Muhammad Fachrie
Enny Itje Sela

TE-TK

2019

MODUL PRAKTIKUM

Pemrograman Berorientasi Objek

Nama : _____

NPM : _____

Fakultas Teknologi Informasi dan Elektro

Universitas Teknologi Yogyakarta

PEMROGRAMAN BERORIENTASI OBJEK

Oleh:

Muhammad Fachrie

Enny Itje Sela

UNIVERSITAS TEKNOLOGI YOGYAKARTA

2019

Diterbitkan oleh:
Universitas Teknologi Yogyakarta
Jl. Siliwangi, Jombor, Sleman, Yogyakarta
Email: publikasi@uty.ac.id
Website: uty.ac.id

Pemrograman Berorientasi Objek

ISBN
978-623-92626-5-5

Oleh: Muhammad Fachrie
Enny Itje Sela

-

Edisi ke-1
Cetakan pertama, 2019

KATA PENGANTAR

Puji syukur ke hadirat Tuhan YME atas segala limpahan karunia dan hidayah-Nya, maka tim penulis akhirnya dapat menyelesaikan Modul Praktikum Pemrograman Berorientasi Objek (PBO) ini. Modul ini disusun seefektif mungkin dan sistematis agar mampu memudahkan para mahasiswa dalam mempelajari dan memahami Mata Kuliah PBO. Setiap pembahasan disertai dengan contoh kode program yang akan membantu mahasiswa mandiri untuk mencari tahu sintaks-sintaks yang sengaja tidak dijelaskan secara gamblang di dalam modul.

Pada bagian awal, mahasiswa dikenalkan tentang lingkungan pemrograman Java sebagai bahasa pemrograman PBO. Mahasiswa dianjurkan menulis kode program melalui *text editor* biasa dan melakukan kompilasi serta mengeksekusi program melalui *console* (*command-prompt*). Selanjutnya, konsep PBO tentang kelas dan objek termasuk karateristik PBO yang meliputi enkapsulasi (*encapsulation),* pewarisan (*inheritance),* polimorfisme (*polymorphism),* dan abstraksi (*abstract).* Materi selanjutnya mengenai dasar pemrograman berbasis GUI (*Graphic User Interface*) menggunakan IDE Netbeans. Keempat bab terakhir membahas penggunaan *library* yang penting untuk menunjang pengembangan aplikasi yang lebih baik, di antaranya materi tentang koneksi aplikasi Java dengan database MySQL, pembacaan dan penulisan data dari dan ke dalam file Excel, pengaksesan comm port pada *hardware* komputer, serta ditutup dengan latihan mengembangkan proyek aplikasi berbasis Java.

Akhirnya, semoga modul ini dapat memberikan manfaat yang sebanyak-banyaknya bagi para pembacanya. Kami pun menyadari bahwa modul ini pasti tidak luput dari kekurangan di berbagai sisi, oleh karena itu, kami dengan senang hati menerima saran dari para pembaca melalui surel kami di: muhammad.fachrie@staff.uty.ac.id atau ke ennysela@staff.uty.ac.id.

Yogyakarta, November 2019

Tim Penulis

Daftar Isi

Halaman Judul i

Kata Pengantar iii

Daftar Isi iv

Pendahuluan v

BAB 1 Mengenal Environment Pemrograman Java 1
BAB 2 Percabangan dan Perulangan 10
BAB 3 Kelas (*Class*), Objek (*Object)*, dan Class Diagram 15
BAB 4 Enkapsulasi 28
BAB 5 Constructor 32
BAB 6 Pewarisan (*Inheritance)* 47
BAB 7 Polimorfisme (*Polymorphism)* 52
BAB 8 Kelas Abstrak (*Abstract Class)* 62
BAB 9 Interface 71
BAB 10 Pemrograman Berbasis GUI (1) 79
BAB 11 Pengaksesan Basis Data 88
BAB 12 Membaca File dari Ms. Excel 94
BAB 13 Java Class Library 99
BAB 14 Mengakses Port pada Hardware Komputer 110

Daftar Pustaka 114

PENDAHULUAN

1. Deskripsi Materi

Pemrograman berorientasi objek (PBO) Praktikum ditujukan untuk mengimplementasikan konsep perancangan berorientasi objek ke dalam bentuk pemrogramaman. Bahasa pemrograman yang digunakan pada praktikum ini adalah Java, editor teks yang digunakan adalah notepad, dan StarUML digunakan untuk menggambarkan skema. Modul ini diawali dengan pembahasan mengenai pengenalan Bahasa Java, mulai dari proses instalasi, pengaturan classpath, struktur program, hingga mekanisme bagaimana sebuah kode program dikompilasi dan dieksekusi dalam Java, tipe data, variable, dan operator; skema percabangan; skema perulangan.

Selanjutnya pembahasan mengenai diawali dengan pengenalan kelas, objek, dan class diagram. Kemudian, materi tentang karakteristik PBO: enkapsulasi, pewarisan, polimorfisme, dan abstraksi. Sampai pada tahap ini, proses penulisan program dilakukan menggunakan *text editor* biasa dan dikompilasi menggunakan *console*. Pada tahap berikutnya, pembelajaran membahas mengenai dasar pembuatan aplikasi Java berbasis GUI dengan menggunakan IDE Netbeans. Setelah itu, pembelajaran dilanjutkan dengan penggunaan *library* tambahan untuk mendukung fungsionalitas aplikasi yang dibuat, misalnya *library* JDBC untuk terhubung dengan database MySQL, *library* jExcelAPI yang memungkinkan aplikasi membaca dan menulis data dari/ ke dalam file Excel, serta dasar mengenai proses koneksi comm port pada *hardware* komputer.

2. Prasyarat

- Telah lulus mata kuliah: Algoritma Pemrograman
- Mampu menggunakan *command-prompt* pada sistem operasi berbasis Windows atau Linux.

3. Petunjuk Pemakaian Modul

Modul ini dapat digunakan mahasiswa dengan pertimbangan sebagai berikut :

- Mahasiswa telah memiliki modul dan telah membaca modul sebelum mata praktikum dimulai.
- Mahasiswa menjawab soal latihan yang diberikan mengikuti langkah-langkah yang ada
- Pemberian pengayaan materi bagi mahasiswa yang telah memahami dan menyelesaikan soal latihan
- Memberikan tugas atau tinjauan ulang terhadap materi sekaligus mengidentifikasi kesulitan-kesulitan mahasiswa dalam memahami materi

4. Standar Kompetensi

- Mahasiswa mampu menulis kode program dengan baik dalam Bahasa Java.
- Mahasiswa mampu memahami konsep Pemrograman Berorientasi Objek dan dan dapat membedakannya dengan paradigm pemrograman procedural.
- Mahasiswa mampu mengembangkan aplikasi berbasis GUI.
- Mahasiswa mampu menggunakan *library* yang dapat menunjang fungsionalitas aplikasi yang dibuat.

BAB 1
Mengenal Environment Pemrograman Java

1.1 Capaian Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan:

1. Memahami perintah-perintah dasar: input, output serta tipe data
2. Mengetahui cara mengkompilasi dan mengeksekusi kode program di dalam Java

1.2 Indikator

1. Mampu menulis kode program sederhana
2. Mampu melakukan kompilasi dan eksekusi program melalui *console*

1.3 Uraian Materi

Java dikembangkan oleh suatu tim yang dipimpin oleh Patrick Naughton dan James Gosling dalam suatu proyek dari Sun Microsystem yang memiliki kode Green dengan tujuan untuk meghasilkan bahasa komputer sederhana yang dapat dijalankan di peralatan sederhana dengan tidak terikat pada arsitektur tertentu. Mula-mula James Gosling menyebut bahasa pemrograman yang dihasilkan dengan OAK, tetapi OAK sendiri merupakan nama dari bahasa pemrograman komputer yang sudah ada maka kemudian Sun mengubahnya menjadi Java .

Akhirnya setelah melalui beberapa transformasi dan proses, Sun meluncurkan browser Java yang disebut Hot Java yang mampu menjalankan applet. Versi-versi Java banyak dikembangkan hingga muncul JDK yang mempunyai teknologi Swing untuk menampilkan secara GUI. Teknologi Java yang disebut J2ME (Java 2 Micro Edition), sudah diadopsi oleh Nokia, Siemens, SonyEricson, Motorola, Samsung untuk menghasilkan apikasi mobile baik games maupun software bisnis dan berbagai jenis software lain yang dapat dijalankan di peralatan mobile seperti ponsel.

A. Mengenal Java SDK

SDK adalah kependekan dari Standard Development Kit , yang merupakan bekal utama bagi developer untuk membuat program dan menjalankan program Java. Secara garis besar, Java SDK terdiri dari :

- Java Compiler (Javac)
- Java Virtual Machine (sering disebut juga java Runtime Environment/ JRE)
- Java Class Libraries (koleksi class yang dapat digunakan untuk menghasilkan program Java)

- Java AplletViewer (program untuk menjalankan applet tanpa menggunakan browser)
- Java Debugger dan tool lain

Mula-mula kode program ditulis dengan Bahasa Java (*file* berekstensi **.java**) akan dikompilasi oleh *compiler* menjadi suatu kode objek yang berupa *file* berekstensi **.class** yang disebut dengan istilah ***bytecode***. Jadi, dalam Java, hasil akhir programnya tidak berupa *file* berekstensi **.EXE**, melainkan berkestensi **.class**. Selanjutnya *file bytecode* ini akan dieksekusi baris per baris oleh *interpreter*. Dengan demikian, proses kompilasi hanya dilakukan sekali, akan tetapi proses interpretasi akan dilakukan setiap kali program dieksekusi. Dengan adanya konsep *bytecode* ini, dalam terminologi Java dikenal adanya istilah "*write once, run everywhere*". Ini berarti bahwa sekali kita menulis program dalam Bahasa Java dan melakukan kompilasi terhadapnya, maka *bytecode*-nya dapat dijalankan di dalam *platform* sistem operasi manapun selama komputer tersebut terpasang *Java Virtual Machine* (JVM).

B. Instalasi Java

Sebelum membuat sebuah program menggunakan Bahasa Java, maka kita perlu melakukan instalasi *software* Java Plat form, Standar Edition (Java SE). *Software* ini dapat diunduh melalui *link* http://www.oracle.com/technetwork/java/javase/downloads/index.html untuk memperoleh Java SE versi terbaru (saat modul ini dibuat, versi terbaru adalah Java SE 9.0.1). Setelah Java SE berhasil diinstal, maka berikutnya kita membutuhkan *software text editor* untuk menulis kode program dalam Bahasa Java. Saat ini banyak *text editor* yang dapat digunakan, salah satu di antaranya adalah Notepad++. Untuk mempermudah dalam penulisan kode program, maka kita dapat menggunakan *software* Java IDE (*Integrated Development Environment*), misalnya Netbeans, Eclipse, Jcreator, Intellij Idea, atau yang lainnya.

C. Pengaturan CLASSPATH

Jika kita membuat kode program Java tanpa melalui IDE, maka proses kompilasi dan interpretasi program akan dilakukan melalui *console* (dalam *command prompt*). Agar hal ini dapat dilakukan, maka perlu dilakukan pengaturan CLASSPATH pada *console* sebelum mengkompilasi dan mengeksekusi program. Pengaturan CLASSPATH dilakukan untuk memberitahu komputer dimana *software* Java SE disimpan. Pada umumnya, komputer dengan sistem operasi berbasis Unix tidak memerlukan lagi pengaturan CLASSPATH.

Langkah pertama untuk melakukan pengaturan CLASSPATH (pada komputer berbasis Windows) adalah masuk ke aplikasi *command prompt* (cmd), kemudian ketikkan perintah di bawah ini.

```
C:\ SET PATH="C:\Program Files\Java\jdk1.8.0_141\bin"
```

Untuk menguji apakah pengaturan CLASSPATH tersebut sudah benar atau tidak, maka ketikkan perintah **javac** pada cmd. Jika keluar peringatan *'javac' is not recognized as an*

internal or external command, operable program or batch file, berarti pengaturan CLASSPATH gagal, jika tidak, maka berarti pengaturan telah berhasil.

D. Tipe Data

Di dalam Bahasa Java, tipe data dasar dapat dibedakan menjadi empat kelompok, yakni tipe data *integer* (bilangan bulat), tipe data *floating-point* (bilangan riil), tipe data karakter, dan tipe data *boolean*. Perlu diketahui bahwa pada Java, *String* bukan merupakan sebuah tipe data, melainkan sebuah objek, akan tetapi pada penggunaannya mirip seperti menggunakan sebuah tipe data.

1. Tipe Data *Integer*

Tipe data *integer* (bilangan bulat) pada Java terbagi menjadi empat jenis tipe, yaitu `byte`, `short`, `int`, dan `long`. Semua tipe data ini bersifat *signed* (bertanda), yaitu tipe data yang dapat merepresentasikan nilai negatif maupun positif. Tidak seperti pada kebanyakan bahasa pemrograman lainnya, Java tidak mendukung nilai *unsigned* (tipe tanpa tanda, yakni tipe data yang hanya dapat menyimpan nilai positif).

Tipe Data	Ukuran (*bit*)	Rentang
`Byte`	8	-128 s.d. 127
`Short`	16	-32.768 s.d. 32.767
`Int`	32	-2.147.483.648 s.d. 2.147.483.647
`Long`	64	-9.223.372.036.854.775.808 s.d. 9.223.372.036.854.775.807

2. Tipe Data Floating-Point

Tipe *floating-point* (bilangan riil) digunakan untuk merepresentasikan nilai-nilai yang mengandung pecahan atau angka desimal di belakang koma. Tipe data ini dibedakan menjadi dua macam, yakni tipe data `float` dan `double`.

Tipe Data	Ukuran (*bit*)	Rentang
`Float`	32	3.4e-038 s.d. 3.4e+038
`double`	64	1.7e-308 s.d. 1.7e+308

3. Tipe Data Karakter

Pada Java, tipe data `char` merupakan karakter *Unicode* yang dapat merepresentasikan semua karakter yang ada, yakni semua karakter yang terdapat dalam semua bahasa (dalam lingkup internasional), seperti Bahasa Latin, Arab, Yunani, dsb.

4. Tipe Data *Boolean*

Seperti bahasa pemrograman pada umumnya, tipe data `boolean` pada Java digunakan untuk menampung nilai logika, yakni *true* dan *false*. Perlu diingat, pada Java, nilai *true* tidak sama dengan nilai 1, begitupun dengan nilai *false* tidak sama dengan nilai 0. Tipe *boolean* adalah tipe data yang dikembalikan oleh semua operator relasional, seperti '==', '>,' '<', '>=', dan '<='.

E. Variabel

Variabel terdiri atas variabel class, instance, dan lokal. Variabel class adalah variabel yang dapat diakses oleh semua method di dalam class, termasuk *static method* seperti *main*. Variabel instance sama halnya dengan variabel class, tetapi tidak menggunakan kata static dalam pendeklarasiannya. Variabel instance terhubung dengan instance dari class. Jadi kita hanya bisa menggunakannya ketika membuat instance dari class. Karena static method tidak terhubung dengan instance dari class, kita tidak bisa menggunakan variabel instance dalam static method dan memasukkannya dalam main method. Variabel lokal adalah variabel yang dideklarasikan di dalam badan method. Jadi kita hanya dapat menggunakan variabel tersebut hanya di dalam method. Method lain dalam class tidak peduli akan keberadaan variabel tersebut. Dan variabel lokal hanya akan ada jika method (yang memiliki variabel lokal tersebut) dieksekusi. Dalam pendeklrasiannya kita tidak perlu menambahkan kata static (seperti yang harus dilakukan pada saat mendeklarasikan variabel class). Jika kata static digunakan dalam pendeklarasian variabel lokal, maka compiler akan menghasilkan pesan error dan menolak untuk melakukan compile terhadap program.

Sebuah variabel dideklarasikan dengan terlebih dahulu menuliskan tipe data yang ingin digunakan, kemudian baru menuliskan nama variabelnya, seperti pada contoh di bawah ini.

```
int angka1;
int angka1, angka2;
double x,y,z;
char simbol;
String kalimat;
```

Pada saat deklarasi variabel, kita juga dapat langsung menginisialisasi nilai dari variabel tersebut, seperti contoh di bawah ini.

```
int angka1 = 100;
int angka1 = 100, angka2 = -5;
double x = 1/12;
char simbol = '+';
String kalimat = "Modul Praktikum PBO - UTY";
```

Contoh variabel statis:

```
public class SayHello
{
// deklarasi class variable
static String helloMessage = "Hello Programmer";
public static void main(String args[])
{
System.out.println(helloMessage);
System.out.println(myNickNameIs);
}
// deklarasi class variable
static String myNickNameIs = "ghaNOZ2480";
}
```

F. Operator

Java merupakan bahasa pemrograman yang kaya akan operator. Secara umum, operator di dalam Java dibagi menjadi lima bagian: operator aritmatika, relasional, logika, dan *bitwise*, dan *assignment*.

1. Operator Aritmatika

Operator ini digunakan pada operasi-operasi aritmatika seperti penjumlahan, pengurangan, pembagian dan lain-lain. Tabel berikut ini menunjukkan daftar tabel operator aritmatika di dalam Java.

Operator	Keterangan
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Modulus (sisa bagi)
++	Increment (menaikkan nilai dengan 1)
--	Decrement (menurunkan nilai dengan 1)

2. Operator Relasional

Operator tersebut digunakan untuk membandingkan dua buah nilai (variabel) atau lebih, dimana operator ini akan mengembalikan atau menghasilkan nilai *True* atau *False*. Berikut ini tabel yang berisi daftar operator relasional di dalam Java.

