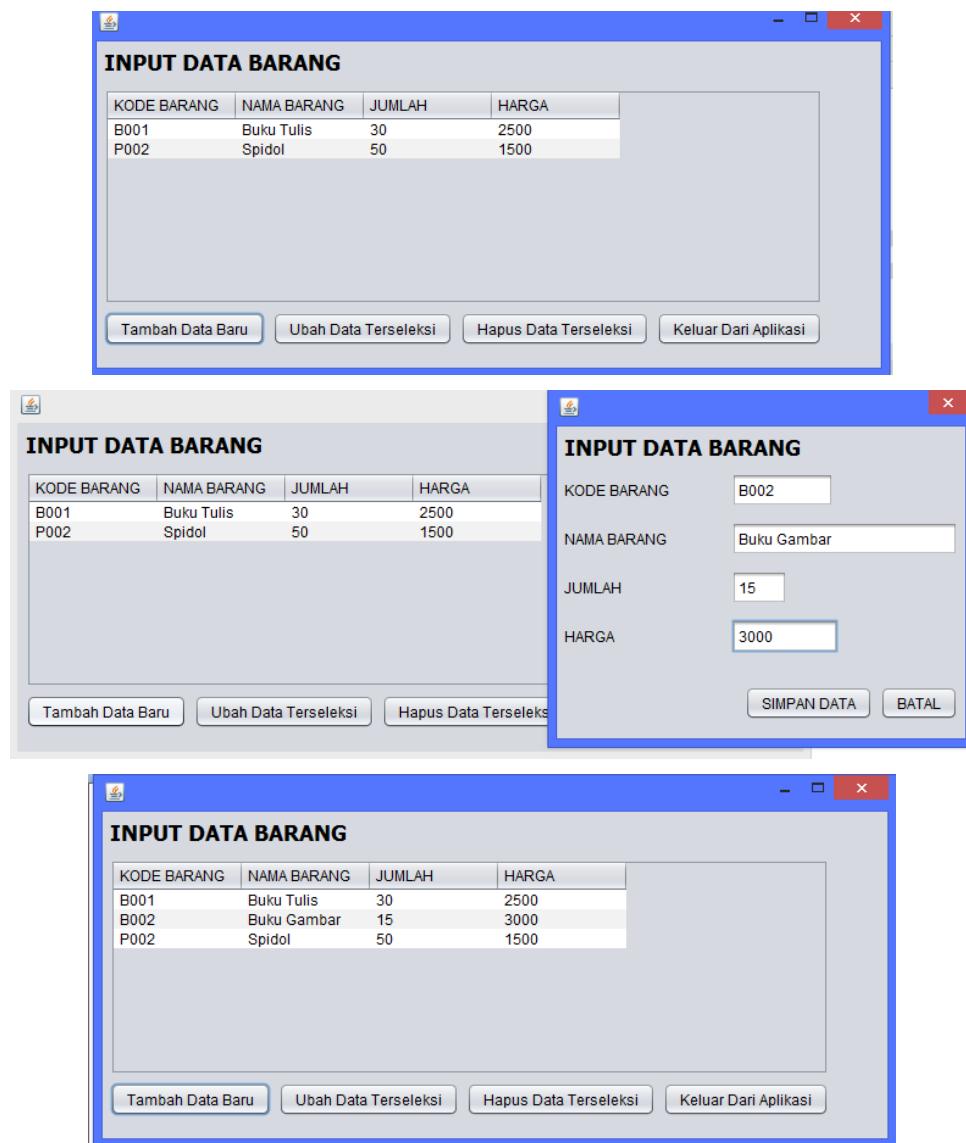
- 6) Double klik di tombol **BATAL**. Ketikkan kode program dibawah ini.

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

- 7) Buka class **F_Utama**, double klik pada tombol **Tambah Data Baru**. Lalu ketikkan kode program dibawah ini.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    F_Tambah_Data tambah = new F_Tambah_Data(this, true);  
    tambah.tambahBarang();  
    reload();  
}
```

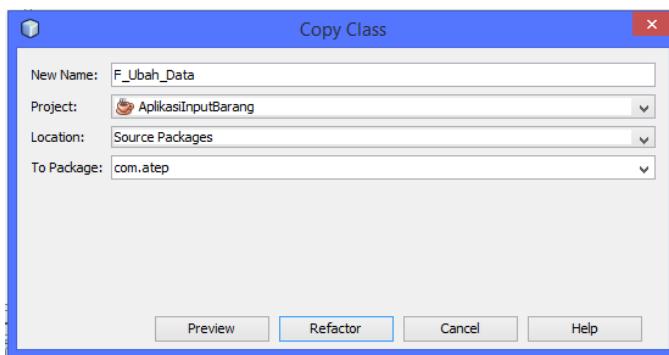
- 8) Untuk menjalankan programnya silahkan tekan kombinasi tombol **Shift+F6** pada class **F_Utama.java**. Maka akan muncul Form Utama kemudian klik tombol **Tambah Data Baru**, lalu isi datanya kalau sudah terisi semua klik tombol **SIMPAN**.



5. Mengubah Data Dari Aplikasi ke MySQL

Ketika data yang sudah dimasukan mengalami kesalahan maka perlu adanya perubahan untuk diperbaiki dan disimpan kembali. Untuk membuat fasilitas ini, ikuti langkah berikut :

- 1) Biar lebih cepat copykan class **F_Tambah_Data.java** dan copy-an nya beri nama **F_Ubah_Data.java**.



- 2) Untuk tampilannya silahkan ubah sesuaikan dengan form yang akan kita buat yaitu **UBAH DATA BARANG**.

- 3) Lakukan perubahan dan penambahan dalam kode programnya juga, silahkan sesuaikan.

a) Pada **class**

```
public class F_Ubah_Data extends javax.swing.JDialog {  
    private barang barang;  
    public F_Ubah_Data(java.awt.Frame parent, boolean modal) {  
        super(parent, modal);  
        initComponents();  
    }  
    public void ubahBarang (barang brg) {  
        this.barang = brg;  
        txt_kode.setText(barang.getKode_barang());  
        txt_nama.setText(barang.getNama_barang());  
        txt_jumlah.setText(String.valueOf(barang.getJumlah()));  
        txt_harga.setText(String.valueOf(barang.getHarga()));  
    }  
}
```

```
        setVisible(true);
        setLocationRelativeTo(getParent());
    }
```

- b) Pada tombol **SIMPAN**

```
private void btn_simpanActionPerformed(java.awt.event.ActionEvent evt) {
    barang.setKode_barang(txt_kode.getText());
    barang.setNama_barang(txt_nama.getText());
    barang.setJumlah(Integer.parseInt(txt_jumlah.getText()));
    barang.setHarga(Long.parseLong(txt_harga.getText()));
    fasilitas perintah = new fasilitas();
    perintah.update(barang);
    dispose();
}
```

- 4) Selanjutnya buka class **F_Utama.java**. Double klik pada tombol **Ubah Data Terseleksi**. Ketikan kode program dibawah ini.