Operator	Keterangan
==	Sama dengan
!=	Tidak sama dengan
>	Lebih besar

<	Lebih Kecil
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan

3. Operator Logika

Operator ini menghasilkan nilai yang sama dengan operator relasional, hanya saja penggunaannya lebih pada operasi antara dua buah operand yang bertipe `boolean`. Berikut ini tabel yang berisi daftar operator logika di dalam Java.

Operator	Keterangan
&&	Operasi AND
||	Operasi OR
^	Operasi XOR
!	Operasi NOT (Negasi)

4. Operator *Bitwise*

Operator *bitwise* digunakan untuk memanipulasi nilai dari bitnya, sehingga diperoleh nilai yang lain. Berikut ini adalah daftar operator *bitwise* dalam Java.

Operator	Keterangan
&	Operasi Bitwise AND
|	Operasi Bitwise OR
^	Operasi Bitwise XOR
~	Operasi Bitwise NOT
>>	Operasi shift right (geser ke kanan sebanyak n bit)
>>>	Operasi shift right zero fill
<<	Operasi shift left (geser ke kiri sebanyak n bit)

5. Operator Assignment

Operator *assignment* digunakan untuk memberikan suatu nilai ke sebuah varibel. Operator tersebut ditandai dengan tanda sama dengan '='. Tabel di bawah ini menunjukkan daftar operator *assignment*.

Operator	Contoh	Ekuivalen dengan
+=	`b += a`	`b = b+a`
-=	`b -= a`	`b = b-a`
*=	`b *= a`	`b = b*a`
/=	`b /= a`	`b = b/a`
%=	`b %= a`	`b = b%a`

1.4. Latihan

A. Membuat Program Sederhana

1. Menulis program

   Dengan menggunakan Editor teks sederhana (misalnya notepad), tulislah program berikut (nomor baris tidak perlu dituliskan) :

```
1  public class Program1{
2        public static void main(String[] args){
3              System.out.println("Hello World");
4        }
5  }
```

2. Menyimpan program

   Simpanlah program diatas pada suatu folder yang sudah disediakan dengan ketentuan:

   - nama file SAMA PERSIS dengan nama class yang nama *class* yang memiliki *method* main() (ingat : Java bersifat case sensitive)
   - nama file berekstensi .java

   Simpan kode program tersebut dengan nama **Program1.java**, jangan lupa untuk mengubah bagian ”Save as type” menjadi **All Types** (jangan simpan sebagai .txt).

3. Kompilasi program

   Langkah berikutnya adalah mengkompilasi kode program Java yang telah dibuat dengan cara masuk ke dalam *command prompt* kemudian masuk ke direktori dimana kode program tersebut disimpan. Jangan lupa, sebelumnya Anda harus melakukan pengaturan CLASSPATH terlebih dahulu. Ketikkan kata kunci **javac** diikuti dengan nama *file* yang ingin dikompilasi pada *command prompt* seperti contoh di bawah ini.

```
D:\Praktikum\Java\ javac Program1.java
```

   Setelah berhasil melakukan kompilasi, maka secara otomatis pada direktori dimana Anda menyimpan kode program tersebut akan muncul sebuah *file* baru yakni **Program1.class**. Barulah, kita bisa menjalankan *file* Program1.class dengan mengetikkan kata kunci **java** diikuti dengan nama *file* .class yang ingin dijalankan (nama *file* tidak diikuti ekstensi .class) seperti contoh di bawah ini.

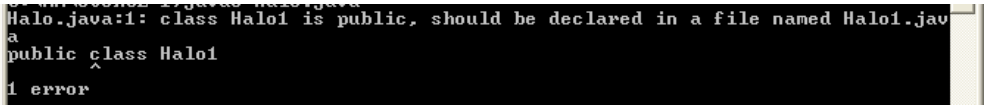
```
D:\Praktikum\Java\ java Program1
```

   Jika berhasil, maka pada *command prompt* tersebut akan muncul tulisan ”*Hello World*”.

B. Identifikasi kesalahan

Jika masih ada pesan kesalahan, periksa lagi program diatas, termasuk proses penyimpanannya. Beberapa kesalahan yang sering muncul :

1. Nama file yang disimpan tidak sesuai dengan nama class-nya.



2. Tidak ada void main pada program.

3. Berikan contoh jenis-jenis kesalahan yang lain dan tunjukkan cara memperbaikinya

C. Penggunakan variabel dan tipe data

Tuliskan program di bawah ini dan lakukan kompilasi. Apakah luaran yang dihasilkan?

```
1   public class Program1{
2         //Variabel class
3         int a,b;
4         double c;
5
6         public static void main(String[] args){
7               //Variabel lokal
8               int hasilTambah;
9               double hasilKali;
10
11              a = 10;
12              b = -5;
13              hasilTambah = a + b;
14              hasilKali = hasilTambah * c;
15
16              System.out.println("Hasil: " + hasilKali);
17        }
18  }
```

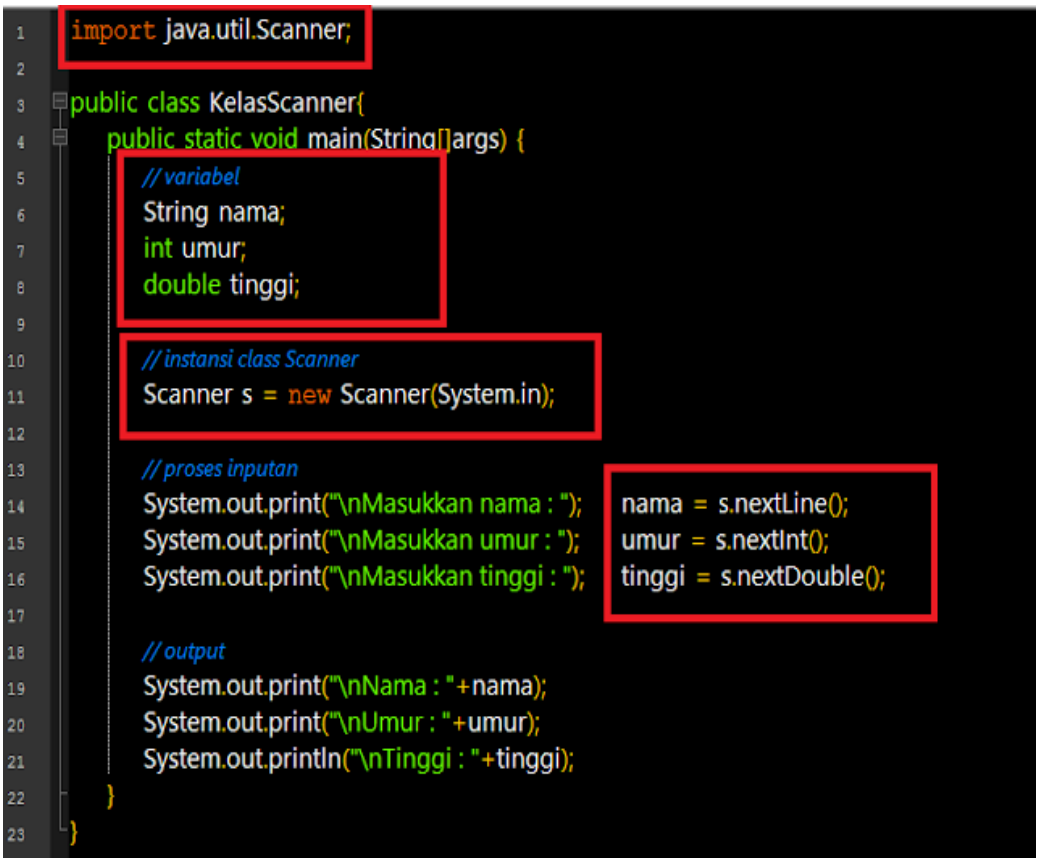
Penjelasan program

Baris ke-3 dan ke-4 pada kode program di atas merupakan deklarasi variabel global berupa variabel a dan b yang bertipe `int` serta variabel c yang bertipe `double`. Sedangkan baris ke-8 dan ke-9 merupakan deklarasi dari variabel lokal berupa variabel hasilTambah yang bertipe `int` dan variabel hasilKali yang bertipe `double` yang dideklarasikan di dalam blok program utama (*method* `main()`).

D. Memberikan input melalui keyboad dan menampilkan ke layar

Untuk mendapatkan inputan dari keyboard, ada 3 perintah input pemrograman java yang akan dibahas yaitu perintah input **BufferedReader**, perintah input **Scanner**, dan perintah input **JOptionPane.showInputDialog**. Sedangkan untuk perintah output program java ada 2 yaitu menggunakan **System.out.print** dan **JOptionPane.showMessageDialog**.

Pada modul ini akan dikenalkan perintah memasukkan data melalui keyboard menggunakan Scanner menampilkan input ke layar menggunakan System.out.print . Contoh program:



1.5 Tugas

1. Tulislah program di bawah ini dan lakukan kompilasi.

```
public class SayHelloC
{
public static void main(String args[])
{
  // deklarasi variabel lokal dalam  main method
    helloMessage = "Hello World";
    System.out.println(helloMessage);
    String helloMessage;
} // end of main method
}
```

a. Jika terjadi kesalahan pada saat kompilasi, tunjukkan pesan kesalahan yang ditampilkan pada layar
b. Apa arti pesan kesalahan tersebut?
c. Bagaimana cara memperbaikinya?

2. Buatlah sebuah program untuk menghitung harga akhir barang dari sebuah supermarket yang telah didiskon sebesar 25%! Input dari program adalah harga awal barang sebelum didiskon (melalui keyboard)

BAB 2
Percabangan dan Perulangan

2.1 Capaian Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan:
1. Memahami skema percabangan dan perulangan
2. Memahami kapan percabangan dan perulangan dapat digunakan

2.2 Indikator

1. Mahasiswa mampu menggunakan percabangan dan perulangan dalam pemrograman
2. Mahasiswa mampu memecahkan kasus yang bersifat percabangan dan perulangan

2.3 Uraian Materi

A. Percabangan

Di dalam Java, percabangan dapat diimplementasikan dalam dua jenis pernyataan: pernyataan `if` dan pernyataan `switch-case`. Keduanya memiliki fungsi yang sama, tetapi ada sedikit perbedaan di antara keduanya.

Pernyataan `if`

Pernyataan `if` dapat diimplementasikan untuk menangani percabangan yang didasarkan atas satu, dua, atau lebih dari dua kondisi. Pernyataan `if` digunakan apabila jumlah kondisi yang ada tidak terlalu banyak.

Bentuk `if` dengan satu kondisi adalah bentuk yang paling sederhana, mengandung suatu pernyataan tunggal yang dieksekusi jika kondisi yang ada terpenuhi. Sintaks dasarnya adalah sebagai berikut:

```
if (kondisi) {
     statement1;
     statement2;
     …
}
```

Pernyataan `if` dengan dua kondisi digunakan untuk mengambil keputusan berdasarkan dua kondisi. Apabila kondisi pertama terpenuhi, maka program akan mengeksekusi serangkaian *statement*, namun jika kondisi pertama tidak terpenuhi, maka program akan mengeksekusi *statement* lainnya. Berikut sintaks dasarnya:

```
if (kondisi) {
     //statement yang akan dilakukan jika kondisi benar
     Statement1;
     Statement2;
     …
} else {
     //statement yang akan dilakukan jika kondisi salah
     Statement1;
     Statement2;
     …
}
```

Apabila kondisi yang ada jumlahnya lebih dari dua, maka kita perlu menambahkan blok `else if` pada kode program kita. Sintaks dasarnya adalah sebagai berikut:

```
if (kondisi_1){
      //statement yang akan dilakukan jika kondisi_1 benar
      Statement1;
      Statement2;
      …
} else if (kondisi_2){
      //statement yang akan dilakukan jika kondisi_1 salah
      Statement1;
      Statement2;
      …
} else {
      //statement yang akan dilakukan jika kondisi_1 dan kondisi_2 salah
      Statement1;
      Statement2;
      …
}
```

Ada kalanya kondisi yang ada sangat banyak, mungkin lebih dari 5 kondisi, sehingga menulis kode percabangan menggunakan pernyataan `if` dirasa tidak efisien lagi, maka kita dapat menggunakan pernyataan `switch-case` sebagai alternatifnya. Berikut sintaks dasarnya:

```
switch (ekspresi) {
      case nilai1:
            //statement yang akan dilakukan jika ekspresi = nilai1
            statement1;
            statement2;
            break;
      case nilai2:
            //statement yang akan dilakukan jika ekspresi = nilai2
            statement1;
            statement2;
            break;
      …
      default:
            statement1;
            statement2;
}
```

B. **Perulangan**

Perulangan digunakan untuk melakukan sejumlah operasi yang sama secara berulang-ulang sampai tercapai kondisi berhentinya. Di dalam Java terdapat tiga skema perulangan: `for`, `while`, dan `do-while`. Masing-masing memiliki kegunaan tersendiri.

Skema perulangan `for` digunakan apabila jumlah perulangan yang akan dilakukan sudah diketahui. Sintaks dasarnya adalah sebagai berikut:

```
for (inisialisasi; kondisi; iterasi){
     //statement yang akan diulang
     ...
}
```

Secara fungsional, perulangan while sama saja dengan perulangan for, hanya saja perulangan while biasa digunakan apabila jumlah perulangan yang akan dilakukan belum diketahui pasti jumlahnya. Skema ini akan terus melakukan perulangan selama kondisi yang diset terpenuhi. Sintaks dasarnya adalah sebagai berikut:

```
While(kondisi){
    //statement yang akan diulang
    ...
    iterasi
}
```

Perulangan ini mirip dengan skema perulangan `while`, akan tetapi perbedaannya adalah pada proses pengecekan kondisi perulangan. Pada skema perulangan `while`, program akan mengecek kondisi terlebih dahulu, jika kondisi bernilai *true*, maka *statement* di dalam perulangan akan dieksekusi. Sebaliknya, pada skema perulangan `do-while`, program terlebih dahulu mengeksekusi *statement* di dalam perulangan, baru kemudian melakukan pengecekan kondisi. Jika kondisi bernilai *true*, maka program akan mengeksekusi kembali *statement* di dalam perulangan. Dari sini bisa disimpulkan bahwa perulangan `do-while` setidaknya akan mengeksekusi *statement* di dalam perulangan minimal satu kali apapun kondisi perulangannya. Berikut adalah sintaks umum skema perulangan `do-while`.

```
do {
    //statement yang akan diulang
    ...
    iterasi
} while(kondisi)
```

2.4 Latihan

1. Tuliskan program di bawah ini.

```
1  public class Percabangan2{
2     public static void main(String[] args){
3        //Variabel lokal
4        int a = 7;
5
6        if ((a > 0) && (a % 2 == 0)){
7           System.out.println(a + "adalah bilangan genap positif");
8        } else if ((a < 0) && (a % 2 == 0)){
9           System.out.println(a + "adalah bilangan genap negatif");
10       } else if ((a > 0) && (a % 2 != 0)){
11          System.out.println(a + "adalah bilangan ganjil positif");
12       } else {
13          System.out.println(a + "adalah bilangan ganjil negatif");
14       }
15    }
16 }
```

Program di atas menghasilkan output berupa kalimat ”7 adalah bilangan ganjil positif” karena kondisi_1 (baris ke-6) dan kondisi_2 (baris ke-8) tidak terpenuhi, sedangkan kondisi_3 (baris ke-10) terpenuhi.

a. Tulislah algoritma program tersebut
b. Apa luaran jika dilakukan kompilasi ?

2. Diberikan sebuah kode program:

```
1  public class Percabangan3{
2     public static void main(String[] args){
3        //Variabel lokal
4        int nomorHari = 2;
5
6        switch (nomorHari){
7           case 1:
8              System.out.println("Hari ke-" +nomorHari+ ": Senin");
9              break;
10          case 2:
11             System.out.println("Hari ke-" +nomorHari+ ": Selasa");
12             break;
13          default:
14             System.out.println("Tidak ada hari ke-" +nomorHari);
15       }
16    }
17 }
```

a. Apakah luaran program di atas?
b. Jika variabel `nomorHari` diisi melalui keyboard dengan nilai 3, maka program di atas akan menampilkan apa?

2.5 Latihan

1. Dari program contoh 2, tambahkan semua hari dalam seminggu.

2. Buatlah kode program untuk menentukan apakah berat badan seseorang ideal atau tidak. Berat badan ideal seseorang dapat dihitung dengan rumus berikut ini:
   a. BBI = (TB-100) - ((TB-100)*10%) , untuk pria (TB = Tinggi Badan)
   b. BBI = (TB-100) - ((TB-100)*15%) , untuk wanita.

   Jika berat badan seseorang lebih besar dari nilai BBI-nya, maka berat badannya tidak ideal. Buatlah kode programnya!

3. Jika diketahui sebuah kode program berikut ini:

```
1  public class Soal{
2        public static void main(String[] args){
3              int a, b;
4
5              a = 6;
6              b = -3;
7
8              if ((a + b) >= (b - a)){
9                    a += 2;
10                   b += a;
11                   if(a % 2 == 0){
12                         a = b;
13                   } else{
14                         b += a + 1;
15                   }
16             }
17
18             a = -b - a;
19       }
20 }
```

Tentukan nilai a dan b jika program di atas dieksekusi!

BAB 3
Kelas (*Class*), Objek (*Object)*, dan Class Diagram

3.1 Capaian Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan:
1. Mengetahui konsep kelas dan objek
2. Mengetahui class diagram

3.2 Indikator

1. Mampu menjelaskan kelas dan objek
2. Mampu membuat class diagram

3.3 Uraian Materi

A. Kelas (*class)*

Bahasa Java merupakan bahasa pemrograman yang berorientasi obyek sehingga konsep obyek dan kelas (class) menjadi penting. Dalam dunia nyata, ada banyak obyek misalnya obyek orang, obyek mobil, obyek pohon dsb. Bagaimana memindahkan obyek yang ada dalam dunia nyata menjadi obyek dalam pemrograman khususnya Java ?

Misalkan obyek dalam dunia nyata adalah obyek orang. Obyek tersebut dapat diceritakan tentang ciri-cirinya , yaitu tinggi badan, berat badan, warna rambut, warna kulit, jenis kelami, menggunakan kacamata dll. Dalam pemrograman , obyek didunia nyata akan menjadi CLASS, dan ciri-ciri obyek akan menjadi variabel class yang disebut sebagai DATA MEMBER.

Berdasarkan contoh diatas, pendefinisian pada pemrograman adalah :

```
Class Handphone
{
   String merk;
   String tipe;
   String warna;
   double harga;
}
```

```
class Mobil
{
      // Variabel class
      private int warna;
      private      String
merek;
}
```

```
Class Orang
   {
     String nama;
     Int tinggi_badan;
     Int berat_badan;
     String warna_kulit;
     Boolean berkacamata;
   }
```

Struktur pembuatan class, adalah sebagai berikut:

```
1   class NamaClass
2   {
3         //Isi class
4   }
```

Keterangan:

- **Nama_Kelas** harus sesuai dengan nama file.
- Contoh: **class Handphone**, maka nama filenya harus diberi nama dengan **Handphone.java**.

B. Atribut

Atribut merupakan ciri-ciri yang melekat pada suatu objek. Berikut adalah contoh *syntax* atribut.

[access_modifier] [tipe_data] [nama_variabel] = **[value]**;

Keterangan:

- **[access_modifier]** digunakan untuk memberi batasan hak *class* maupun *method*. *Access modifier* akan dijelaskan pada sub bab berikutnya
- **[tipe_data]** menjelaskan apakah variabel tersebut bertipe *String, int, double,* dan sebagainya
- **[nama_variabel]** merupakan sebutan (definisi) variabel tersebut
- **[value]** merupakan nilai dari variable tersebut
  Contoh: **private String warna = "merah";**

C. Method

Method merupakan tindakan/aksi yang bisa dikerjakan oleh CLASS. **Method** merupakan fungsi-fungsi yang digunakan untuk memanipulasi nilai-nilai pada atribut dan/atau untuk melakukan hal-hal yang dapat dilakukan oleh objek itu sendiri. Dalam hal ini method dapat berisi sekumpulan program yang telah terbungkus. Dengan method, kita bisa memanggil kumpulan program tersebut hanya dengan memanggil nama methodnya sehingga pekerjaan jadi lebih singkat dan tidak boros menuliskan program. Selain itu, program menjadi lebih terstruktur, praktis, dan efisien. Contoh:

METHOD Class Handphone	**METHOD Class Mobil**	**METHOD Class Orang**
- Memasukan warna - Memasukkan tipe - Memasukkan merek	- memasukan data mobil - gerak mobil belok kiri - menampilkan informasi	-memasukkan data orang - tertawa -menangis -menampilkan informasi

Method yang mengembalikan nilai biasanya berupa **sub program berjenis fungsi**. Sedangkan method yang tidak mengembalikan nilai biasanya berupa **sub program berjenis prosedur**.

Berikut adalah contoh *syntax* pembuatan method.

[access_modifier] [tipe_data] nama_method(……)

Keterangan:

- **[access_modifier]** digunakan untuk memberi batasan hak *class* maupun *method*. *Access modifier* akan dijelaskan pada sub bab berikutnya
- **[tipe_data]** menjelaskan apakah variabel tersebut bertipe *String, int, double*, dan sebagainya
- **[nama_method]** merupakan sebutan (definisi) method tersebut. Umumnya method selalu diakhiri dengan tanda kurung **()**
- **(…….)** berisi parameter apabila diperlukan.

Contoh

```
public void masuk_info_mobil(int mrk, String nm)
      {
            this.merek = mrk;
```

Implementasi method dalam pemrograman adalah:

```
public void menangis()
            {
```

```
public void  tertawa()
            {
```

D. Objek

Dalam pemrograman, suatu class dapat mempunyai banyak objek dan objek akan mewarisi data member dan method yang sama dari suatu class. Objek disebut juga i***nstance of Class*** merupakan objek yang diinstan atau dibuat dari *class*.

Untuk membuat obyek dari class Orang, digunakan keyword ‘new’. Contoh :

```
Orang   org1 = new Orang (“Samuel”);

Orang   org2 = new Orang (“Ari”);
```

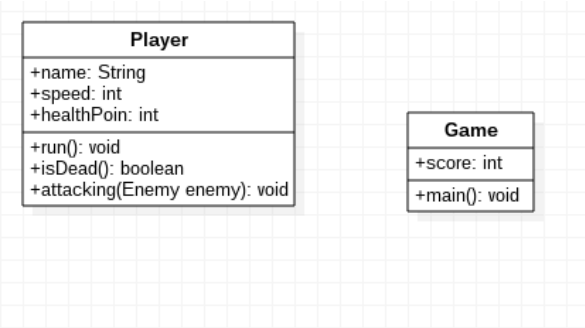
Sehingga class Orang , saat ini mempunyai 2 obyek.

E. Kata Kunci ‘this’

Kata kunci **this** digunakan untuk membedakan variabel yang dideklarasikan pada parameter di dalam method dengan variabel yang dideklarasikan pada *class*. Untuk penggunaan **this** dapat anda lihat pada soal latihan.

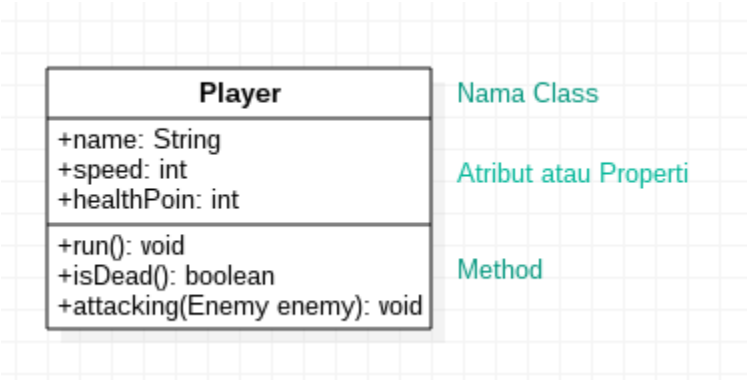
F. Class Diagram

Class Diagram adalah sebuah diagram yang menggambarkan hubungan antar *class*. *Class Diagram* dapat kita buat dengan aplikasi perancangan (CASE), seperti StarUML.

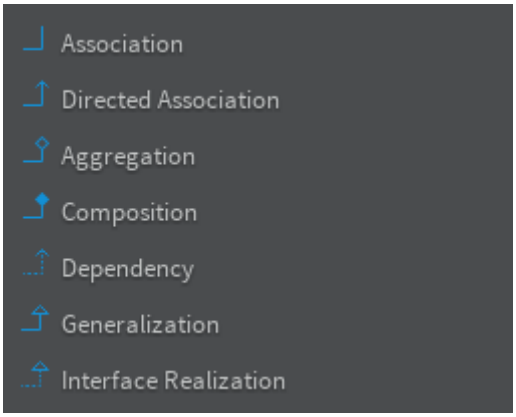


Sebauh *class* digambarkan dengan sebuah tabel 1 kolom dan 3 baris.

Baris pertama berisi nama *class*; Baris kedua berisi atribut; dan Baris ketiga berisi method.



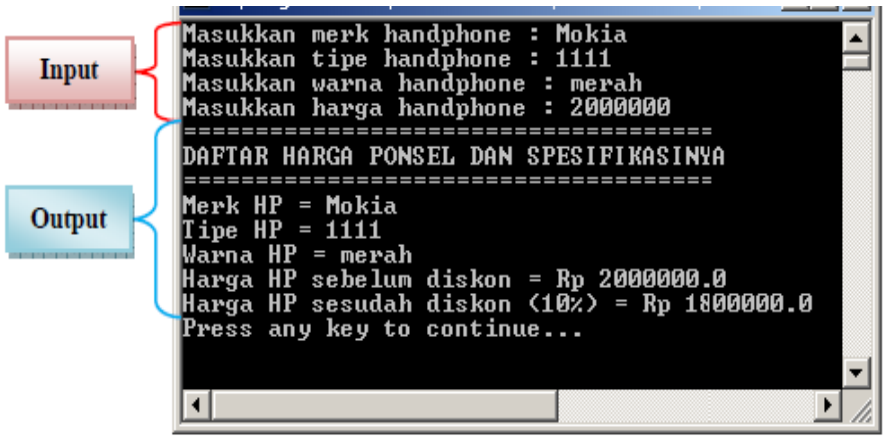
Selain itu, terdapat garis yang menggambarkan hubungan antar *class*.



Class Diagram biasanya digunakan oleh *software engineer*untuk merancang software dengan paradigma OOP.

3.4 Latihan

Berdasarkan contoh di atas tentang handphone, buatlah 2 buah class yang terdiri dari class handphone dan class utama. Class utama digunakan untuk memanggil class handphone. Ketika class utama dijalankan, hasilnya akan tampak seperti di bawah ini:



Sedangkan pada class handphone harus memiliki beberapa ketentuan sebagai berikut:

Atribut berisi **merk, tipe, warna,** dan **harga**

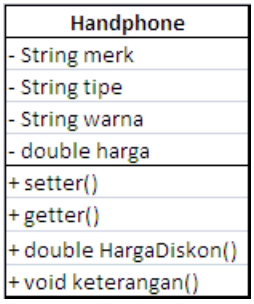
- Terdapat method **setter** dan **getter** untuk mengeset dan mengambil nilai dari **merk, tipe, warna,** dan **harga**
- Terdapat method **HargaDiskon()** untuk menghitung harga handphone sesudah diskon. Diskon yang diperoleh adalah 10%
- Terdapat method **keterangan()** untuk mencetak *statement* tentang harga handphone sesudah diskon

Jawab:

Langkah 1: Membuat skema

Skema diagram digunakan untuk membantu anda dalam membantu logika anda untuk pembuatan program. Tanda "-" dilambangkan sebagai ***private***.

Sedangkan tanda "+" dilambangkan sebagai ***public***. Berikut adalah skema diagramnya.



Langkah 2: class Handphone (ketikkan script berikut)

a. Membuat kerangka class Handphone

Setelah anda membuat class Handphone, simpan file tersebut dengan nama **Handphone.java**. Di dalam class Handphone, juga menyediakan tempat untuk mendeklarasikan variabel, setter dan getter.

b. Mendeklarasi variabel yang dibutuhkan

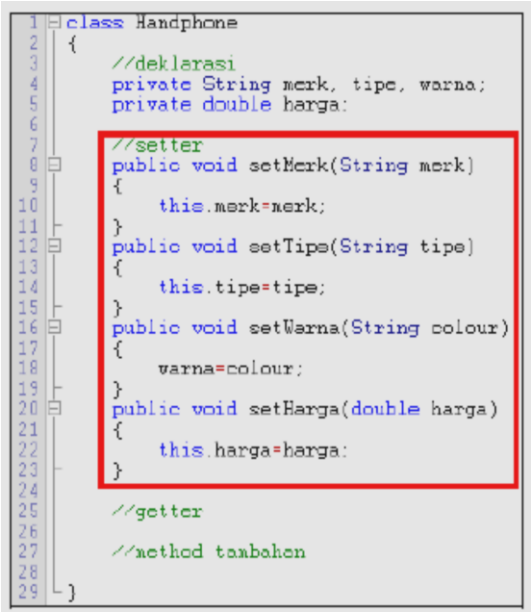
Setelah kita membuat kerangka class, maka diperlukan pendeklarasian variabel yang nantinya digunakan sebagai tempat menyimpan data yang bersifat sementara (*temporary*). Gambar di bawah ini menunjukkan pendeklarasian variabel.

```
1  class Handphone
2  {
3      //deklarasi
4      private String merk, tipe, warna;
5      private double harga;
6
7      //setter
8
9      //getter
10
11     //method tambahan
12
13 }
```

c. Membuat method setter

Setelah membuat variabel yang dibutuhkan pada class Handphone, langkah selanjutnya anda buat method setter untuk masing-masing variabel. Hal ini bertujuan untuk mengeset nilai yang diperoleh dari class Utama yang nantinya akan kita gunakan ke dalam class Handphone. Perlu diketahui pula, bahwa dalam pembuatan variabel pada parameter di method setter, harus sesuai dengan tipe data pada variabel yang dideklarasikan di kelas.

Gambar di bawah ini menunjukkan deklarasi setter.



```
class Handphone
{
    //deklarasi
    private String merk, tipe, warna;
    private double harga;

    //setter
    public void setMerk(String merk)
    {
        this.merk=merk;
    }
    public void setTipe(String tipe)
    {
        this.tipe=tipe;
    }
    public void setWarna(String colour)
    {
        warna=colour;
    }
    public void setHarga(double harga)
    {
        this.harga=harga;
    }

    //getter

    //method tambahan
}
```

Sebagai tambahan informasi, dalam pembuatan method setter, kita menggunakan sub program berjenis prosedur. Hal ini dikarenakan data yang akan kita set, tidak terdapat umpan balik ke dalam program. Handphone (lihat *script* yang diberi kotak berwarna biru). Apabila variabel tersebut tersebut tidak diberi keyword ***this***, maka variabel tersebut akan mengacu kepada variabel yang dideklarasikan pada parameter method setter (lihat *script* yang diberi kotak berwarna hijau). Anda bisa menggunakan keyword ***this*** atau tidak apabila ada perbedaan deklarasi nama variabel pada class Handphone dengan parameter pada method setter (lihat *script* yang diberi kotak berwarna ungu).

d. Membuat method getter

Setelah membuat method setter, anda tinggal membuat method getter untuk mengambil nilai dari masing-masing variabel. Hal ini bertujuan untuk mengambil nilai dari variabel pada class Handphone yang nantinya akan kita kembalikan ke dalam class Utama. Perlu diketahui pula, bahwa dalam pembuatan variabel pada parameter di method getter, harus sesuai dengan tipe data pada variabel yang dideklarasikan di kelas. Gambar di bawah ini menunjukkan deklarasi getter.

```
class Handphone
{
    //deklarasi
    private String merk, tipe, warna;
    private double harga;

    //setter
    public void setMerk(String merk)
    public void setTipe(String tipe)
    public void setWarna(String colour)
    public void setHarga(double harga)

    //getter
    public String getMerk()
    {
        return merk;
    }
    public String getTipe()
    {
        return tipe;
    }
    public String getWarna()
    {
        return warna;
    }
    public double getHarga()
    {
        return harga;
    }


    //method tambahan

}
```

Sebagai tambahan informasi, dalam pembuatan method getter, kita menggunakan sub program berjenis fungsi karena dibutuhkan umpan balik dalam pengambilan data.

d. Membuat method tambahan

Seperti namanya, method ini hanya sebagai tambahan apabila ada permintaan soal untuk mengolah data-data yang telah kita set dan get ke dalam bentuk informasi. Seperti soal yang diminta, anda diminta untuk menghitung dan mencetak harga handphone sesudah diskon. Gambar di bawah ini menunjukkan pembuatan method HargaDiskon() dan method keterangan().

```
class Handphone
{
    //deklarasi
    private String merk, tipe, warna;
    private double harga;

    //setter
    public void setMerk(String merk)
    public void setTipe(String tipe)
    public void setWarna(String colour)
    public void setHarga(double harga)

    //getter
    public String getMerk()
    public String getTipe()
    public String getWarna()
    public double getHarga()

    //method tambahan
    public double HargaDiskon()
    {
        double diskon = 0.1 * getHarga();
        double total = getHarga() - diskon;

        return total;
    }

    public void keterangan()
    {
        System.out.println ("Harga HP sesudah diskon (10%) = Rp " + HargaDiskon());
    }
}
```

♉ Langkah 3: class Utama (ketikkan script berikut)

a. Membuat kerangka class Utama

Setelah anda membuat class Utama, simpan file tersebut dengan nama

Utama.java. Di dalam class inilah, program anda akan dijalankan.

```
class Utama
{
    public static void main (String [] args)
    {
        //instance of class

        //input

        //output

    }
}
```

b. Membuat *instance of class*

Setelah anda membuat class Utama, langkah berikutnya yang anda lakukan adalah membuat sebuah objek yang bertipe class Handphone. Itulah yang dinamakan **instance of class** (untuk penjelasannya, dapat anda lihat pada sub bab sebelumnya). Misalkan, objek yang saya buat adalah **hp**, maka penulisan *script*-nya adalah sebagai berikut.

```
class Utama
{
    public static void main (String [] args)
    {
        //instance of class
        Handphone hp = new Handphone();

        //input

        //output

    }
}
```

c. Membuat inputan yang diisi user

Sebelum anda membuat inputan yang nantinya akan diisi oleh user, anda dapat menggunakan *class* yang dapat digunakan untuk menerima inputan, salah satunya adalah class **BufferedReader** yang terdapat pada **package java.io**. Untuk mengakses class BufferedReader, anda harus mengimport class tersebut. (boleh juga menggunakan class Scanner). Berikut adalah contoh *script*-nya.

```
import java.io.*;
class Utama
{
    public static void main (String [] args) throws Exception
    {
        BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

        //instance of class
        Handphone hp = new Handphone();

        //input

        //output

    }
}
```