```
private void btn_ubahActionPerformed(java.awt.event.ActionEvent evt) {
    if (jDynamicTable.getSelectedRow() != -1){
        int index =
jDynamicTable.convertRowIndexToModel(jDynamicTable.getSelectedRow());
        barang barang = tableModel.get(index);
        F_Ubah_Data ubah = new F_Ubah_Data(this, true);
        ubah.ubahBarang(barang);
        reload();
    }else {
        JOptionPane.showMessageDialog(this, "Silahkan pilih dulu datanya");
    }
}
```

- 9) Untuk menjalankan programnya silahkan tekan kombinasi tombol **Shift+F6** pada class **F_Utama.java**. Maka akan muncul Form Utama kemudian pilih dulu data yang akan dirubah kemudian klik tombol **Ubah Data Terseleksi**. Jika langsung menekan tombol Ubah Data Terseleksi maka akan muncul kotak dialog pemberitahuan.

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

The image displays four screenshots of a Windows application titled "INPUT DATA BARANG". The application features a table with columns: KODE BARANG, NAMA BARANG, JUMLAH, and HARGA. The data is as follows:

KODE BARANG	NAMA BARANG	JUMLAH	HARGA
B001	Buku Tulis	30	2500
B002	Buku Gambar	15	3000
P002	Spidol	50	1500

Below the table are four buttons: Tambah Data Baru, Ubah Data Terseleksi, Hapus Data Terseleksi, and Keluar Dari Aplikasi.

In the second screenshot, the "Ubah Data Terseleksi" button is clicked, opening a modal window titled "UBAH DATA BARANG". This window contains fields for KODE BARANG (P002), NAMA BARANG (Pensil 2B), JUMLAH (50), and HARGA (1500). It includes "SIMPAN DATA" and "BATAL" buttons.

In the third screenshot, the data has been updated, and the table now shows:

KODE BARANG	NAMA BARANG	JUMLAH	HARGA
B001	Buku Tulis	30	2500
B002	Buku Gambar	15	3000
P002	Pensil 2B	50	1500

In the fourth screenshot, a message dialog box titled "Message" appears, containing the text "Silahkan pilih dulu datanya" (Please select the data first) with an information icon. The "OK" button is visible at the bottom right of the dialog.

6. Menghapus Data Dari Aplikasi ke MySQL

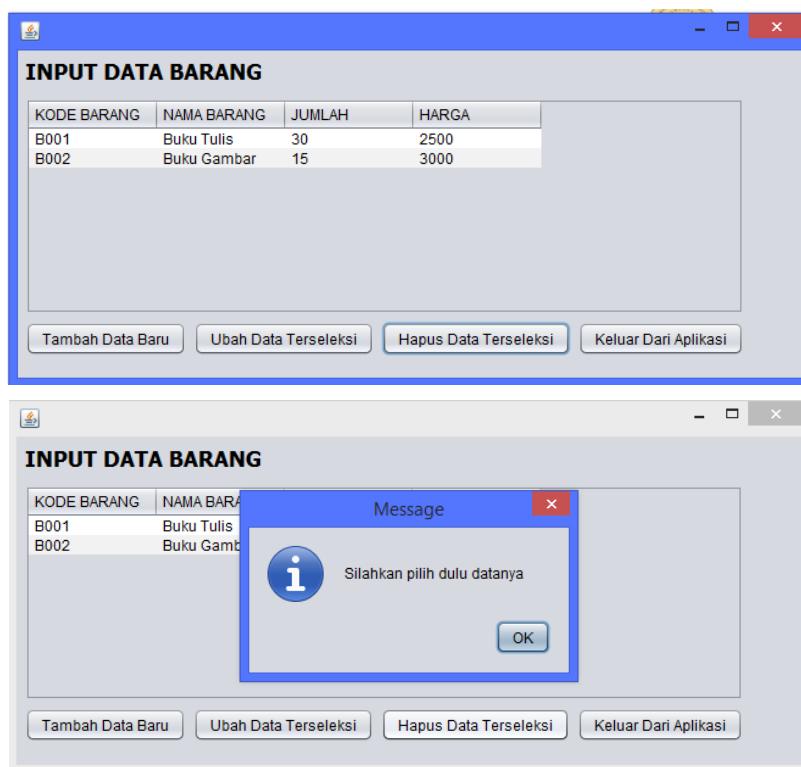
Langkah selanjutnya yaitu diperlukan juga fasilitas untuk hapus data. Caranya sebagai berikut :

- 1) Pada class **F_Utama.java**, double klik tombol **Hapus Data Terseleksi**, ketikkan kode program dibawah ini.