Setelah itu, buatlah sebuah perintah yang akan dicetak oleh program, yang nantinya user dapat mengetahui apa saja yang harus ia lakukan ketika program dijalankan. Setiap inputan dari user, kemudian akan ditampung ke dalam variabel (lihat *script* yang diberi kotak berwarna merah). Setelah ditampung ke dalam variabel, maka data tersebut akan di set satu per satu ke dalam class Handphone (lihat *script* yang diberi kotak berwarna biru). Berikut adalah contoh *script*-nya.

```
import java.io.*;
class Utama
{
    public static void main (String [] args) throws Exception
    {
        BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

        //instance of class
        Handphone hp = new Handphone();

        //input
        System.out.print("Masukkan merk handphone : ");
        String merk_hp = br.readLine();
        System.out.print("Masukkan tipe handphone : ");
        String tipe_hp = br.readLine();
        System.out.print("Masukkan warna handphone : ");
        String warna_hp = br.readLine();
        System.out.print("Masukkan harga handphone : ");
        double harga_hp = Double.parseDouble (br.readLine());

        hp.setMerk(merk_hp);
        hp.setTipe(tipe_hp);
        hp.setWarna(warna_hp);
        hp.setHarga(harga_hp);

        //output

    }
}
```

Catatan:

Cara menge-*set* data ke dalam class Handphone dengan format sebagai berikut:

Nama_Objek.Nama_Method

Cara ini berlaku juga untuk method get() maupun method lainnya

d. Membuat output

Ini adalah langkah terakhir. Ketika data sudah diinput semua, maka diperlukan output dari hasil tampilan program tersebut. Untuk mengambil data-datanya, anda cukup menggunakan method **get()** dalam hal pengambilan data. Berikut adalah contoh *script*-nya.

```
1  import java.io.*;
2  class Utama
3  {
4      public static void main (String [] args) throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //instance of class
9          Handphone hp = new Handphone();
10
11         //input
12         System.out.print("Masukkan merk handphone : ");
13         String merk_hp = br.readLine();
14         System.out.print("Masukkan tipe handphone : ");
15         String tipe_hp = br.readLine();
16         System.out.print("Masukkan warna handphone : ");
17         String warna_hp = br.readLine();
18         System.out.print("Masukkan harga handphone : ");
19         double harga_hp = Double.parseDouble (br.readLine());
20
21         hp.setMerk(merk_hp);
22         hp.setTipe(tipe_hp);
23         hp.setWarna(warna_hp);
24         hp.setHarga(harga_hp);
25
26         //output
27         System.out.println("=====================================");
28         System.out.println("DAFTAR HARGA PONSEL DAN SPESIFIKASINYA");
29         System.out.println("=====================================");
30         System.out.println("Merk HP = "+hp.getMerk());
31         System.out.println("Tipe HP = "+hp.getTipe());
32         System.out.println("Warna HP = "+hp.getWarna());
33         System.out.println("Harga HP sebelum diskon = Rp "+hp.getHarga());
34         hp.keterangan();
35     }
36 }
```

Coba anda perhatikan script pada *line 34*! Pada *script* di atas, anda cukup memanggil nama methodnya saja, tanpa perlu mengetik lagi. Hal ini menunjukkan bahwa penulisan **hp.keterangan()** sama halnya dengan anda mengetikkan **System.out.println ("Harga HP sesudah diskon (10%) = Rp " + HargaDiskon());** pada class Handphone.

3.5 Tugas

1. Ketiklah program berikut dan lakukan kompilasi.

```
// MEMBUAT KELAS MOBILKU
   class Mobilku
   {
      // Variabel class
      private int merek;
      private String nama;

      // informasi mobil --> METHOD
      public void masuk_info(int mrk , String nm)
      {
            this.merek = mrk;
            this.nama = nm;
      }
      public void info()
      {
            System.out.println("No   = "+this.merek);
            System.out.println("Nama = "+this.nama);
      }
  }


// MEMBUAT KELAS MOBIL YANG MEMAKAI KELAS MOBILKU
public class mobil
{
    public static void main(String args[])
    {
      // buat 2 buah obyek mobilku yaitu mobil1 dan mobil2
      mobilku mobil1 = new mobilku();//obyek 1 bernama mobil1
      mobilku mobil2 = new mobilku();//obyek 2 bernama mobil2

      // memberi nomor dan nama obyek mobil1 dan mobil2
      mobil1.masuk_info(1, "HONDA");
      mobil2.masuk_info(2, "TOYOTA");

       // Panggil  method
       mobil1.info();
       mobil2.info();
    }
 }
```

2. Buat skema class Mobilku
3. Apa nama file untuk menyimpan program tersebut?
4. Apa luaran program tersebut?
5. Tambahkan method supaya mobil bisa diidentifikasi jenisnya
6. Tambahkan method lain supaya mobil 1 dan mobil 2 mempunyai warna yang berbeda
7. Buatlah program untuk menggunakan class Orang seperti di atas.

BAB 4
Enkapsulasi

4.1 Capaian Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan mengetahui konsep enkapsulasi dan implementasinya pada PBO

4.2 Indikator

1. Mampu menjelaskan konsep PBO
2. Mampu mengimplementasikan pada PBO

4.3 Uraian Materi

Encapsulation adalah salah satu dari empat konsep OOP fundamental. Tiga lainnya adalah pewarisan, polimorfisme, dan abstraksi. Enkapsulasi adalah mekanisme membungkus data (variabel) dan kode yang bekerja pada data (methode) yang bersama-sama sebagai satu kesatuan. Didalam enkapsulasi, variabel kelas akan disembunyikan dari kelas-kelas lain, dan dapat diakses hanya melalui method kelas itu sendiri. Oleh karena itu, juga dikenal sebagai persembunyian data.

Untuk membuat enkapsulasi di Java:

- Mendeklarasikan variabel kelas sebagai private.
- Menyediakan method setter dan getter umum untuk memodifikasi dan melihat nilai-nilai variabel. Contoh:

```
public class SegiEmpat
{    private int panjang;
     private int lebar;

     public void setPanjang(int pj){
         this.panjang=pj;
     }
     public void setLebar(int lb){
         this.lebar=lb;
     }
     public int getPanjang(){
         return this.panjang;
     }
     public int getLebar(){
         return this.lebar;
     }
     public int luas(){
         return this.panjang*this.lebar;
     }
   }
```

public `setXXX()` dan `getXXX()` adalah *method* akses dari variabel instance dari kelas `EncapTest`. Biasanya, *method* ini disebut sebagai *getter* dan *setter*. Oleh karena itu, setiap kelas yang ingin mengakses variabel harus mengaksesnya melalui *getter* dan *setter*.

Keuntungan **Encapsulation**

- Atribut/properti dalam kelas bisa dibuat hanya-baca atau menulis saja.
- Sebuah kelas dapat memiliki total kontrol atas apa yang disimpan dalam atribut/properti.
- Pengguna kelas tidak tahu bagaimana kelas menyimpan datanya. Sebuah kelas dapat mengubah tipe data dari properti dan pengguna kelas tidak perlu mengubah apapun.

4.4 Latihan

1. Buatlah proyek baru, dan kelas EncapTest.java seperti pada contoh program berikut:

Program EncapTest.java

```
public class EncapTest {
   private String name;
   private String idNum;
   private int age;

   public int getAge() {
      return age;
   }

   public String getName() {
      return name;
   }

   public String getIdNum() {
      return idNum;
   }
```

Program EncapTest.java

2. Untuk memanggil kelas tersebut tambahkan/buatlah `TestEnscapsulasi` seperti pada program utuma di bawah ini:

Program TestEnscapsulasi.java

```
/* File name : RunEncap.java */
public class RunEncap {

   public static void main(String args[]) {

      EncapTest encap = new EncapTest();

      encap.setName("James");

      encap.setAge(20);

      encap.setIdNum("12343ms");

      System.out.print("Name : " + encap.getName() + " Age : " +
encap.getAge());   }

}
```

4.5 Tugas

1. Buatlah program yang memanfaatkan sifat enkapsulasi untuk menghitung keliling dan luas lingkaran.
2. Tambahan tugas dari dosen pengampu.

BAB 5
Constructor

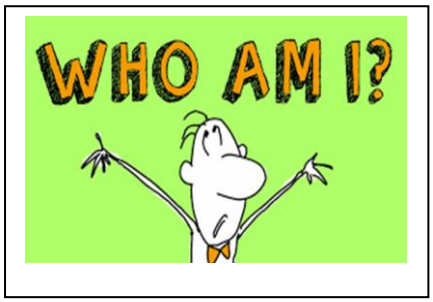
5.1 Capaian Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan:
1. Mengetahui konsep constructor
2. Mengetahui implemtasi constructor dalam program

5.2 Indikator

1. Mampu menjelaskan konsep constructor
2. Mampu mengimplementasikan constructor di dalam program

5.3 Uraian Materi



Setiap manusia pasti memiliki nama. Sadar atau tidak, anda akan memperkenalkan diri anda dengan menyebutkan nama, alamat, tanggal lahir, hobi, dan sebagainya. Namun, pernahkah anda berpikir mengapa banyak orang suka menghapal nama anda ketimbang alamat, tanggal lahir, hobi, dan sebagainya? Padahal terkadang nama banyak dipakai juga pada orang lain? Bahkan nama lengkap saya sendiri pun juga ada yang menggunakannya meskipun ejaannya berbeda. Itu semua karena nama anda adalah unik. Sekali lagi, coba anda bayangkan bagaimana kalau anda tidak mempunyai nama? Orang lain pasti akan kebingungan memanggil anda. Bisa-bisa anda akan dipanggil "Anonymous" sebagai seseorang tanpa nama. Demikian pula dalam pembuatan constructor. Ketika anda membuat sebuah objek dari class manusia (sebut saja "Orang 1"), kemudian anda dapat menge-set nilai berupa nama, alamat, tanggal lahir, dan hobi menggunakan method setter(). Berbeda halnya dengan constructor. **Ketika objek "Orang 1" telah terbentuk, anda langsung memberikan nilai** berupa nama, alamat, tanggal lahir, dan hobi. Hal itu ibarat anda baru lahir di dunia ini dan langsung diberi nama. Itulah merupakan konsep dari Constructor.

A. Constructor

Constructor adalah method yang berfungsi untuk menginisialisasi variabelvariabel instans yang akan dimiliki oleh objek. Constructor dipanggil pada saat proses instansiasi kelas menjadi objek. Jika kelas tidak memiliki method constructor, maka seluruh variabel objek akan diinisialisasi kepada nilai default, sesuai dengan tipe datanya masing-masing. Berikut adalah struktur constructor.

```
class Nama_Kelas
{
    Nama_Kelas()
    {
        //isi constructor
    }

    // isi dari kelas
}
```

Contoh penggunaan constructor:

```
class Login
{
    Login()
    {
        //isi constructor
    }

    // isi dari kelas
}
```

B. Karakteristik Constructor

Berikut ini adalah beberapa karakteristik yang dimiliki oleh constructor:

1. Method constructor harus memiliki nama yang sama dengan nama class
2. Tidak mengembalikan suatu nilai (tidak ada keyword *return*)
3. Satu kelas memiliki lebih dari constructor (*overloading constructor*)
4. Dapat ditambah *access modifier* public, private, protected maupun default
5. Suatu constructor bisa dipanggil oleh constructor lain dalam satu kelas

C. Overloading Constructor

Seperti yang telah dijelaskan poin 3 pada karakteristik constructor bahwa dalam sebuah class dapat memiliki lebih dari satu constructor. Yang membedakan antara constructor yang satu dengan yang lainnya adalah jumlah parameter dan tipe data di dalamnya. Struktut Overloading Constructor adalah sebagai berikut.

```
class Nama_Kelas
{
    Nama_Kelas()
    {
        //isi constructor
    }

    Nama_Kelas(parameter)
    {
        //isi constructor
    }

    // isi dari kelas
}
```

Contoh penggunaan constructor:

```
1  class Login
2  {
3      private String username, password;
4
5      Login()
6      {
7          username = "admin";
8          password = "12345";
9      }
10
11     Login(String username, String password)
12     {
13         this.username = username;
14         this.password = password;
15     }
16
17     // isi dari kelas
18 }
```

Pada contoh di atas, anda melihat class Login dimana memiliki 2 constructor. Selain itu, terdapat perbedaaan cara membuat objek (*instance of class*) dengan menggunakan overloading constructor di class Utama. Jika anda ingin membuat *instance of class* dengan method Login tanpa parameter adalah sebagai berikut:

```
Login user1 = new Login();
```

Pembuatan objek di atas akan memberikan nilai set default username dan password berupa admin dan 12345. Sedangkan jika anda ingin membuat user sendiri (misal: username=edo, password=pb0) melalui *instance of class* dengan method Login menggunakan parameter adalah sebagai berikut:

Login user1 = new Login(“edo”, “pb0”);

D. Overloading

Overloading merupakan suatu mekanisme yang memungkinkan kita untuk membuat lebih dari satu *method* dengan nama yang sama. Namun dengan ketentuan bahwa setiap *method* harus memiliki parameter yang berbeda, baik berbeda dari nama parameternya, atau berbeda dari tipe datanya, atau bahkan berbeda jumlah parameternya. *Overload* dapat dilakukan terhadap *constructor* dan juga *method*.

Overload pada Constructor

Pada kenyataannya, suatu kelas pada umumnya justru memiliki lebih dari satu *constructor*. Artinya, kita dapat melakukan instansiasi objek dengan menggunakan salah satu dari *constructor* yang ada pada kelas yang dirujuk.

Overload pada Method

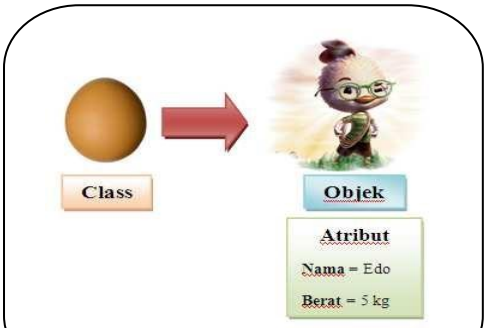
Sama halnya dengan *constructor*, *method* juga dapat dikenai *overload* dengan syarat bahwa kedua dua method yang namanya sama harus memiliki parameter yang berbeda

E. Perbedaan menggunakan Constructor dengan method Setter()

Jika anda melihat dengan teliti, anda pasti akan bertanya-tanya “lalu apa perbedaan penggunaan setter() dengan constructor?”. Dalam method setter(), pertama kali obyek dibuat dari sebuah kelas (*instance of class*). Setelah objek terbentuk, kemudian objek tersebut diberi atribut. Berbeda dengan constructor.

Dalam constructor, obyek yang dibuat dari sebuah kelas (*instance of class*) langsung diberi atribut. Berikut adalah analogi perbedaan setter() dan constructor.

Analogi setter()



Analogi constructor

5.4 Latihan

Berdasarkan contoh di atas tentang login, buatlah 2 buah class yang terdiri dari class Login dan class utama.

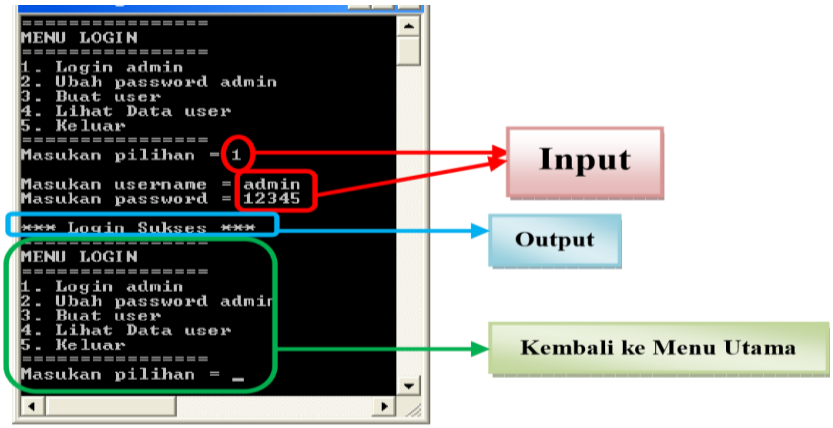
Class Login harus memiliki beberapa ketentuan sebagai berikut:

- Atribut berisi **username** dan **password**
- Terdapat 2 buah constructor Login. Constructor pertama tidak memiliki parameter dan memiliki nilai default username=”admin” dan password=”12345”. Sedangkan constructor kedua memiliki parameter untuk mengeset nilai username dan password berdasarkan inputan user.
- Terdapat method **setter** dan **getter** untuk mengeset/merubah dan mengambil nilai dari **username dan password**

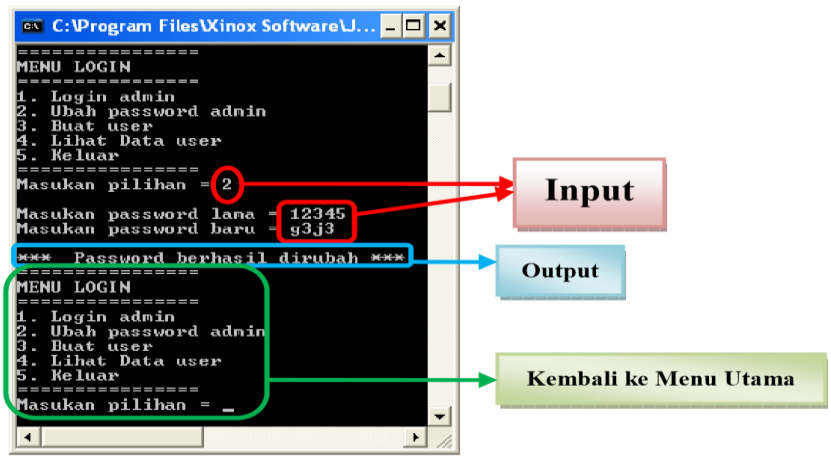
Sedangkan pada Class utama digunakan untuk memanggil class Login.

Ketika class utama dijalankan, hasilnya akan tampak seperti di bawah ini:

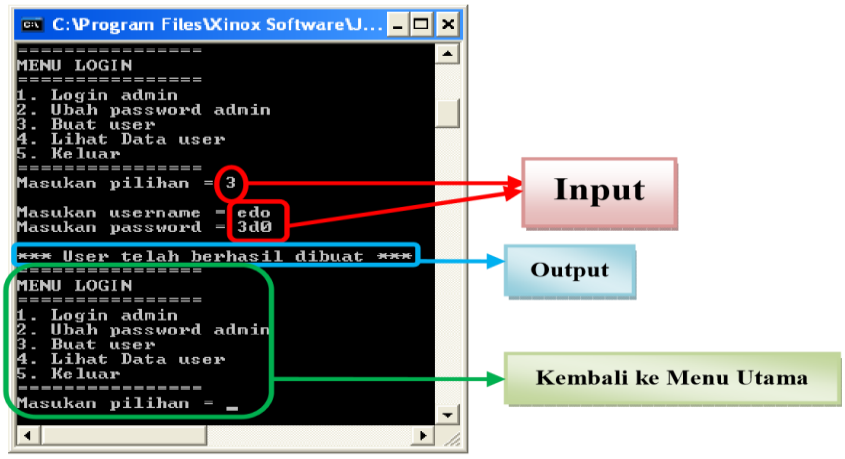
Jika pilihan = 1, maka akan tampil sebagai berikut:



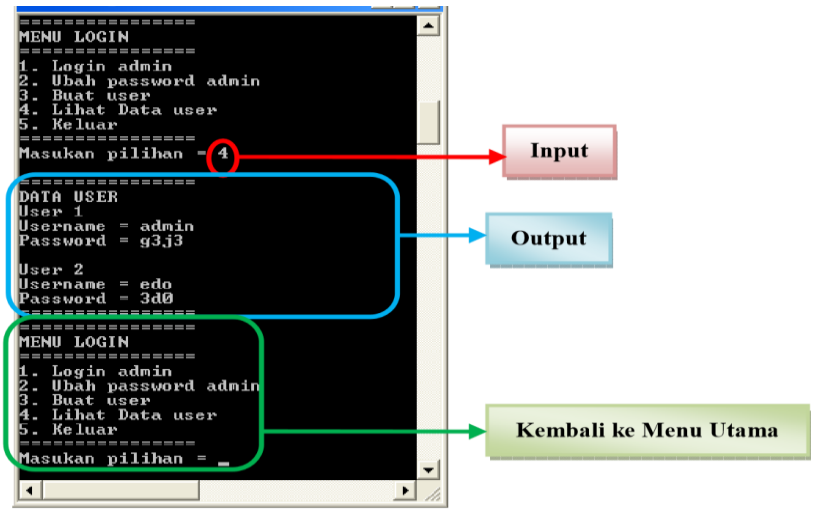
Jika pilihan = 2, maka akan tampil sebagai berikut:



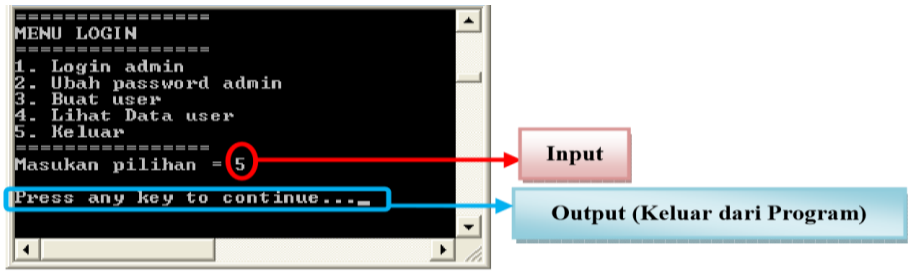
Jika pilihan = 3, maka akan tampil sebagai berikut:



Jika pilihan = 4, maka akan tampil sebagai berikut:



Jika pilihan = 5, maka akan tampil sebagai berikut:



Jawabannya adalah…

Langkah 1: Membuat skema

Skema diagram digunakan untuk membantu anda dalam membantu logika anda untuk pembuatan program. Tanda "-" dilambangkan sebagai ***private***. Sedangkan tanda "+" dilambangkan sebagai ***public***. Berikut adalah skema diagramnya.

Login
- String username
- String password
+ Login()
+ Login(String username, String Password)
+ setter()
+ getter()

Langkah 2: class Login (ketikkan script berikut)

a. Membuat kerangka class Login

```
1  class Login
2  {
3      //deklarasi
4
5      //constructor
6
7      //setter
8
9      //getter
10
11 }
```

Setelah anda membuat class Login, simpan file tersebut dengan nama **Login.java**. Di dalam class Login, saya juga menyediakan tempat untuk mendeklarasikan variabel, setter dan getter.

b. Mendeklarasi variabel yang dibutuhkan

Setelah kita membuat kerangka class, maka diperlukan pendeklarasian variabel yang nantinya digunakan sebagai tempat menyimpan data yang bersifat sementara (*temporary*). Gambar di bawah ini menunjukkan

pendeklarasian variabel.

```
1  class Login
2  {
3      //deklarasi
4      private String username, password;
5
6      //constructor
7
8      //setter
9
10     //getter
11
12 }
```

c. Mendeklarasi constructor

Setelah membuat variabel yang dibutuhkan pada class Login, langkah selanjutnya anda membuat constructor login. Constructor ini nantinya akan digunakan dalam class Utama. Gambar di bawah ini menunjukkan deklarasi constructor.

```
1  class Login
2  {
3      //deklarasi
4      private String username, password;
5
6      //constructor
7      public Login()
8      {
9          username = "admin";
10         password = "12345";
11     }
12
13     public Login(String username, String password)
14     {
15         this.username = username;
16         this.password = password;
17     }
18
19     //setter
20
21     //getter
22
23 }
```

Coba perhatikan kembali script yang telah anda buat seperti gambar di atas. Seperti halnya dengan penggunaan method setter(), di dalam pembuatan constructor Login dengan parameter, anda juga dapat menggunakan keyword ***this***. Penggunaan keyword ***this*** akan mengacu kepada variabel yang dideklarasikan pada class Login (lihat *script* yang diberi kotak berwarna biru pada gambar di bawahnya). Apabila variabel tersebut tersebut tidak diberi keyword ***this***, maka variabel tersebut akan mengacu kepada variabel yang dideklarasikan pada parameter constructor (lihat *script* yang diberi kotak berwarna hijau pada gambar di bawahnya). Penggunaan keyword ***this*** dapat digunakan atau tidak (*optional*) apabila ada perbedaan deklarasi nama variabel pada class Login (untuk lebih jelas mengenai keyword ***this***, anda dapat melihat modul 1).

```
1  class Login
2  {
3      //deklarasi
4      private String username, password;
5
6      //constructor
7      public Login()
8      {
9          username = "admin";
10         password = "12345";
11     }
12
13     public Login(String username, String password)
14     {
15         this.username = username;
16         this.password = password;
17     }
18
19     //setter
20
21     //getter
22
23 }
```

d. Membuat method setter

Setelah membuat variabel yang dibutuhkan pada class Login, langkah selanjutnya anda buat method setter untuk masing-masing variabel. Hal ini bertujuan untuk mengeset atau merubah nilai variabel username dan password sesuai dengan permintaan soal

pada menu yang ke-2 di class Utama nanti. Perlu diketahui pula, bahwa dalam pembuatan variabel pada parameter di method setter, harus sesuai dengan tipe data pada variabel yang dideklarasikan di kelas. Gambar di bawah ini menunjukkan deklarasi setter.

```
class Login
{
    //deklarasi
    private String username, password;

    //constructor
    public Login()
    {
        username = "admin";
        password = "12345";
    }

    public Login(String username, String password)
    {
        this.username = username;
        this.password = password;
    }

    //setter
    public void setUsername(String username)
    {
        this.username=username;
    }
    public void setPassword(String password)
    {
        this.password=password;
    }

    //getter

}
```

e. Membuat method getter

Setelah membuat method setter, anda tinggal membuat method getter untuk mengambil nilai dari masing-masing variabel. Hal ini bertujuan untuk mengambil nilai dari variabel pada class Login yang nantinya akan kita kembalikan ke dalam class Utama. Perlu diketahui pula, bahwa dalam pembuatan variabel pada parameter di method getter, harus sesuai dengan tipe data pada variabel yang dideklarasikan di kelas. Gambar di bawah ini menunjukkan deklarasi getter.

```
class Login
{
    //deklarasi
    private String username, password;

    //constructor
    public Login()

    public Login(String username, String password)

    //setter
    public void setUsername(String username)
    public void setPassword(String password)

    //getter
    public String getUsername()
    {
        return username;
    }
    public String getPassword()
    {
        return password;
    }
}
```

Langkah 3: class Utama (ketikkan script berikut)

a. Membuat kerangka class Utama

```
1  import java.io.*;
2  class Utama
3  {
4      public static void main (String [] args) throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //instance of class
9
10         //menu
11
12         //input
13
14         //proses + output
15     }
16 }
```

Setelah anda membuat class Utama, simpan file tersebut dengan nama **Utama.java**. Di dalam class inilah, program anda akan dijalankan. Sebagai catatan, dalam pembuatan class di atas, saya sudah menambahkan class BufferedReader (line 6) yang berada pada package java.io.* (line 1) yang digunakan untuk menerima inputan user.

b. Membuat instance of class

Setelah anda membuat class Utama, langkah berikutnya yang anda lakukan adalah membuat sebuah objek yang bertipe class Login. Pembuatan variabel dengan bertipe kelas itulah yang dinamakan ***instance of class*** (untuk penjelasannya, dapat anda lihat pada modul 1). Misalkan, objek yang saya buat adalah **user1** dan **user2**, dimana **user1** menggunakan constructor Login tanpa parameter, sedangkan **user2** menggunakan constructor Login dengan parameter.

```
1  import java.io.*;
2  class Utama
3  {
4      public static void main (String [] args) throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //instance of class
9          Login user1 = new Login();
10         Login user2 = new Login("","");
11
12         //menu
13
14         //input
15
16         //proses + output
17     }
18 }
```

Coba perhatikan kembali pembuatan *instance of class*. Pada *line 9*, objek “user1” yang telah terbentuk akan mereferens ke constructor Login tanpa parameter. Sedangkan pada

line 10, kita mendeklarasikan objek “user2” bertipe class Login, dimana nilai yang kita berikan masih belum diketahui. Karena tipe data username dan password bertipe *String*, maka saya menggunakan tanda petik ganda (“ ”) untuk memberi nilai awal berupa kosong. Variabel **user 2** yang telah dibuat nantinya akan digunakan pada **case 3**.

c. Membuat menu dan perulangan menu

Menu digunakan untuk mempermudah user dalam melakukan transaksi, seperti halnya buku menu yang disajikan seorang pelayan di sebuah restoran. Dalam pembuatan menu, diperlukan tombol “next” dan “back” sehingga user dapat leluasa memposisikan diri pada transaksi yang ingin dia lakukan. Untuk itulah, diperlukan perulangan menu guna mengantisipasi hal itu. Gambar di bawah ini menunjukkan pembuatan menu dan perulangan menu.

```
1  import java.io.*;
2  class Utama
3  {
4      public static void main (String [] args) throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //instance of class
9          Login user1 = new Login();
10         Login user2 = new Login("","");
11
12         while(true)
13         {
14             //menu
15             System.out.println("================");
16             System.out.println("MENU LOGIN");
17             System.out.println("================");
18             System.out.println("1. Login admin");
19             System.out.println("2. Ubah password admin");
20             System.out.println("3. Buat user");
21             System.out.println("4. Lihat Data user");
22             System.out.println("5. Keluar");
23             System.out.println("================");
24
25             //input
26
27             //proses + output
28
29         }
30
31     }
32 }
```

Line 15-23 menunjukkan menu yang kita butuhkan dalam contoh soal di atas. Sedangkan proses perulangan menu, saya menggunakan **while** yang berada di luar menu (bagi anda yang tidak terbiasa menggunakan “while”,

anda juga bisa menggunakan “do…while” maupun “for” dalam perulangannya). Di dalam “while”, saya menggunakan kondisi bernilai “*true*”, dimana program tersebut akan mengulang menu tersebut berulang kali. Untuk keluar dari menu tersebut, akan saya bahas nanti pada langkah **poin (h)**.

d. Membuat inputan yang diisi user

Setelah menu dan perulangan menu selesai kita buat, maka kita membutuhkan inputan user untuk memilih menu tersebut. Berikut adalah contoh *script*-nya.

```
1  import java.io.*;
2  class Utama
3  {
4      public static void main (String [] args) throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //instance of class
9          Login user1 = new Login();
10         Login user2 = new Login("","");
11
12         while(true)
13         {
14             //menu
15             System.out.println("================");
16             System.out.println("MENU LOGIN");
17             System.out.println("================");
18             System.out.println("1. Login admin");
19             System.out.println("2. Ubah password admin");
20             System.out.println("3. Buat user");
21             System.out.println("4. Lihat Data user");
22             System.out.println("5. Keluar");
23             System.out.println("================");
24
25             //input
26             System.out.print("Masukan pilihan = ");
27             int pilih = Integer.parseInt (br.readLine());
28
29             System.out.println();
30
31             //proses + output
32
33         }
34     }
35 }
```

Sekedar tambahan, “System.out.println();” pada line 29 hanya digunakan untuk memberikan jarak antara proses dengan inputan dari user.

e. Mengecek inputan user

Inputan user yang nantinya akan diisi, akan menentukan pilihan yang dieksekusi (kayak teroris aja ya… 💻). Untuk itu, dibutuhkan, pengecekan inputan user dengan menu yang dipilih. Di sini, saya menggunakan **switch…case…** dikarenakan penggunaannya lebih mudah dalam mengecek sebuah menu.

```
25             //input
26             System.out.print("Masukan pilihan = ");
27             int pilih = Integer.parseInt (br.readLine());
28
29             //proses + output
30             switch(pilih)
31             {
32                 //jika pilih = 1
33                 case 1:
34                     //isi pilihan bernilai 1 ketika dijalankan
35                     break;
36
37                 //jika pilih = 2
38                 case 2:
39                     //isi pilihan bernilai 2 ketika dijalankan
40                     break;
41
42                 //jika pilih = 3
43                 case 3:
44                     //isi pilihan bernilai 3 ketika dijalankan
45                     break;
46
47                 //jika pilih = 4
48                 case 4:
49                     //isi pilihan bernilai 4 ketika dijalankan
50                     break;
51
52                 //jika nilai pilih yang dimasukkan bukan 1,2,3 atau 4, maka program akan keluar secara otomatis
53                 default:
54                     System.exit(0);
55             }
56
57         }
58
59     }
60 }
```

Sekedar tambahan, dalam setiap case jangan lupa menambahkan **break** yang bertugas untuk menghentikan proses pengecekan menu apabila salah satu case sudah terpenuhi dan telah dieksekusi. Penggunaan **default** ditujukan apabila pilihan 1,2,3 atau 4 tidak

sesuai dengan inputan user. Jika anda perhatikan baik-baik, “System.exit(0);” pada line 54 bertujuan untuk keluar dari menu dan mengakhiri program.

f. Mengisi Case 1 (Login Admin)

Ketika user memilih inputan menu no. 1, maka dilakukan beberapa proses sebagai berikut:

- Line 36-39: berisi permintaan inputan username dan password yang nantinya akan diisi oleh user
- Line 44-51: berisi pengecekan apakah user dan password yang diinputkan sesuai dengan isi data user dan password pada class Login. Jika hasilnya bernilai **true**, maka program akan mencetak tulisan ***** Login Sukses *****. Jika hasilnya bernilai **false**, maka program akan mencetak tulisan ***** Login Gagal *****

```
29          //proses + output
30          switch(pilih)
31          {
32              //jika pilih = 1
33              case 1:
34                  //isi pilihan bernilai 1 ketika dijalankan
35
36                  System.out.print("Masukan username = ");
37                  String my_user = br.readLine();
38                  System.out.print("Masukan password = ");
39                  String my_password = br.readLine();
40
41                  System.out.println();
42
43                  //cek apakah user dan password yang diinputkan sesuai dengan isi data user dan password pada class Login
44                  if (my_user.equals(user1.getUsername()) && my_password.equals(user1.getPassword()))
45                  {
46                      System.out.println("*** Login Sukses ***");
47                  }
48                  else
49                  {
50                      System.out.println("*** Login Gagal ***");
51                  }
52                  break;
```

g. Mengisi Case 2 (Ubah Password Admin)

Ketika user memilih inputan menu no. 2, maka dilakukan beberapa proses sebagai berikut:

- Line 60-63: berisi permintaan inputan password lama dan password baru yang nantinya akan diisi oleh user
- Line 67-75: berisi pengecekan apakah password lama yang diinputkan sesuai dengan isi password yang sudah ada pada class Login. Jika hasilnya bernilai **true**, maka program akan merubah password lama dengan password baru dan kemudian mencetak tulisan ***** Password berhasil dirubah*****. Jika hasilnya bernilai **false**,

maka program akan mencetak tulisan ***** Anda salah memasukkan password lama *****

```
56                  //jika pilih = 2
57                  case 2:
58                      //isi pilihan bernilai 2 ketika dijalankan
59
60                      System.out.print("Masukan password lama = ");
61                      String old_password = br.readLine();
62                      System.out.print("Masukan password baru = ");
63                      String new_password = br.readLine();
64
65                      System.out.println();
66
67                      if (old_password.equals(user1.getPassword()))
68                      {
69                          user1.setPassword(new_password);
70                          System.out.println("***  Password berhasil dirubah ***");
71                      }
72                      else
73                      {
74                          System.out.println("*** Anda salah memasukkan password lama ***");
75                      }
76                      break;
```

5.5 Tugas

Dari latihan di atas:

1. **Mengisi Case 3 (Buat User)**

   Ketika user memilih inputan menu no. 3, maka dilakukan beberapa proses sebagai berikut:

   - permintaan inputan username dan password yang nantinya akan diisi oleh user
   - setelah anda membuat username dan password baru, program akan mencetak tulisan ***** User telah berhasil dibuat ***** (line 91).

```
78                  //jika pilih = 3
79                  case 3:
80                      //isi pilihan bernilai 3 ketika dijalankan
81
82                      System.out.print("Masukan username = ");
83                      String create_user = br.readLine();
84                      System.out.print("Masukan password = ");
85                      String create_password = br.readLine();
86
87                      user2 = new Login(create_user,create_password);
88
89                      System.out.println();
90
91                      System.out.println("*** User telah berhasil dibuat ***");
92                      break;
```

2. **Mengisi Case 4 (Lihat Data User)**

Untuk dapat mengetahui apakah username dan password yang anda buat berhasil masuk ke dalam class Login, anda dapat memilih **case 4** untuk melihat **data user**.