```
private void btn_hapusActionPerformed(java.awt.event.ActionEvent evt) {  
    if (jDynamicTable.getSelectedRow() != -1){  
        if (JOptionPane.showConfirmDialog(this, "Anda yakin akan menghapus  
data ini?", "Hapus Data", JOptionPane.OK_CANCEL_OPTION) == JOptionPane.OK_OPTION){  
            int index =  
            jDynamicTable.convertRowIndexToModel(jDynamicTable.getSelectedRow());  
            barang barang = tableModel.get(index);  
            perintah.delete(barang.getKode_barang());  
            reload();  
        }  
    }else{  
        JOptionPane.showMessageDialog(this, "Silahkan pilih dulu datanya");  
    }  
}
```

- 2) Untuk menjalankan programnya silahkan tekan kombinasi tombol **Shift+F6** pada class **F_Utama.java**. Maka akan muncul Form Utama kemudian pilih dulu data yang akan dihapus kemudian klik tombol **Hapus Data Terseleksi**. Maka akan muncul pesan pemberitahuan tentang data yang akan dihapus. Jika langsung menekan tombol **Hapus Data Terseleksi** maka akan muncul kotak dialog pemberitahuan.





7. Keluar Dari Aplikasi

Untuk keluar dari aplikasi kode programnya yaitu menggunakan dispose (). Caranya double klik di tombol Keluar Dari Aplikasi dan silahkan ketikkan kode program dibawah ini.

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

MODUL 15

MEMBUAT REPORT

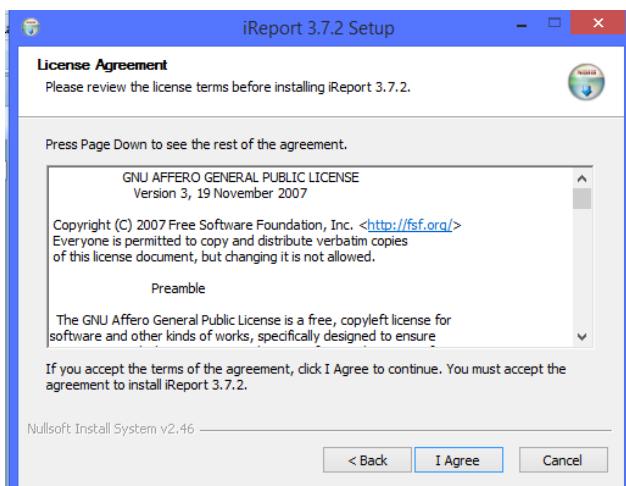
Selanjutnya kita akan mencoba membuat report, dimana report ini merupakan fasilitas yang sangat penting dalam sebuah aplikasi yaitu untuk melihat dan mencetak informasi. Dalam praktikum ini fasilitas untuk membuat report yang digunakan yaitu menggunakan **iReport-5.5.0-plugin**.

1. Menginstall iReport-3.7.2

Pertama-tama kita akan menginstall **iReport-3.7.2** sebagai media untuk membuat reportnya. Untuk lebih jelas silahkan ikuti langkah berikut : Silahkan double klik **iReport-3.7.2-windows-installer.exe**, lalu ikuti langkah berikut :



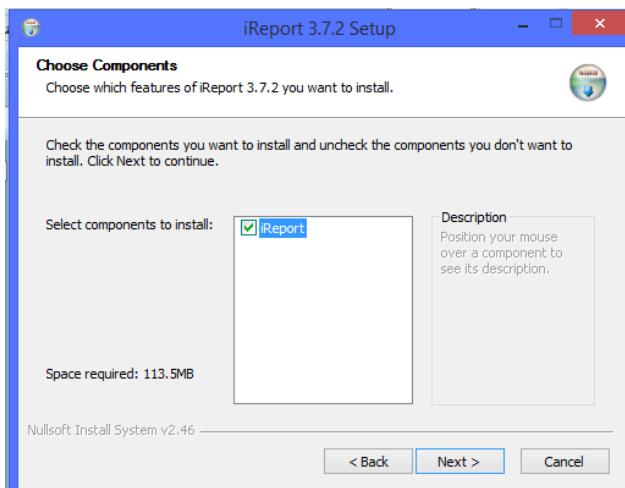
Klik tombol **Next**



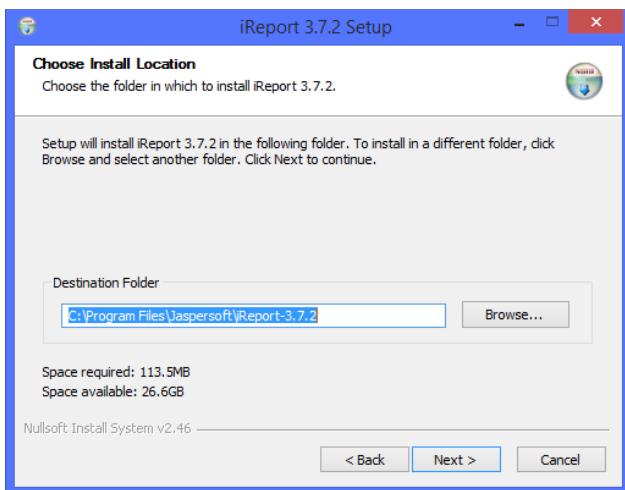
Klik tombol **I Agree**

Object Oriented Programming (OOP)

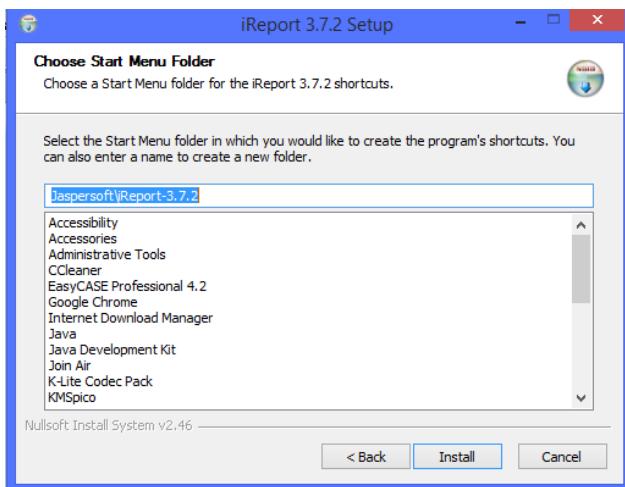
Guru : BEBEN SUTARA, S.Kom., M.T



Ceklis iReport lalu klik Next



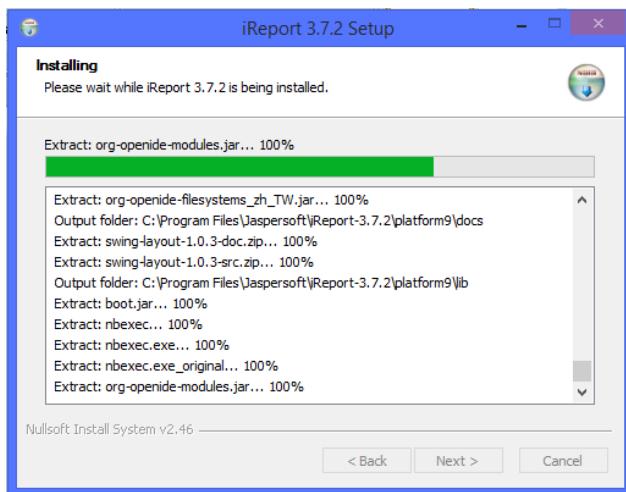
Tentukan lokasi installasi lalu klik tombol Next



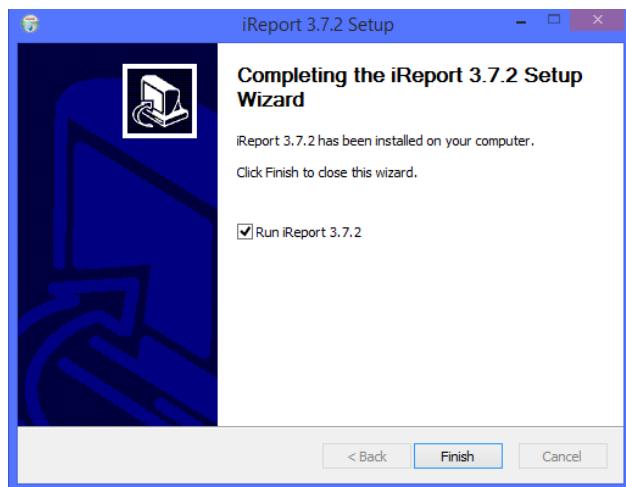
Klik tombol Install

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



Tunggu sampai proses selesai



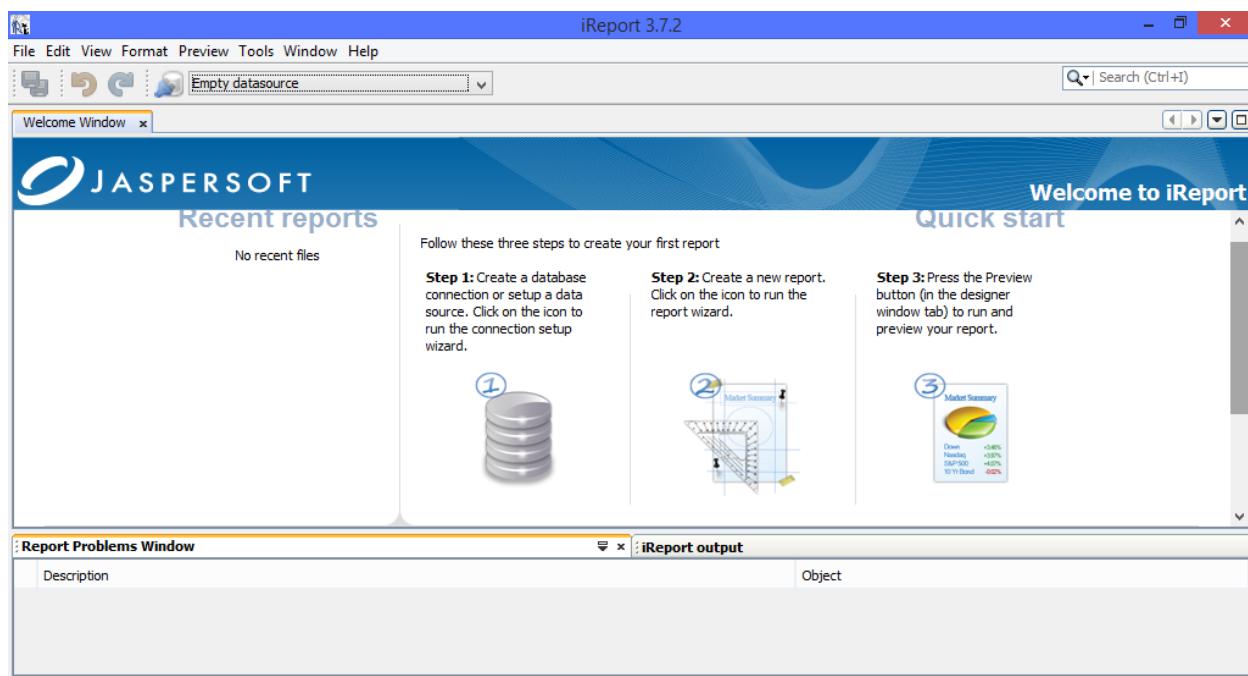
Klik tombol **Finish**



Jendela pembuka

Object Oriented Programming (OOP)

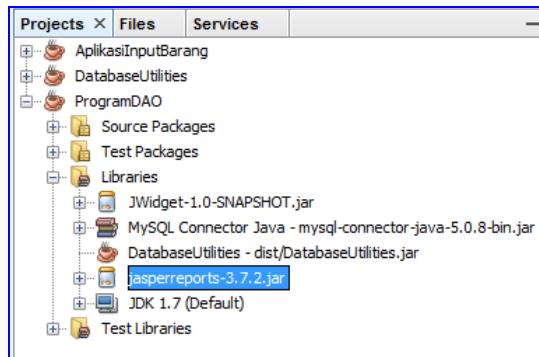
Guru : BEBEN SUTARA, S.Kom., M.T



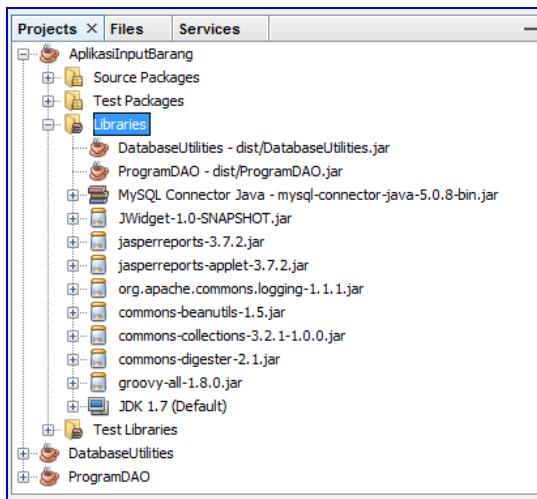
Area kerja iReport

2. Menambahkan Library JasperReport

Setelah **iReport-5.5.0-plugin** telah terinstall, lalu tambahakan Library **JasperReport** pada project **ProgramDAO**.



Lalu tambahkan beberapa Library yang sudah disediakan pada project **AplikasiInputBarang**.



3. Membuat Fasilitas viewReport

Untuk membuat fasilitas tampil report, akan dibuat pada project **ProgramDAO** sehingga akan terkumpul fasilitas seperti **insert**, **update**, **delete**, **selectAll**, dan **viewReport**. Caranya buka class **fasilitas** pada project **ProgramDAO**, buatlah satu method untuk menampilkan report yang diberi nama **viewReport**. Perhatikan kode program penambahan yang diberi tanda huruf **bold**.

```
package com.atep;
import com.mysql.jdbc.PreparedStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.view.JasperViewer;
public class fasilitas {
    private Connection koneksi;
    private String ReportPath="report/";
    public fasilitas() {
        koneksi = KoneksiDatabase.getKoneksi();
    }
    public void insert(barang barang){
        PreparedStatement prepare = null;
        try {
```

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
String           sql          ="INSERT      INTO      barang
(kode_barang,nama_barang,jumlah,harga)VALUES(?, ?, ?, ?)";
    prepare = (PreparedStatement) koneksi.prepareStatement(sql);
    prepare.setString(1, barang.getKode_barang());
    prepare.setString(2, barang.getNama_barang());
    prepare.setInt(3, barang.getJumlah());
    prepare.setLong(4, barang.getHarga());
    prepare.executeUpdate();
    System.out.println("Prepare statement berhasil dibuat");
}catch(SQLException ex){
    System.out.println("Prepare statement gagal dibuat");
    System.out.println("Pesan : " + ex.getMessage());
}finally{
    if (prepare != null){
        try{
            prepare.close();
            System.out.println("Prepare statemen berhasil ditutup");
        }catch(SQLException ex){
            System.out.println("Prepare statemen gagal ditutup");
            System.out.println("Pesan : " + ex.getMessage());
        }
    }
}
}

public void update(barang barang){
PreparedStatement prepare = null;
try {
    String sql  ="UPDATE barang SET nama_barang=?,jumlah=?,harga=? WHERE
kode_barang=?";
    prepare = (PreparedStatement) koneksi.prepareStatement(sql);
    prepare.setString(1, barang.getNama_barang());
    prepare.setInt(2, barang.getJumlah());
    prepare.setLong(3, barang.getHarga());
    prepare.setString(4, barang.getKode_barang());
    prepare.executeUpdate();
}
```