Ketika user memilih inputan menu no. 4, maka dilakukan beberapa proses sebagai berikut:

a. mencetak isi data username dan password yang sudah ada pada variabel **user 1**. Apabila anda melakukan perubahan password **admin** dari nilai *default*-nya adalah **12345** pada **case 2**, berarti data user yang ditampilkan adalah username **admin** dengan **password yang sudah anda rubah**.

b. mencetak isi data username dan password yang sudah anda buat pada **case 3**

BAB 6
Pewarisan (*Inheritance)*

6.1 Capaian Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan: 1 Mengetahui konsep pewariasan 2 Mengetahui implemtasi pewarisan dalam program

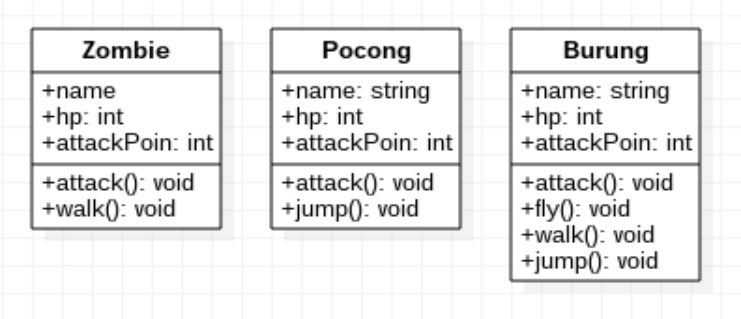
6.2 Indikator

1. Mampu menjelaskan konsep pewarisan 2. Mampu mengimplementasikan pewarisan di dalam program

6.3 Uraian Materi

Seperti yang sudah kita modul sebelumnya, sebuah class atau objek bisa saling berhubungan dengan class yang lain. Salah satu bentuk hubungannya adalah *inheritance*(pewarisan). Hubungan ini seperti hubungan keluarga antara orang tua dan anak. Sebuah class di Java, bisa memiliki satu atau lebih keturunan atau class anak. Class anak akan memiliki warisan properti dan method dari class ibu.

Misalkan dalam Game, kita akan membuat class-class musuh dengan perilaku yang berbeda.



Kode untuk masing-masing kelas seperti ini:

File: `Zombie.java`

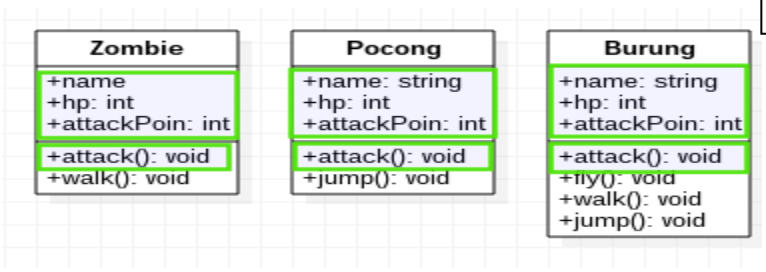
```
class Zombie {

    String name;

    int hp;

    int attackPoin;


    void attack(){
```

File: `Pocong.java`

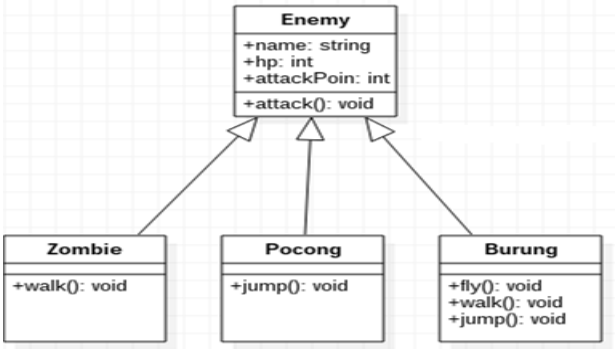
File: `Burung.java`

```
class Burung {

    String name;

    int hp;

    int attackPoin;


    void attack(){

        // ...

    }

    void walk(){

        //...

    }
```

Penulisan kode seperti di atas, tidak salah, namun tidak efektif, karena properti dan method yang sama ditulis berulang-ulang. Solusinya adalah menggunakan inheritance. Berikut gambaran member class yang sama:



Setelah menggunakan inheritance, maka akan menjadi seperti ini:



Inheritance bisa digambarkan dengan garis hubung Generalization.

Class Enemy adalah class induk yang memiliki anak Zombie, Pocong, dan Burung. Apapun properti yang ada di class induk, akan dimiliki juga oleh class anak. Lalu bagaimana bentuk kodenya dalam Java? Bentuk kodenya akan seperti ini:

File: Enemy.java

```
class Enemy {
    String name;
    int hp;
    int attackPoin;

    void attack(){
```

File: Zombie.java

```
class Zombie extends Enemy {
    void walk(){
      System.out.println("Zombie jalan-jalan");
    }
}
```

Pada class anak, kita menggunakan kata kunci extends untuk menyatakan kalau dia adalah class turunan dari Enemy.

File: Pocong.java

```
class Pocong extends Enemy {
    void jump(){
        System.out.println("loncat-loncat!");
    }
```

File: Burung.java

```
class Burung extends Enemy {
    void walk(){
        System.out.println("Burung berjalan");
    }
    void jump(){
        System.out.println("Burung loncat-loncat");
    }
```

Lalu, bila kita ingin membuat objek dari class-class tersebut,

```
Enemy monster = new Enemy();
Zombie zumbi = new Zombie();
Pocong hantuPocong = new Pocong();
Burung garuda = new Burung();
```

6.4.Latihan

Berikut ini merupakan contoh program Java inheritance. Dalam contoh ini anda dapat mengamati 2 class yaitu Kalkulasi dan Kalkulasiku. Dengan menggunakan kata kunci extends, Kalkulasiku mewarisi penambahan dan pengurangan methods dari class Kalkulasi. Buat projek baru dengan nama Kalkulasiku.java dan silahkan buat program seperti dibawah ini, kemudian kompile dan jalankan.

```
class Kalkulasi{
    int z;

    public void penambahan(int x, int y){
        z = x + y;
        System.out.println("Hasil Penambahan : " + z);
    }

    public void pengurangan(int x, int y){
        z = x - y;
        System.out.println("Hasil Pengurangan : " + z);
    }
}

public class Kalkulasiku extends Kalkulasi{
    public void perkalian(int x, int y){
        z = x * y;
        System.out.println("Hasil Perkalian : " + z);
    }

    public static void main(String args[]){
        int a = 10;
        int b = 5;
        Kalkulasiku tes = new Kalkulasiku();
        tes.penambahan(a, b);
        tes.pengurangan(a, b);
        tes.perkalian(a, b);
    }
}
```

a. Simpan program tersebut dengan nama Kalkulasiku.java, kemudian lakukan kompilasi.
b. Hasil luaran program adalah

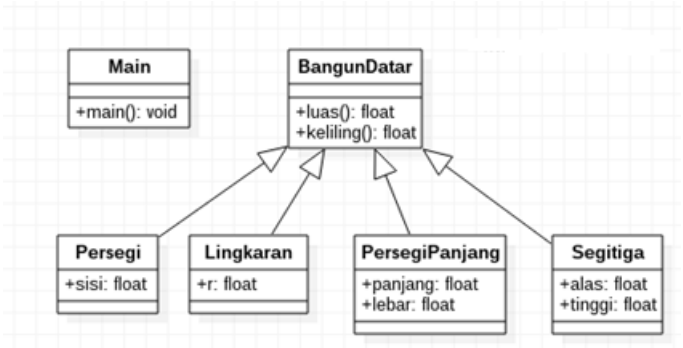
```
Hasil Penambahan : 15

Hasil Pengurangan : 5

Hasil Perkalian : 50
```

c. Berikan penjelasan mengenai program tersebut

6.5. Tugas

1. Buatlah program lengkap yang memanfaatkan class Enemy beserta turunannya.

2. Diketahui class diagram sebagai berikut. Buatlah program yang berfungsi untuk menghitung luas dan keliling



MODUL 7
Polimorfisme (Polymorphism)

7.1 Capaian Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan: 1 Mengetahui konsep polymorphism dan overriding method 2 Mengetahui implemtasi polymorphism dan overriding method

7.2 Indikator

1. Mampu menjelaskan konsep polymorphism dan overriding method 2. Mampu mengimplementasikan polymorphism dan overriding method di dalam program

7.3 Uraian Materi

Saya mempunyai sebuah kata "**bisa**" dan kata tersebut akan saya gunakan dalam kalimat sebagai berikut:

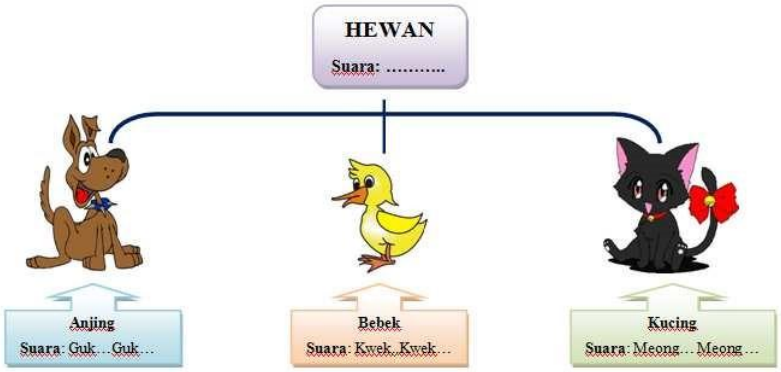
a. Praktikan **bisa** memahami konsep Polymorphism dalam Praktikum Pemrograman Berorientasi Objek
b. Ular itu mempunyai **bisa** yang sangat berbahaya

Dari 2 kalimat berikut, dapatkah anda mengetahui arti dari kata "bisa" dalam kalimat yang saya berikan tadi? Apakah arti dari kata "**bisa**" antar satu kalimat dengan kalimat yang lain memiliki arti yang sama?

Kata "**bisa**" pada kalimat pertama memiliki arti **dapat atau mampu**. Sedangkan kata "**bisa**" pada kalimat kedua memiliki arti **racun**. Dalam kalimat di atas, kata yang sama memiliki 2 makna yang berbeda. Itulah salah satu konsep Polymorphism. Untuk lebih jelas mengenai konsep polymorhism, saya akan menjelaskannya pada sub bab berikutnya.

A. Polymorphism

Polymorhism merupakan konsep OOP dimana variable/method dari sebuah kelas, dipanggil ulang pada kelas turunannya dengan perilaku yang berbeda-beda antar tiap kelas. Perhatikan contoh gambar berikut ini:



Setiap hewan memiliki suara. Namun suara pada masing-masing hewan berbeda-beda. Perilaku yang berbeda-beda itulah yang menjadi ciri khas polymorphism. Penggunaan teknik polymorphism dapat diketahui melalui pembuatan *instance-of-class* pada class Utama. Seperti halnya gambar di atas, maka pendeklarasian objek yang dibuat (di-instanskan) berasal dari class induknya, yakni class Hewan. Sehingga objek tersebut berbentuk *array-of-class*.

Penerapan konsep polymorphism dapat dilihat di soal latihan.

Kriteria penggunaan Polymorphism

Dalam penggunaan polymorphism, ada 2 kriteria yang harus dipenuhi, antara lain:

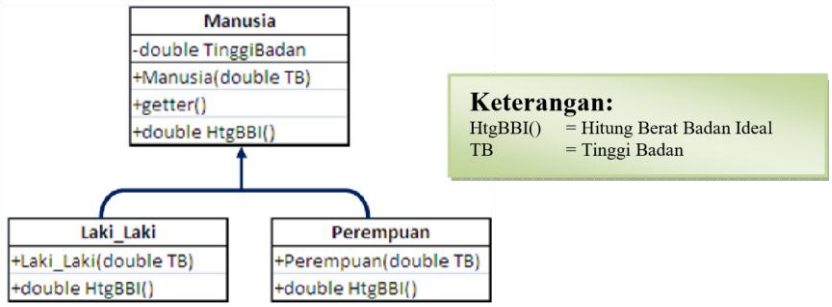
Method dari kelas turunan yang akan dieksekusi, dipanggil oleh objek dari kelas induk
Nama method yang digunakan pada kelas induk harus ditulis ulang (*overriding method*) pada kelas turunannya, dengan asumsi nama dan tipe data method harus sama

B. Overriding Method

Overriding method adalah proses pendeklarasian ulang nama method pada kelas utama kepada kelas turunannnya, Dalam pembuatan overriding method, nama dan tipe data method harus sama dengan kelas induknya guna pembuatan polymorphism. Untuk lebih jelasnya, akan saya bahas pada soal latihan.

7.4 Soal Latihan

Perhatikan gambar di bawah ini!

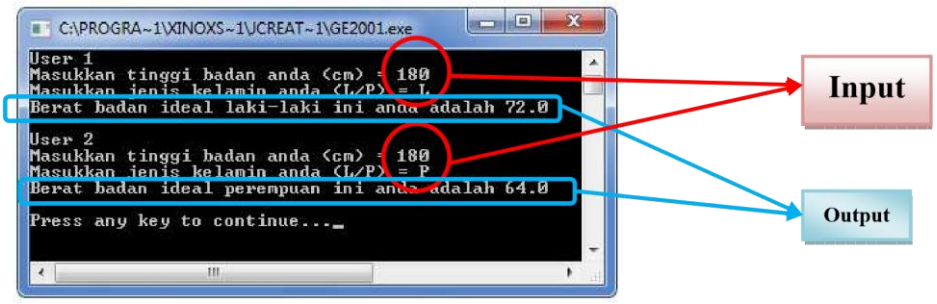


Buatlah class untuk menghitung Berat Badan Ideal sesuai dengan rancangan gambar di atas! Rumus hitung berat badan ideal adalah sebagai berikut:

Laki-Laki = (tinggi badan(cm)-100) kg x 90%

Untuk Perempuan = (tinggi badan(cm)-100) kg x 80%

Tambahkan pula class Utama yang digunakan untuk memanggil class Mahasiswa. Ketika class Utama dijalankan, hasilnya akan tampak seperti di bawah ini:



Jawabannya adalah…

Setelah anda membaca soal tersebut dengan baik dan seksama, langkah pertama yang harus anda lakukan adalah mengidentifikasi class mana yang harus dibuat terlebih dahulu.

Langkah 1: class Manusia (ketikkan script berikut)

a. Membuat kerangka class Manusia

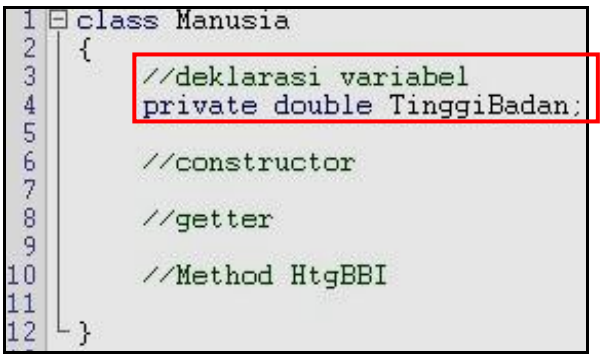
```
1  class Manusia
2  {
3      //deklarasi variabel
4
5      //constructor
6
7      //getter
8
9      //Method HtgBBI
10
11 }
```

Setelah anda membuat class Manusia, simpan file tersebut dengan nama **Manusia.java**. Di dalam class Manusia, saya juga menyediakan tempat untuk mendeklarasikan variabel, constructor, getter dan method HtgBBI() untuk menghitung berat badan ideal.

b. Mendeklarasi variabel yang dibutuhkan

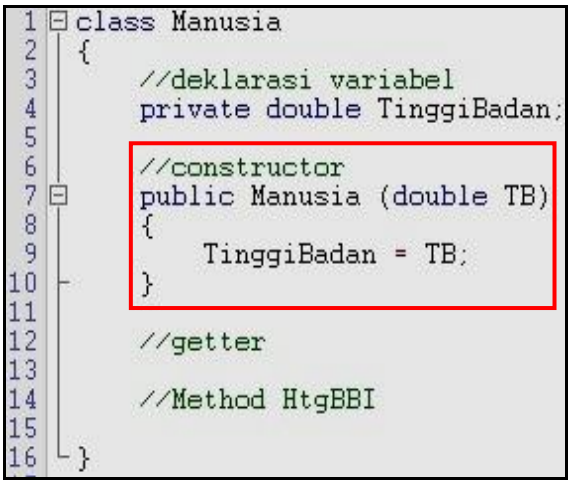
Setelah kita membuat kerangka class, maka diperlukan pendeklarasian variabel yang nantinya digunakan sebagai tempat menyimpan data yang bersifat sementara (*temporary*). Gambar di bawah ini menunjukkan

pendeklarasian variabel.