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
        System.out.println("Prepare statement berhasil dibuat");
    }catch(SQLException ex){
        System.out.println("Prepare statement gagal dibuat");
        System.out.println("Pesan : " + ex.getMessage());
    }finally{
        if (prepare != null){
            try{
                prepare.close();
                System.out.println("Prepare statemen berhasil ditutup");
            }catch(SQLException ex){
                System.out.println("Prepare statemen gagal ditutup");
                System.out.println("Pesan : " + ex.getMessage());
            }
        }
    }
}

public void delete(String kode_barang){
    PreparedStatement prepare = null;
    try {
        String sql ="DELETE FROM barang WHERE kode_barang=?";
        prepare = (PreparedStatement) koneksi.prepareStatement(sql);
        prepare.setString(1, kode_barang);
        prepare.executeUpdate();
        System.out.println("Prepare statement berhasil dibuat");
    }catch(SQLException ex){
        System.out.println("Prepare statement gagal dibuat");
        System.out.println("Pesan : " + ex.getMessage());
    }finally{
        if (prepare != null){
            try{
                prepare.close();
                System.out.println("Prepare statemen berhasil ditutup");
            }catch(SQLException ex){
                System.out.println("Prepare statemen gagal ditutup");
                System.out.println("Pesan : " + ex.getMessage());
            }
        }
    }
}
```

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

```
        }
    }
}

public List<barang> selectAll(){
    PreparedStatement prepare = null;
    ResultSet result = null;
    List<barang> list = new ArrayList<>();
    try {
        String sql ="SELECT * FROM barang ORDER BY kode_barang";
        prepare = (PreparedStatement) koneksi.prepareStatement(sql);
        result = prepare.executeQuery();
        while (result.next()){
            barang barang = new barang();
            barang.setKode_barang(result.getString("kode_barang"));
            barang.setNama_barang(result.getString("nama_barang"));
            barang.setJumlah(result.getInt("jumlah"));
            barang.setHarga(result.getLong("harga"));
            list.add(barang);
        }
        System.out.println("Prepare statement berhasil dibuat");
        return list;
    }catch(SQLException ex){
        System.out.println("Prepare statement gagal dibuat");
        System.out.println("Pesan : " + ex.getMessage());
        return list;
    }finally{
        if (prepare != null){
            try{
                prepare.close();
                System.out.println("Prepare statemen berhasil ditutup");
            }catch(SQLException ex){
                System.out.println("Prepare statemen gagal ditutup");
                System.out.println("Pesan : " + ex.getMessage());
            }
        }
    }
}
```

```
        if (result != null){
            try{
                result.close();
                System.out.println("Resultset berhasil ditutup");
            }catch(SQLException ex){
                System.out.println("Resultset gagal ditutup");
                System.out.println("Pesanan : " + ex.getMessage());
            }
        }
    }

    public void viewReport(String nm_report){
        String reportSource;
        String reportDest;
        reportSource=ReportPath +"laporan/" + nm_report + ".jrxml";
        reportDest=ReportPath +"hasil/" + nm_report + ".html";
        try {
            JasperReport jasperReport = JasperCompileManager.compileReport(reportSource);
            JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, null,
koneksi);
            JasperExportManager.exportReportToHtmlFile(jasperPrint, reportDest);
            JasperViewer.viewReport(jasperPrint, false);
        } catch (JRException e) {
            e.printStackTrace();
        }
    }
}
```

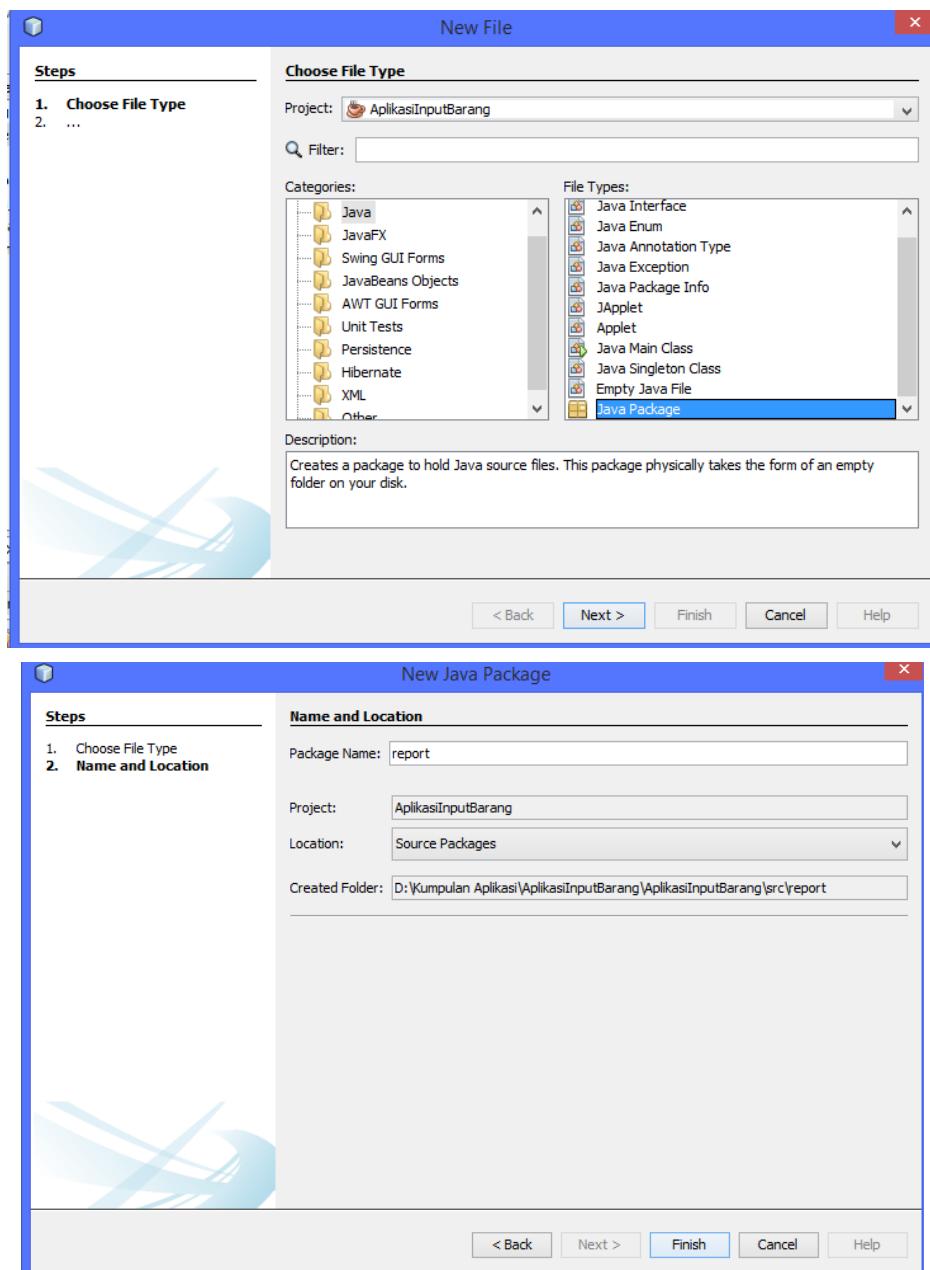
4. Membuat Package dan Report

Package dibutuhkan untuk menyimpan report, untuk membuat package dan report ikuti langkah dibawah ini.

- 1) Pada project **AplikasiInputBarang**, buatlah file baru.

Object Oriented Programming (OOP)

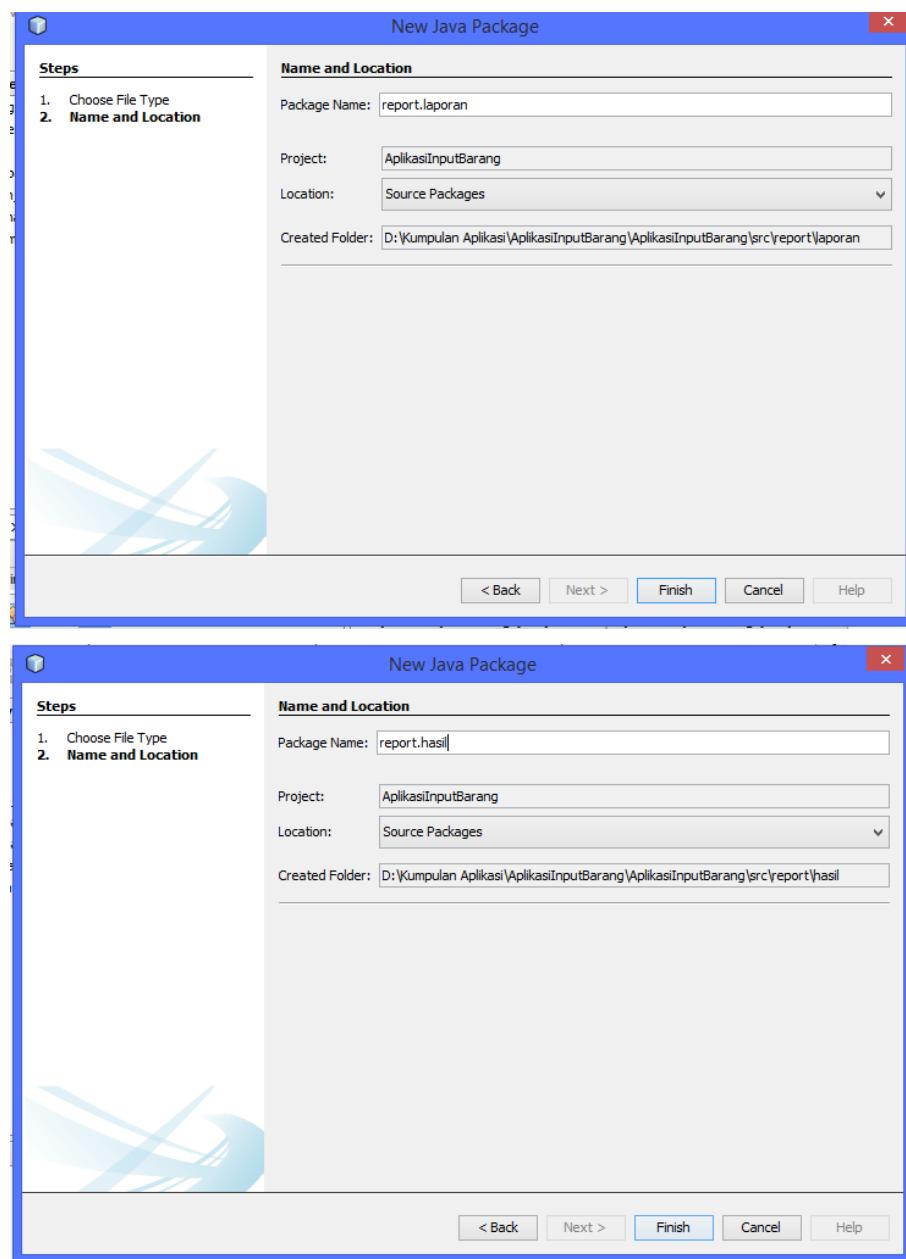
Guru : BEBEN SUTARA, S.Kom., M.T



- 2) Kemudian pada package report, tambahkan dua buah package lagi yaitu **report.laporan** dan **report.hasil** seperti tampilan dibawah ini.

Object Oriented Programming (OOP)

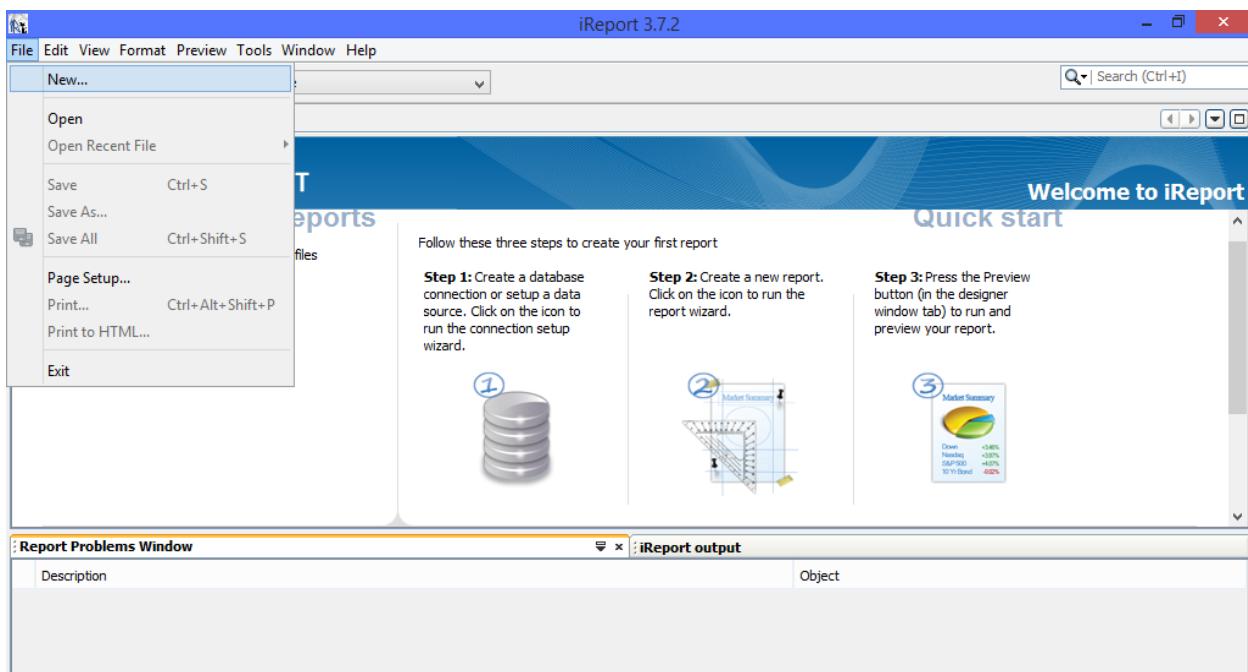
Guru : BEBEN SUTARA, S.Kom., M.T



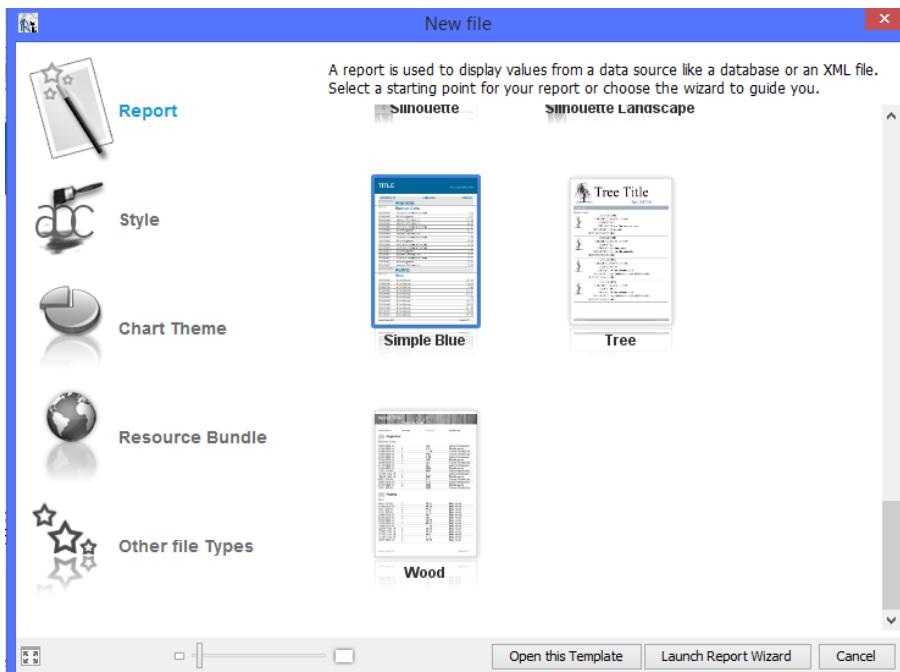
Selanjutnya tinggal buat reportnya, caranya sebagai berikut :