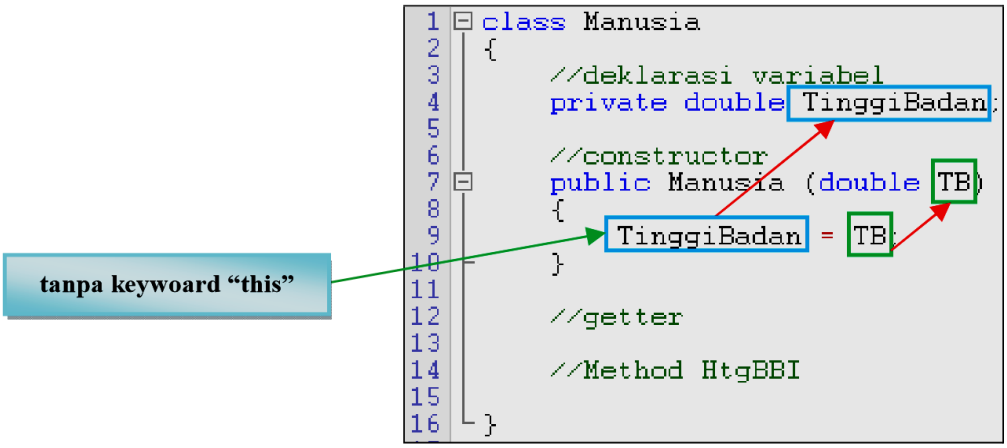


c. Mendeklarasi constructor

Setelah membuat variabel yang dibutuhkan pada class Manusia, langkah selanjutnya anda membuat constructor Manusia. Constructor ini nantinya akan digunakan dalam class Laki_Laki dan class Perempuan. Gambar di bawah ini menunjukkan deklarasi constructor.



Coba perhatikan kembali script yang telah anda buat seperti gambar di atas. Karena terdapat perbedaan deklarasi nama variabel pada class Manusia dan deklarasi variabel pada constructor Manusia, maka keyword ***this*** boleh digunakan atau tidak (*optional*). Untuk lebih jelas mengenai keyword ***this***, anda dapat melihat modul 1.



d. Membuat method getter

Setelah membuat constructor, anda tinggal membuat method getter untuk mengambil nilai dari masing-masing variabel. Hal ini bertujuan untuk mengambil nilai dari variabel pada class Manusia yang nantinya akan kita gunakan ke dalam class Laki_Laki maupun class Perempuan. Perlu diketahui pula, bahwa dalam pembuatan variabel pada parameter di method getter, harus sesuai dengan tipe data pada variabel yang dideklarasikan di kelas. Gambar di bawah ini menunjukkan deklarasi getter.

```
class Manusia
{
    //deklarasi variabel
    private double TinggiBadan;

    //constructor
    public Manusia (double TB)
    {
        TinggiBadan = TB;
    }

    //getter
    public double getTB()
    {
        return TinggiBadan;
    }

    //Method HtgBBI

}
```

e. Membuat method HtgBBI()

Setelah membuat method getter(), anda tinggal membuat method HtgBBI() yang bertugas untuk menghitung berat badan ideal berdasarkan tinggi badan. Karena di dalam class Manusia, belum terdapat rumus (hanya mendeklarasikan saja) dan meminta nilai balik (return), maka diberikan nilai default 0.0. Gambar di bawah ini menunjukkan deklarasi method HtgBBI().

```
class Manusia
{
    //deklarasi variabel
    private double TinggiBadan;

    //constructor
    public Manusia (double TB)
    {
        TinggiBadan = TB;
    }

    //getter
    public double getTB()
    {
        return TinggiBadan;
    }

    //Method HtgBBI
    public double HtgBBI()
    {
        return 0.0;
    }
}
```

Langkah 2: class Laki_Laki (ketikkan script berikut)

```
class Laki_Laki extends Manusia
{
    //constructor
    public Laki_Laki (double TB)
    {
        super(TB);
    }

    //Method HtgBBI() merupakan method overidding dari superclass-nya
    public double HtgBBI()
    {
        return (super.getTB()-100)*0.9;
    }
}
```

Setelah anda membuat class Laki_Laki, simpan file tersebut dengan nama **Laki_Laki.java**. Di dalam class Laki_Laki, terdapat penggunaan **extends** yang menunjukkan bahwa class Laki_Laki merupakan turunan dari class Manusia.

Coba perhatikan kembali script yang telah anda buat seperti gambar di atas. Pada constructor Laki_Laki, terdapat keyword "***super***". Keyword ini akan memanggil constructor Manusia (sesuai isi parameter) yang merupakan class induk. Sedangkan pada method HtgBBI() dilakukan pendeklarasian kembali (overriding method) sesuai dengan kelas induknya, dimana method HtgBBI() diberi rumus untuk menghitung berat badan ideal laki-laki.

Langkah 3: class Perempuan (ketikkan script berikut)

```
1  class Perempuan extends Manusia
2  {
3      //constructor
4      public Perempuan (double TB)
5      {
6          super(TB);
7      }
8
9      //Method HtgBBI() merupakan method overidding dari superclass-nya
10     public double HtgBBI()
11     {
12         return (super.getTB()-100)*0.8;
13     }
14 }
```

Setelah anda membuat class Perempuan, simpan file tersebut dengan nama **Perempuan.java**. Di dalam class Perempuan, terdapat penggunaan **extends** yang menunjukkan bahwa class Perempuan merupakan turunan dari class Manusia.

Seperti halnya dengan class Laki_Laki, class Perempuan memiliki constructor dan method HtgBBI() yang sama. Perbedaannya terletak pada nama class, nama constructor, dan isi rumus method HtgBBI().

Langkah 4: class Utama (ketikkan script berikut)

a. Membuat kerangka class Utama

```
1  import java.io.*;
2  class Utama
3  {
4      public static void main(String[] args)  throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //instance of class
9
10         //deklarasi variabel
11
12         //input
13
14         //proses + output
15
16     }
17 }
```

Setelah anda membuat class Utama, simpan file tersebut dengan nama **Utama.java**. Di dalam class inilah, program anda akan dijalankan. Sebagai catatan, dalam pembuatan class di atas, saya sudah menambahkan class BufferedReader (line 6) yang berada pada package java.io.* (line 1) yang digunakan untuk menerima inputan user.

b. Membuat *instance of class*

Setelah anda membuat class Utama, langkah berikutnya yang anda lakukan adalah membuat sebuah objek yang bertipe kelas induknya, yaitu class Manusia. Di sinilah

konsep polymorphism digunakan. Karena class Manusia memiliki 2 kelas turunan, maka objek yang kita buat berbentuk *array-of-class*. Untuk itu, diperlukan panjang elemen array guna menampung data, baik yang terdapat pada class Laki_Laki maupun class Perempuan (dimisalkan panjang elemen adalah 2). Perlu diketahui pula, bahwa panjang elemen array yang diberi nilai 2, bukan berasal dari 2 kelas turunan pada kelas induknya, melainkan karena inputan yang ingin dilakukan adalah sebanyak 2 kali. Data bisa diperoleh dari 2 orang laki-laki, 2 orang perempuan, atau 1 orang laki-laki dan 1 orang perempuan.

```
1  import java.io.*;
2  class Utama
3  {
4      public static void main(String[] args)  throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //instance of class
9          Manusia[] m = new Manusia[2];
10
11         //deklarasi variabel
12
13         //input
14
15         //proses + output
16
17     }
18 }
```

c. Mendeklarasikan variable dan perulangan inputan user

Langkah berikutnya adalah mendeklarasikan variabel guna membantu dalam menghitung index array pada objek "m". Selain itu, diperlukan perulangan dalam mengisi inputan user.

```
1  import java.io.*;
2  class Utama
3  {
4      public static void main(String[] args)  throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //instance of class
9          Manusia[] m = new Manusia[2];
10
11         //deklarasi variabel
12         int x=0;
13
14         do
15         {
16             //input
17
18             //proses + output
19
20             x++;
21         }while (x<2);
22
23     }
24 }
```

Pada line 20, yang berisi "x++;", terdapat proses increment untuk menambah isi+1pada variable x

d. Membuat inputan yang diisi user

Selanjutnya kita membutuhkan inputan user untuk mengisi data tersebut.

Berikut adalah contoh *script*-nya.

```
1  import java.io.*;
2  class Utama
3  {
4      public static void main(String[] args)  throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //instance of class
9          Manusia[] m = new Manusia[2];
10
11         //deklarasi variabel
12         int x=0;
13
14         do
15         {
16             //input
17             System.out.println("User "+(x+1));
18             System.out.print("Masukkan tinggi badan anda (cm) = ");
19             double t = Double.parseDouble(br.readLine());
20             System.out.print("Masukkan jenis kelamin anda (L/P) = ");
21             String jk=br.readLine();
22
23             //proses + output
24
25             x++;
26         }while (x<2);
27
28     }
29 }
```

e. Mengecek inputan user dan mencetak penghitungan berat badan ideal Setelah membuat inputan user, maka dilakukan pengecekan apakah jenis kelamin yang dipilih adalah laki-laki atau perempuan. Hal ini akan menentukan kelas turunan mana yang akan diakses.

- Line 24: penggunaan method "**equals()"** untuk mengecek inputan bertipe String dan String. Sedangkan method "**toUpperCase()**" digunakan untuk menconvert semua inputan user (baik ditulis dalam huruf keci atau huruf besar) menjadi huruf besar semua.
- Line 26 dan line 32: menunjukkan penggunaan objek pada kelas turunannya.

```
1  import java.io.*;
2  class Utama
3  {
4      public static void main(String[] args)  throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //instance of class
9          Manusia[] m = new Manusia[2];
10
11         //deklarasi variabel
12         int x=0;
13
14         do
15         {
16             //input
17             System.out.println("User "+(x+1));
18             System.out.print("Masukkan tinggi badan anda (cm) = ");
19             double t = Double.parseDouble(br.readLine());
20             System.out.print("Masukkan jenis kelamin anda (L/P) = ");
21             String jk=br.readLine();
22
23             //proses + output
24             if (jk.toUpperCase().equals("L"))
25             {
26                 m[x]=new Laki_Laki(t);
27                 System.out.println("Berat badan ideal laki-laki ini anda adalah "+m[x].HtgBBI());
28                 System.out.println();
29             }
30             else
31             {
32                 m[x]=new Perempuan(t);
33                 System.out.println("Berat badan ideal perempuan ini anda adalah "+m[x].HtgBBI());
34                 System.out.println();
35             }
36
37             x++;
38         }while (x<2);
39
40     }
41 }
```

7.5 Tugas

1. Perhatikan program di bawah ini

```java
// Nama file : Polimorphism.java
// Contoh penerapan konsep polimorphism

public class Polimorphism {

   public static void main(String[ ] args) {

      cetakObyek(new Balok());
      cetakObyek(new PersegiPanjang());
      cetakObyek(new BangunDatar());
      cetakObyek(new Object());
   }

   public static void cetakObyek(Object obyek) {
      System.out.println(obyek);
   }

} // Akhir kelas Polimorphism

class Balok extends PersegiPanjang {
   public String toString() {
      return "Mempunyai sisi panjang, lebar dan tinggi";
   }
}

class PersegiPanjang extends BangunDatar {
   public String toString() {
      return "Mempunyai sisi panjang dan lebar";
   }
}

class BangunDatar extends Object {
   public String toString() {
      return "Mempunyai berbagai bentuk";
   }
}
```

   a. Berikan penjelasan tentang program tersebut.
   b. Gambarkan skema yang mungkin dibuat
   c. Ketikkan program dan lakukan kompilasi, apakah terjadi kesalahan syntax? Jika masih ada yang salah, silakan dibenarkan
   d. Apa luarannya

MODUL 8
Kelas Abstrak (Abstract Class)

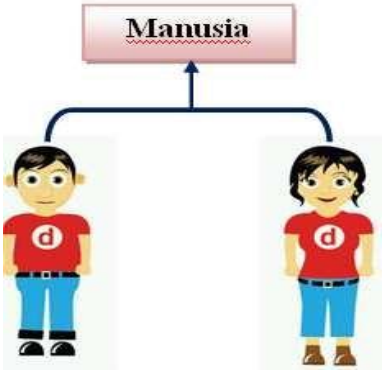
8.1 Capaian Pembelajaran

Mahasiswa dapat mengenal dan memahami konsep abstract class, abstract method dan keyword "*final*" serta penerapannya dalam konsep OOP

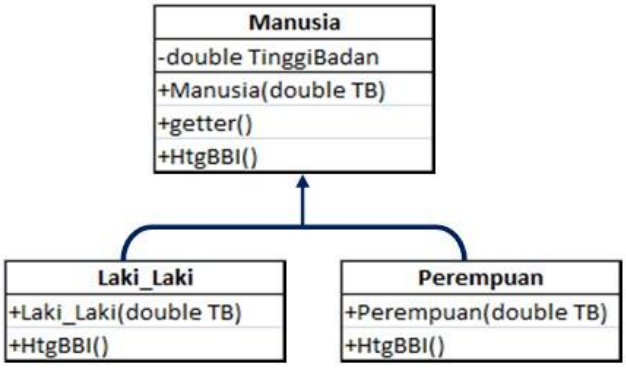
8.2 Indikator

Mampu menjelaskan dan mengimplementasika konsep abstract class, abstract method dan keyword "*final*" serta penerapannya dalam konsep OOP

8.3 Uraian Materi



Anda pasti masih ingat contoh program yang menghitung berat badan ideal berdasarkan tinggi badan. Di dalam contoh tersebut, terdapat class Manusia, class Laki_Laki, dan class Perempuan seperti contoh bagan di bawah ini.



Dalam abstract class, class tertinggi merupakan kelas Manusia. Apabila dalam kelas abstract terdapat **abstract method**, maka method tersebut hanya mendeskripsi method tanpa isi methodnya. Untuk lebih jelas, simak konsep abstract method di bawah ini.

A Abstract Class

Abstract Class merupakan kelas yang berada pada posisi tertinggi dalam sebuah hierarki kelas. Sesuai dengan namanya, abstract class dapat didefinisikan pada class itu sendiri. Berikut adalah cara mendeklarasikan abstract class:

```
abstract class Nama_Kelas
{
    //isi abstract class
}
```

Contoh:

```
abstract class Manusia
{
    //isi abstract class
}
```

Di dalam abstract class dapat juga diberi **abstract method** (optional). Penggunaan abstract method tidak diperlukan statement dalam method tersebut.

Berikut adalah cara mendeklarasikan abstract method pada abstract class:

```
abstract class Nama_Kelas
{
    //untuk sub program berjenis prosedur
    [access_modifier] abstract void [nama_method]();

    //untuk sub program berjenis function
    [access_modifier] abstract [tipe_data] [nama_method]();
}
```

Contoh:

```
abstract class Manusia
{
    //untuk sub program berjenis prosedur
    public abstract void cetak();

    //untuk sub program berjenis function
    public abstract double HtgBBI();
}
```

Catatan:

Apabila dalam abstract class terdapat abstract method dan kelas tersebut diturunkan ke kelas turunannya, maka method tersebut harus dideklarasikan ulang (overriding method) dengan diberi statement pada isi methodnya. Untuk lebih jelas penggunaan overriding method, dapat dilihat pada soal latihan

Apabila class tersebut merupakan abstract class, maka class tersebut bisa terdapat abstract method atau tidak (optional). Sedangkan apabila kelas tersebut, terdapat abstract method, maka kelas tersebut **wajib** berbentuk abstract class

B. Keyword "*final*"

Keyword "*final*" digunakan untuk mencegah suatu class diturunkan atau suatu method dilakukan pendeklarasian ulang (overriding method). Berikut adalah cara mendeklarasikan *final* dalam class:

```
final class Nama_Kelas
{
    //isi class
}
```

Contoh:

```
final class Manusia
{
    //isi class
}
```

Sedangkan cara mendeklarasikan "final" pada method adalah sebagai berikut:

```
final class Nama_Kelas
{
    //sub program berjenis prosedur
    [access_modifier] final void [nama_method]()
    {
        //isi method
    }

    //sub program berjenis function
    [access_modifier] final [tipe_data] [nama_method]()
    {
        //isi method
        return [isi_nilai];
    }
}
```

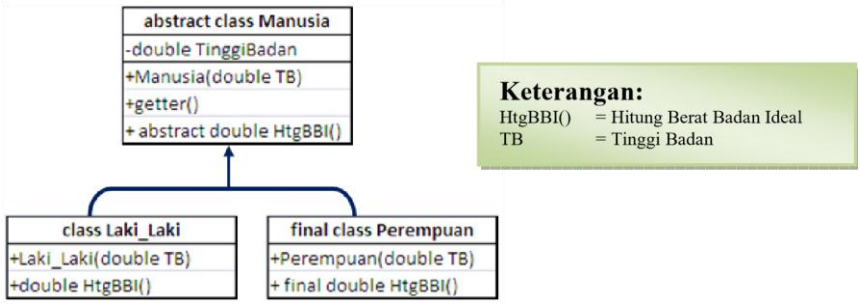
Contoh

```
final class Manusia
{
    //untuk sub program berjenis prosedur
    public final void cetak()
    {
        //isi method
    }

    //untuk sub program berjenis function
    public final double HtgBBI()
    {
        //isi method
        return 0;
    }
}
```

Untuk lebih detail dalam penggunaan keyword "*final*", akan dibahas lebih lanjut melalui soal latihan

8.4 Latihan

Perhatikan gambar di bawah ini!