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



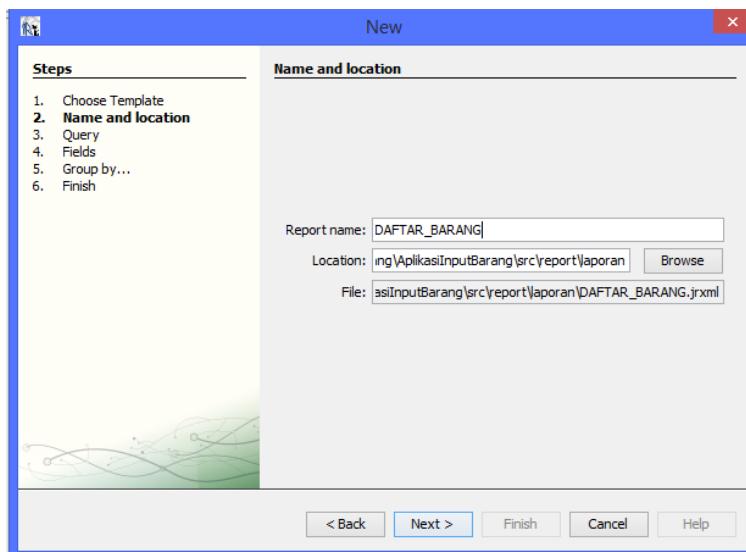
Pilih **File** lalu pilih **New**



Misalnya pilih **Style** Reportnya yaitu **Simple Blue**

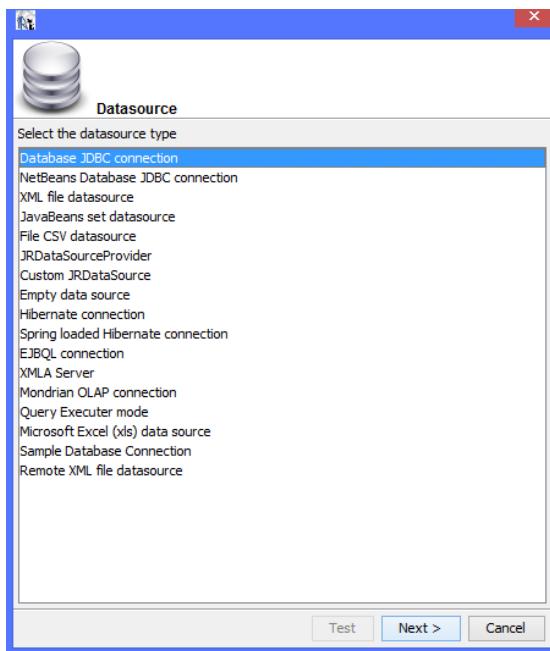
Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



Isilah report nama : **DAFTAR_BARANG**, tentukan location : **D:/..../src/report/laporan** lalu klik tombol

Next



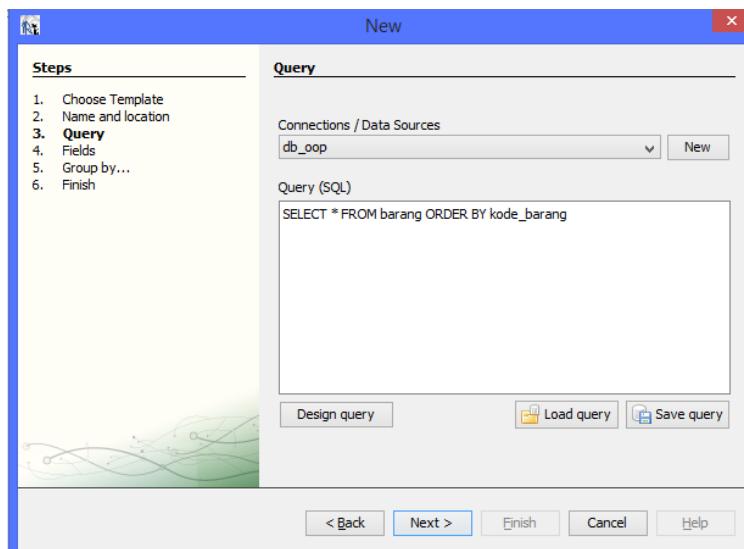
Buat file baru (**New**) pilih **Database JDBC connection** lalu klik tombol **Next**

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



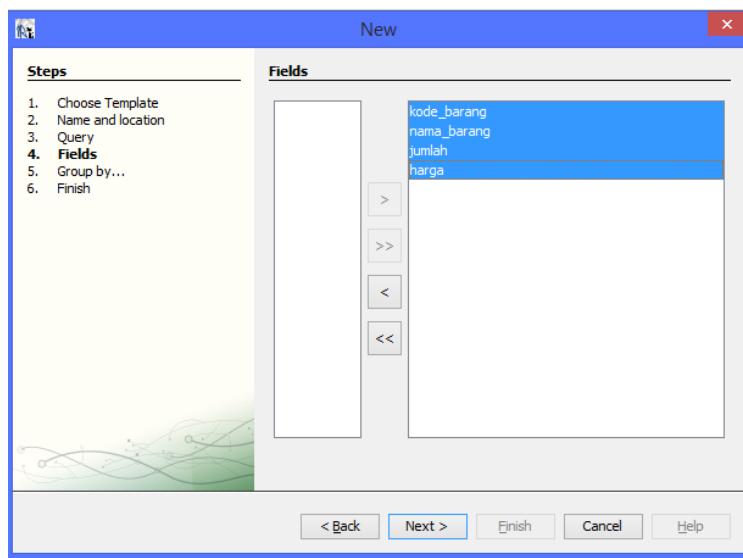
Masukkan Name : **db_oop**, JDBC Driver : **MySQL (com.mysql.jdbc.Driver)**, JDBC URL :
jdbc:mysql://localhost/db_oop, Username : **root**, Password : <dikosongkan>



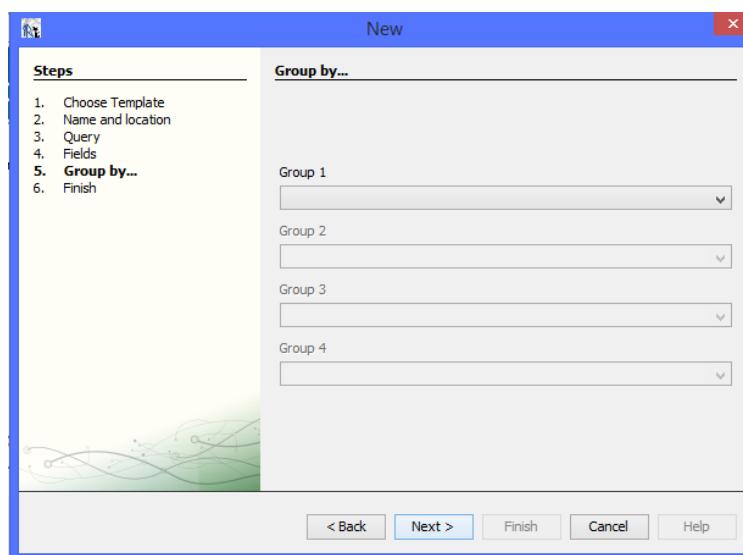
Klik tombol **Next**

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



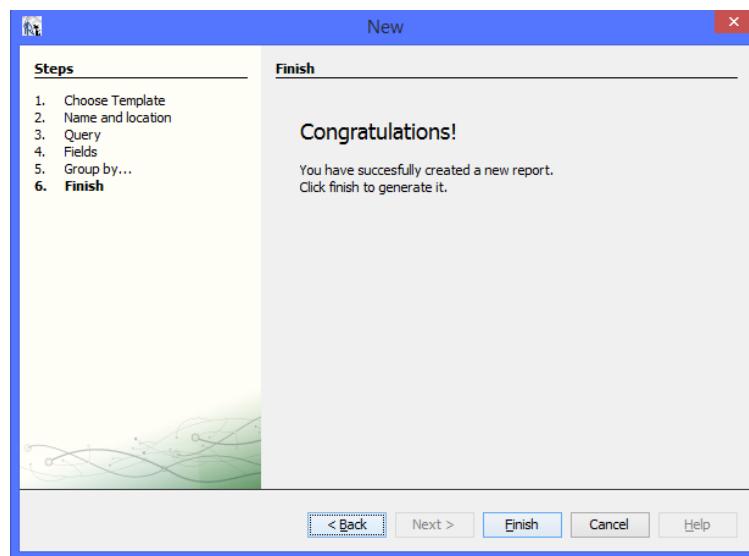
Pindahkan semua **fields** dengan mengklik tombol **>>** lalu klik tombol **Next**



Pada tab **Group by** silahkan pilih Group-nya, namun biasanya group digunakan ketika ada field yang ingin dikelompokan. Lalu klik tombol **Next**

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



Klik tombol **Finish**

The screenshot shows the iReport 3.7.2 software interface. The menu bar includes File, Edit, View, Format, Preview, Tools, Window, Help. The title bar says "iReport 3.7.2". The main area shows a report titled "DAFTAR_BARANG.jrxml". The report has a "TITLE" section at the top. Below it is a table with columns: "kode_barang", "nama_barang", "jumlah", and "harga". The table contains some sample data. The "Page Header" section also contains some code. The "Report Inspector" panel on the left lists various report components like Styles, Fields, Variables, etc. The "Palette" panel on the right shows "Report Elements" such as Chart, Ellipse, Image, Barcode, List, and Line. The "Report Problems Window" and "iReport output" panels are also visible at the bottom.

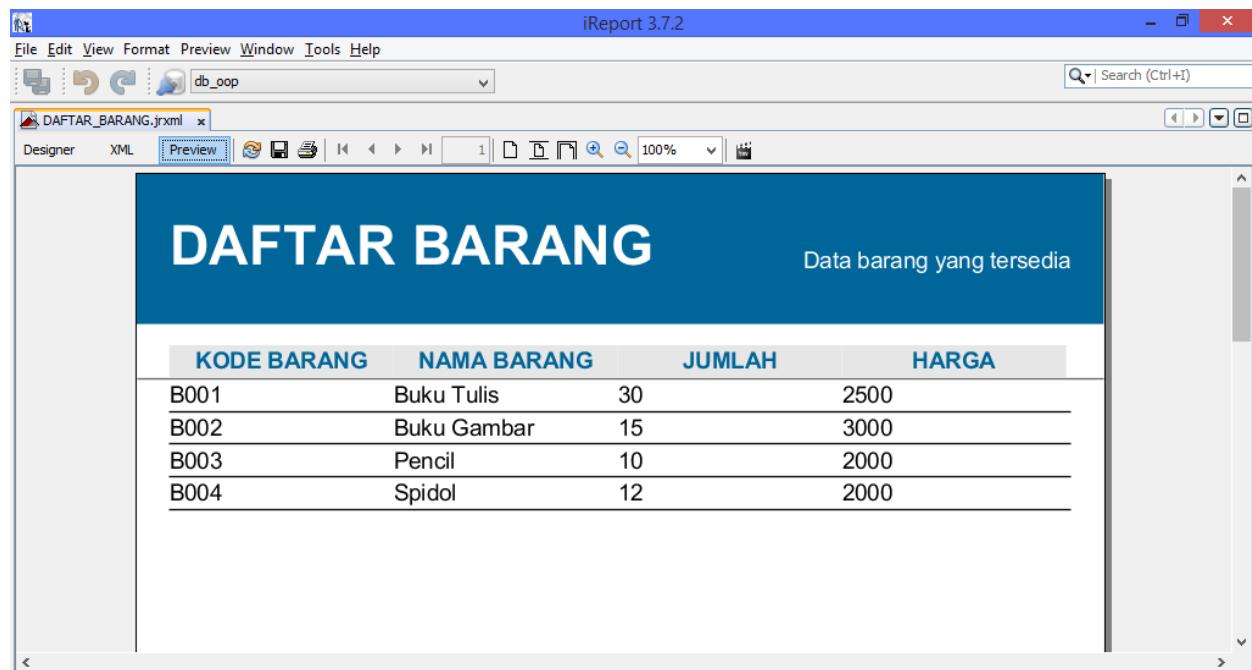
Hasil tampilan report

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



Ubahlah **Title** dan **Field** disesuaikan dengan kebutuhan



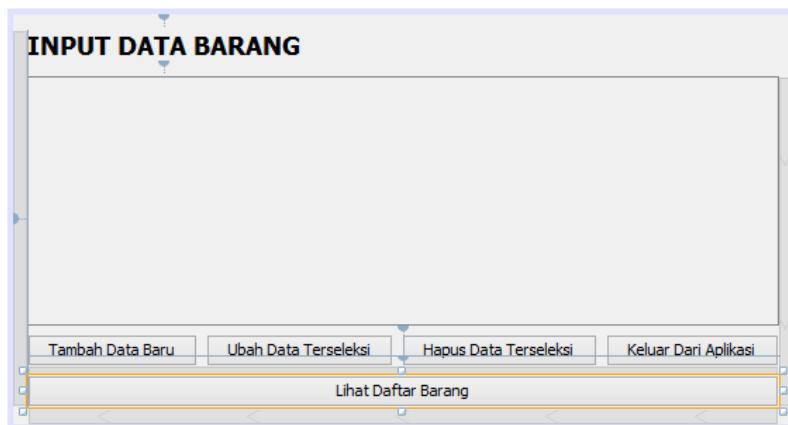
Klik tombol **Preview**

5. Menambahkan Tombol Lihat Daftar Barang

Untuk membuat tombol lihat report, kita akan menggunakan class **F_Utama.java**. dengan menambahkan satu tombol yaitu **Lihat Daftar Barang** dengan nama variable nya yaitu **btn_lihat**, seperti tampilan dibawah ini.

Object Oriented Programming (OOP)

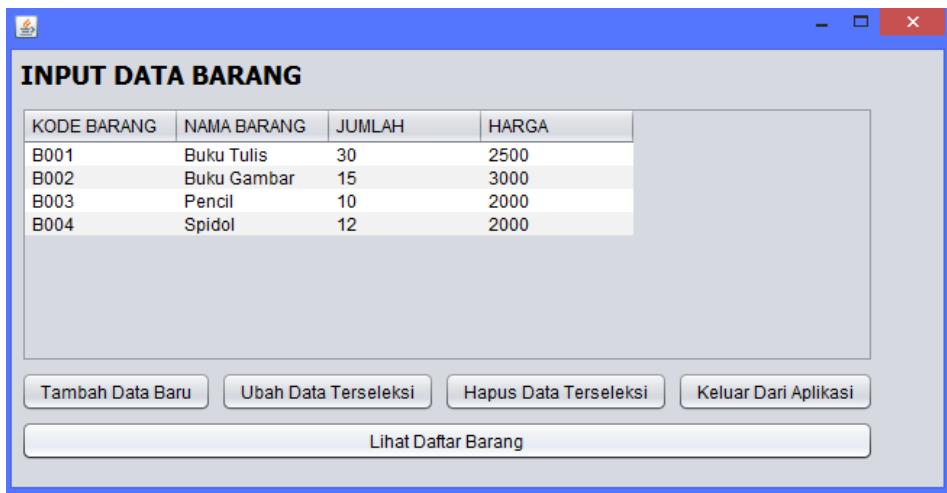
Guru : BEBEN SUTARA, S.Kom., M.T



Untuk kode programnya silahkan double klik tombol **Lihat Daftar Barang** dan ketikkan kode program dibawah ini.

```
private void btn_lihatActionPerformed(java.awt.event.ActionEvent evt) {  
    perintah.viewReport("daftar_barang");  
}
```

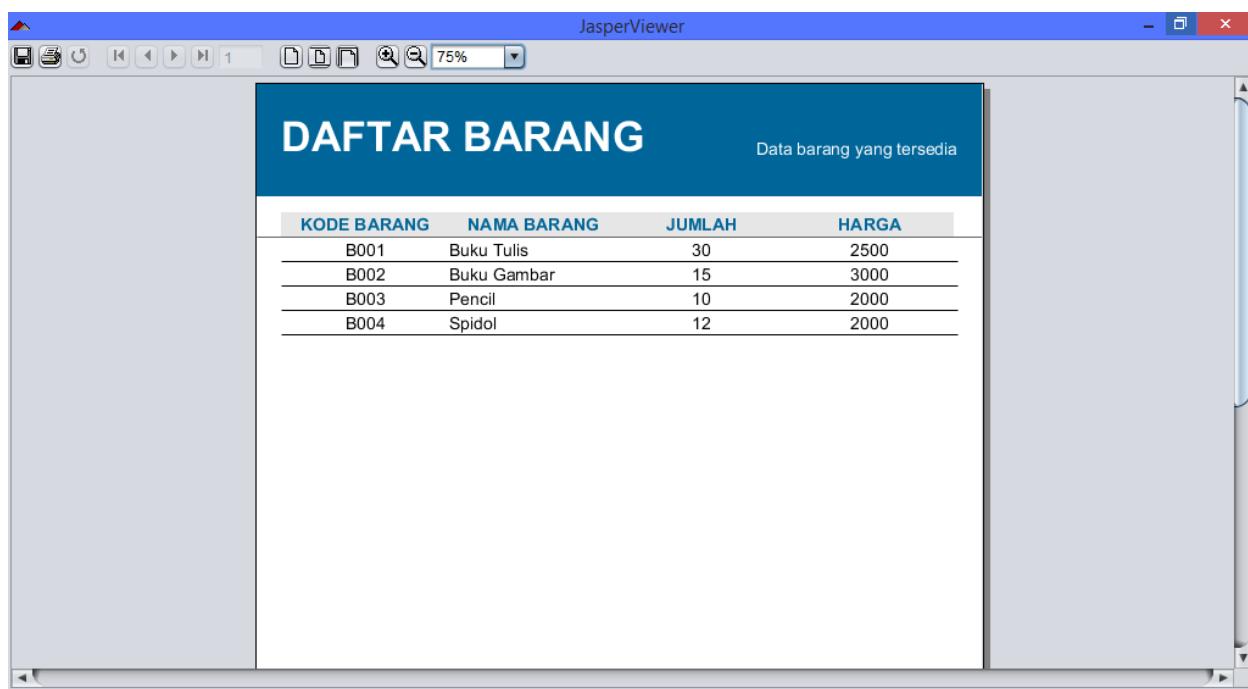
Silahkan jalankan programnya. (**Shift+F6**)



Klik tombol **Lihat Daftar Barang**

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



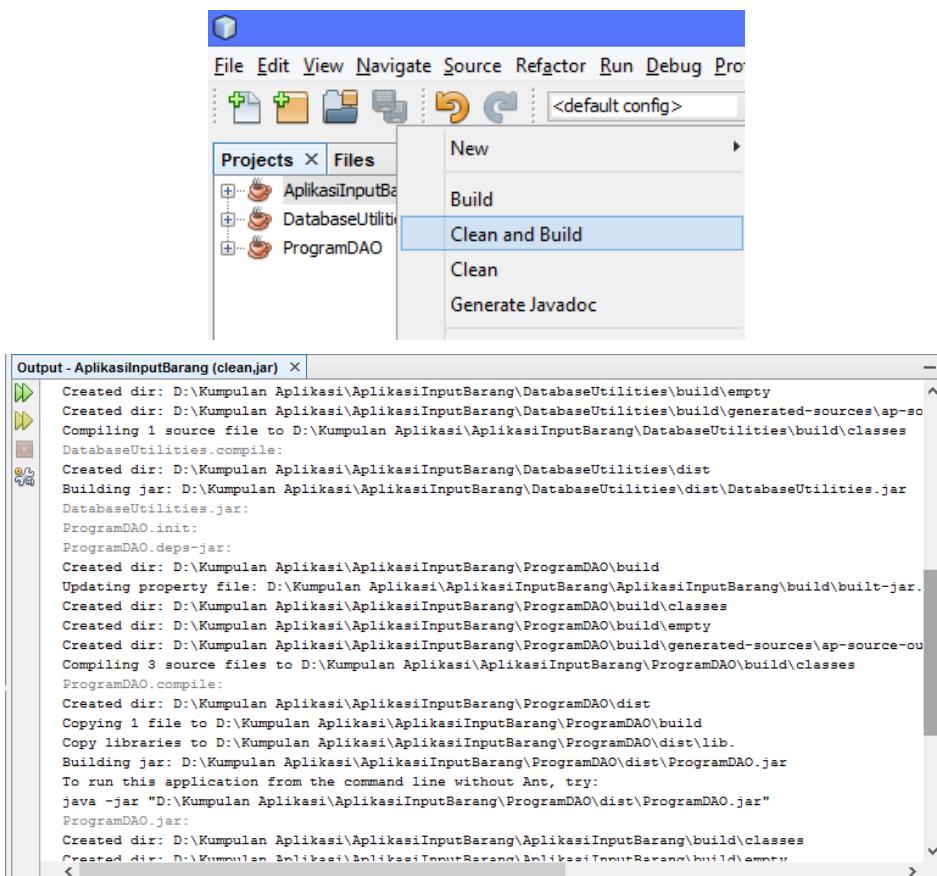
MODUL 16

PENDISTRIBUSIAN APLIKASI

Pada praktikum sebelumnya kita sudah membuat beberapa project yang digunakan untuk membuat aplikasi yaitu Aplikasi Input Barang. Selanjutkan kita coba lakukan pendistribusian sehingga aplikasi yang dibuat dapat berjalan tanpa program mentahnya.

Untuk melakukan pendistribusian, ikuti langkah berikut :

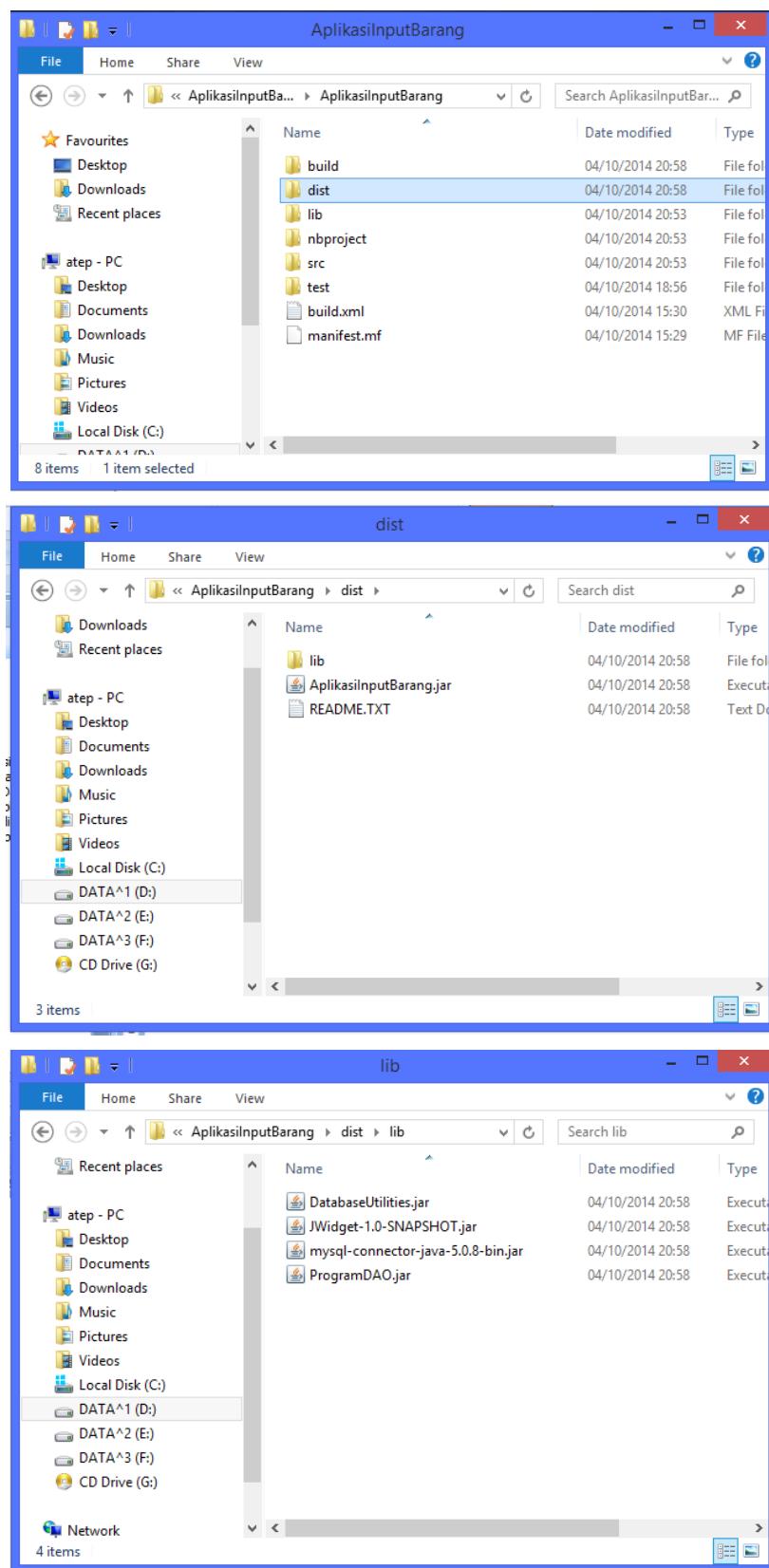
- 1) Klik kanan pada project **AplikasiInputBarang**, pilih **Clean and Build**.



- 2) Kemudian lihat pada folder dimana tempat penyimpanan project-project tersebut. Maka akan ditemukan folder **dist**, yang mana didalamnya terdapat file **AplikasiInputBarang.jar** dan Folder **lib** yang menyimpan seluruh library yang dibutuhkan untuk menjalankan aplikasi.

Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T



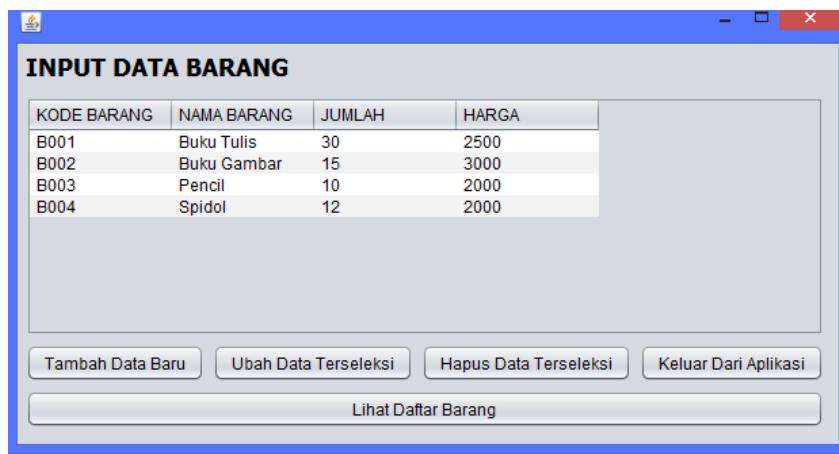
Object Oriented Programming (OOP)

Guru : BEBEN SUTARA, S.Kom., M.T

- 3) Kemudian copy-kan folder **report** yang berada di alamat src/report ke folder **dist**.

Name	Date modified	Type	Size
lib	06/10/2014 21:40	File folder	
report	06/10/2014 21:50	File folder	
AplikasiInputBarang.jar	06/10/2014 21:40	Executable Jar File	87 KB
README.TXT	06/10/2014 21:40	Text Document	2 KB

- 4) Untuk mencoba menjalankan aplikasi silahkan double klik file **AplikasiInputBarang.jar** maka akan muncul tampilan seperti dibawah ini sehingga aplikasi sudah dapat digunakan.



DAFTAR PUSTAKA

Herbert Schildt (2005). *A Beginner's Guide, 3rd Edition*

Huda, M., dan Bunafit (2005). *Trik Rahasia Pemrograman Database dengan Java*. PT Elex Media Komputindo – Jakarta.