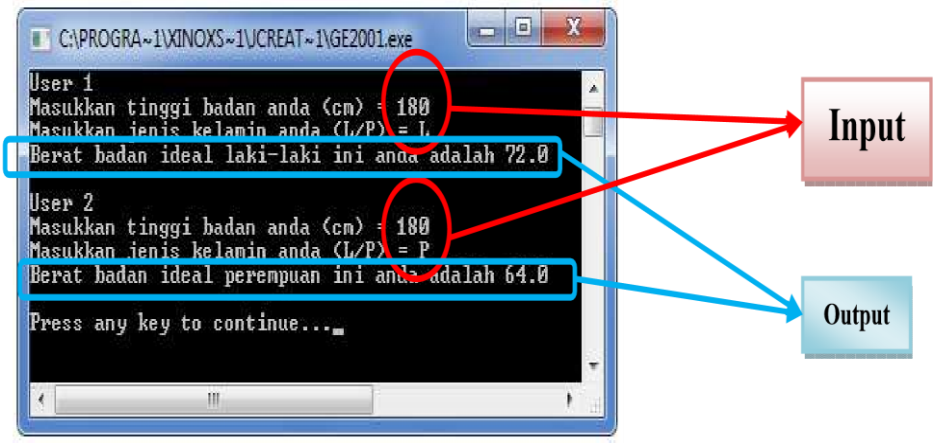


Buatlah class untuk menghitung Berat Badan Ideal sesuai dengan rancangan gambar di atas! Rumus hitung berat badan ideal adalah sebagai berikut:

Laki-Laki = (tinggi badan(cm)-100) kg x 90%

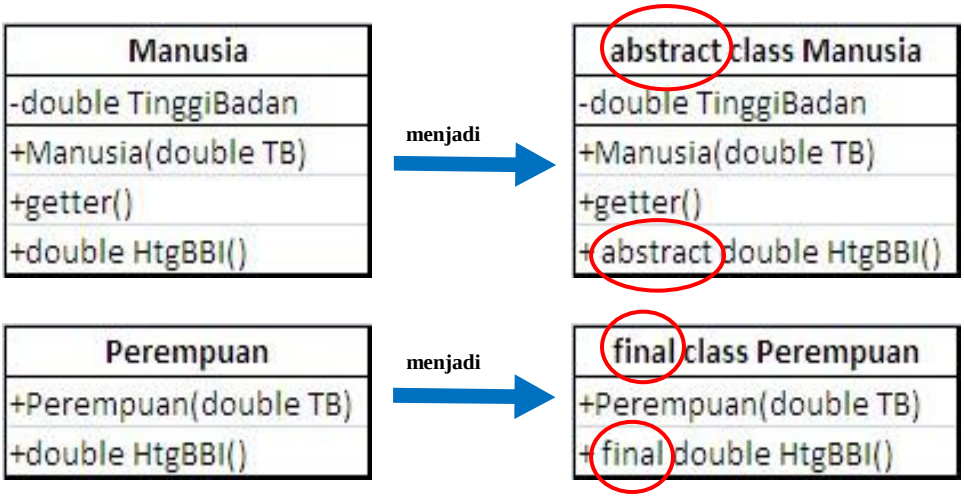
Untuk Perempuan = (tinggi badan(cm)-100) kg x 80%

Tambahkan pula class Utama yang digunakan untuk memanggil class Mahasiswa. Ketika class Utama dijalankan, hasilnya akan tampak seperti di bawah ini:



Jawabannya adalah…

Ketika anda perhatikan baik-baik bagan soal latihan modul ini dengan soal latihan modul sebelumnya, maka anda akan menemukan perbedaan sebagai berikut:



Langkah 1: class Manusia (ketikkan script berikut)

a. Membuat kerangka class Manusia

```
abstract class Manusia
{
    //deklarasi variabel

    //constructor

    //getter

    //Method HtgBBI

}
```

Setelah anda membuat class Manusia, simpan file tersebut dengan nama **Manusia.java**. Di dalam class Manusia, saya juga menyediakan tempat untuk mendeklarasikan variabel, constructor, getter dan method HtgBBI() untuk menghitung berat badan ideal.

b. Mendeklarasi variabel yang dibutuhkan

Setelah kita membuat kerangka class, maka diperlukan pendeklarasian variabel yang nantinya digunakan sebagai tempat menyimpan data yang bersifat sementara (*temporary*). Gambar di bawah ini menunjukkan pendeklarasian variabel.

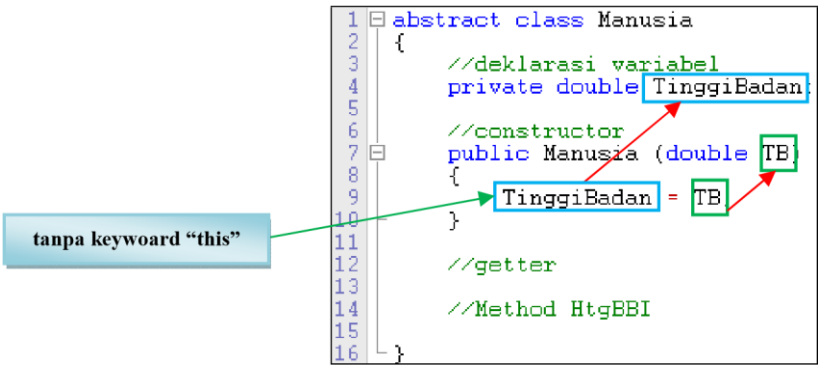
```
abstract class Manusia
{
    //deklarasi variabel
    private double TinggiBadan;

    //constructor

    //getter

    //Method HtgBBI

}
```

c. Mendeklarasi constructor

Setelah membuat variabel yang dibutuhkan pada class Manusia, langkah selanjutnya anda membuat constructor Manusia. Constructor ini nantinya akan digunakan dalam class Laki_Laki dan class Perempuan. Gambar di bawah ini menunjukkan deklarasi constructor.

```
1  abstract class Manusia
2  {
3      //deklarasi variabel
4      private double TinggiBadan;
5
6      //constructor
7      public Manusia (double TB)
8      {
9          TinggiBadan = TB;
10     }
11
12     //getter
13
14     //Method HtgBBI
15
16 }
```

Coba perhatikan kembali script yang telah anda buat seperti gambar di atas. Karena terdapat perbedaan deklarasi nama variabel pada class Manusia dan deklarasi variabel pada constructor Manusia, maka keyword ***this*** boleh digunakan atau tidak (*optional*). Untuk lebih jelas mengenai keyword ***this***, anda dapat melihat modul 1.



d. Membuat method getter

Setelah membuat constructor, anda tinggal membuat method getter untuk mengambil nilai dari masing-masing variabel. Hal ini bertujuan untuk mengambil nilai dari variabel pada class Manusia yang nantinya akan kita gunakan ke dalam class Laki_Laki maupun class Perempuan. Perlu diketahui pula, bahwa dalam pembuatan variabel pada parameter di method getter, harus sesuai dengan tipe data pada variabel yang dideklarasikan di kelas. Gambar di bawah ini menunjukkan deklarasi getter.

```
abstract class Manusia
{
    //deklarasi variabel
    private double TinggiBadan;

    //constructor
    public Manusia (double TB)
    {
        TinggiBadan = TB;
    }

    //getter
    public double getTB()
    {
        return TinggiBadan;
    }

    //Method HtgBBI

}
```

e. Membuat method HtgBBI()

Setelah membuat method getter(), anda tinggal membuat method HtgBBI() yang bertugas untuk menghitung berat badan ideal berdasarkan tinggi badan. Karena method HtgBBI() merupakan abstract method, maka method tersebut tidak terdapat isi method. Gambar di bawah ini menunjukkan deklarasi method HtgBBI().

```
abstract class Manusia
{
    //deklarasi variabel
    private double TinggiBadan;

    //constructor
    public Manusia (double TB)
    {
        TinggiBadan = TB;
    }

    //getter
    public double getTB()
    {
        return TinggiBadan;
    }

    //Method HtgBBI
    public abstract double HtgBBI();
}
```

Langkah 2: class Laki_Laki (ketikkan script berikut)

Karena class Laki_Laki pada soal latihan 4 = soal latihan 5, maka script ini tidak mengalai perubahan. Adapun scriptnya adalah sebagai berikut:

```
class Laki_Laki extends Manusia
{
    //constructor
    public Laki_Laki (double TB)
    {
        super(TB);
    }

    //Method HtgBBI() merupakan method overidding dari superclass-nya
    public double HtgBBI()
    {
        return (super.getTB()-100)*0.9;
    }
}
```

Setelah anda membuat class Laki_Laki, simpan file tersebut dengan nama **Laki_Laki.java**. Di dalam class Laki_Laki, terdapat penggunaan **extends** yang menunjukkan bahwa class Laki_Laki merupakan turunan dari class Manusia.

Coba perhatikan kembali script yang telah anda buat seperti gambar di atas. Pada constructor Laki_Laki, terdapat keyword "***super***". Keyword ini akan memanggil constructor Manusia (sesuai isi parameter) yang merupakan class induk. Sedangkan pada method HtgBBI() dilakukan pendeklarasian kembali (overriding method) sesuai dengan kelas induknya, dimana method HtgBBI() diberi rumus untuk menghitung berat badan ideal laki-laki.

Langkah 3: class Perempuan (ketikkan script berikut)

```
final class Perempuan extends Manusia
{
    //constructor
    public Perempuan (double TB)
    {
        super(TB);
    }

    //Method HtgBBI() merupakan method overidding dari superclass-nya
    public final double HtgBBI()
    {
        return (super.getTB()-100)*0.8;
    }
}
```

Setelah anda membuat class Perempuan, simpan file tersebut dengan nama **Perempuan.java**. Di dalam class Perempuan, terdapat penggunaan **extends** yang menunjukkan bahwa class Perempuan merupakan turunan dari class

Manusia.

Seperti halnya dengan class Laki_Laki, class Perempuan memiliki constructor dan method HtgBBI() yang sama. Perbedaannya terletak pada nama class, nama constructor, dan isi rumus method HtgBBI().

Keyword “final” pada *line 1* digunakan untuk mencegah pembuatan kelas baru dari kelas turunan perempuan. Sedangkan keyword “final” pada *line 10* digunakan untuk mencegah pendeklarasian ulang (overriding method) pada kelas turunannya.

Langkah 4: class Utama (ketikkan script berikut)

Adapun scriptnya adalah sebagai berikut:

```
import java.io.*;
class Utama
{
    public static void main(String[] args)  throws Exception
    {
        BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
        //instance of class
        Manusia[] m = new Manusia[2];

        //deklarasi variabel
        int x=0;

        do
        {
            //input
            System.out.println("User "+(x+1));
            System.out.print("Masukkan tinggi badan anda (cm) = ");
            double t = Double.parseDouble(br.readLine());
            System.out.print("Masukkan jenis kelamin anda (L/P) = ");
            String jk=br.readLine();

            //proses + output
            if (jk.toUpperCase().equals("L"))
            {
                m[x]=new Laki_Laki(t);
                System.out.println("Berat badan ideal laki-laki ini anda adalah "+m[x].HtgBBI());
                System.out.println();
            }
            else
            {
                m[x]=new Perempuan(t);
                System.out.println("Berat badan ideal perempuan ini anda adalah "+m[x].HtgBBI());
                System.out.println();
            }
            x++;
        }while (x<2);

    }
}
```

Untuk penjelasan script pada class Utama, dapat dilihat pada soal latihan modul 4. Perlu diketahui pula, bahwa ***instance of class*** **pada class Utama <u>tidak wajib</u> menggunakan array of Class (konsep Polymorphism), tapi anda bisa juga membuat objek berdasarkan kelas turunannya.**

8.5 Tugas

1. Buat program untuk mencetak ciri makhluk hidup (manusia, hewan, tumbuhan). Ciri ciri didefiniskan kedalam suatu method yang bertipe abstrak dalam sebuah abstract class. Kemudian buatlah class baru (manusia, hewan, dan tumbuhan) untuk meng-extends class tersebut.
2. Diberikan oleh dosen pengampu

BAB 9
Interface

9.1 Capaian Pembelajaran

Mahasiswa dapat mengenal dan memahami konsep interface, serta penerapan dalam interface dalam konsep OOP

9.2 Indikator

Mampu menjelaskan dan mengimplementasika konsep interface, serta penerapan dalam interface dalam konsep OOP

9.3 Uraian Materi

Ilustrasi 1

Ketika anda diberi tugas menerjemahkan sebuah buku berbahasa Inggris ke dalam bahasa Indonesia, namun ada beberapa kata dalam buku tersebut yang sangat asing bagi

anda. Apa yang akan anda lakukan untuk memecahkan masalah tersebut?

Ilustrasi 2

Ketika anda meminjam sebuah buku dan anda ingin mengetahui informasi apa saja yang ada pada buku tersebut, apa yang akan anda lakukan?

Meminjam kamus adalah salah satu alternatif jawaban dalam ilustasi 1. Mengapa? Karena di dalam kamus tersimpan kumpulan kata (atau biasa disebut dengan *vocabulary*) yang anda butuhkan untuk membantu anda dalam

mengartikan kata tersebut

Demikian juga untuk ilustrasi 2. Apabila jawaban anda adalah **melihat daftar isi**, berarti anda mengerti kegunaan daftar isi. Karena di dalam daftar isi, terdapat halaman dari tiap bab yang akan membantu anda dalam mempermudah mencari informasi yang anda butuhkan

Kamus dan daftar isi merupakan alat bantu berupa kumpulan informasi sebagai sarana pendukung dalam membantu anda untuk mengolah dan mengembangkan informasi yang anda peroleh. Seperti halnya ketika anda menggunakan Interface pada pemrograman berbasis

objek. Interface berisi sekumpulan konstanta/deklarasi method tanpa menyertakan/ menuliskan body methodnya. Method atau variabel yang terdapat pada kelas Inteface dapat digunakan lebih dari satu kelas dengan cara memanggil kelas interface tersebut.

Interface adalah kumpulan method yang hanya memuat deklarasi dan struktur method tanpa detail implementasinya. Cara mendeklarasikan *interface* adalah sebagai berikut:

```
interface Nama_Interface
{
    //deklarasi variabel dan/atau method
}
```

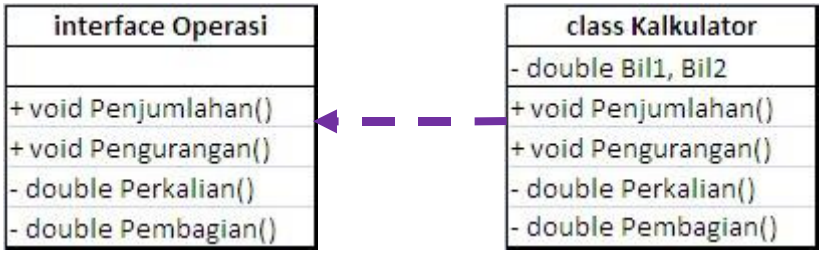
Contoh:

```
interface Operasi
{
    //deklarasi variabel dan/atau method
    public void Penjumlahan();
    public void Pengurangan();
    public double Perkalian();
    public double Pembagian();
}
```

Pada contoh di atas, method yang dideklarasikan pada interface Operasi tidak terdapat statement apapun, baik itu rumus atau hanya sebuah nilai balik di dalamnya. Hal ini dikarenakan interface hanyalah sebuah berisi kumpulan konstanta maupun method tanpa menyertakan/ menuliskan body methodnya.

Perlu diketahui pula, bahwa **an interface is not a class and classes can only implement interfaces** (sebuah interface bukanlah sebuah kelas dan kelas hanya bisa mengimplementasi interface). Sehingga **jangan anda menganggap bahwa interface adalah super class dimana memiliki kelas trurunan.**

Penggunaan (implementasi) *interface* dalam sebuah kelas dapat anda lihat melalui skema OOP di bawah ini:



Coba anda lihat anak panah yang berwarna ungu tersebut. Anak panah itu merupakan gambaran bahwa **class Kalkulator merupakan implementasi dari interfaces Operasi**, dimana method-method yang terdapat pada interface Operasi harus dideklarasi ulang (overriding method) pada kelas Kalkulator. *Interface* dilambangkan dengan anak panah dengan garis putus-putus, sedangkan *inheritance* dilambangkan dengan anak panah dengan garis lurus ( ).

Dalam pemrograman OOP, implementasi interfaces menggunakan keyword **implements**. Berikut adalah cara mengimplementasikan interface ke dalam pemrograman Java:

```
class Nama_Kelas implements Nama_Interface
{
    //isi kelas
}
```

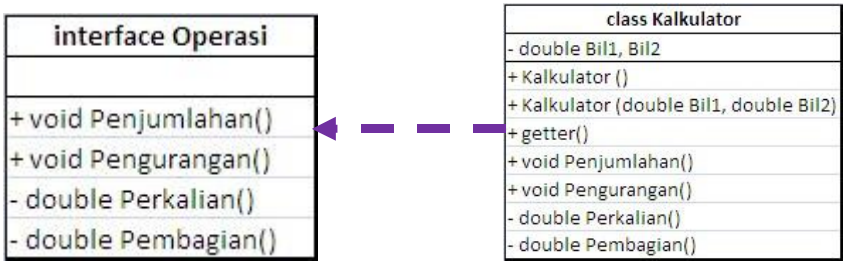
Contoh:

```
class Kalkulator implements Operasi
{
    public void Penjumlahan()
    {
        //isi method Penjumlahan()
    };

    public void Pengurangan()
    {
        //isi method Pengurangan()
    };

    public double Perkalian()
    {
        //isi method Perkalian()
        return 0;
    };

    public double Pembagian()
    {
        //isi method Pembagian()
        return 0;
    };
};
```

Agar anda lebih memahami bagaimana implementasi interface dalam sebuah kelas, simak contoh latihan di bawah ini.

9.4 Soal Latihan

Berdasarkan contoh di atas tentang kalkulator, buatlah program sesuai dengan skema di bawah ini:

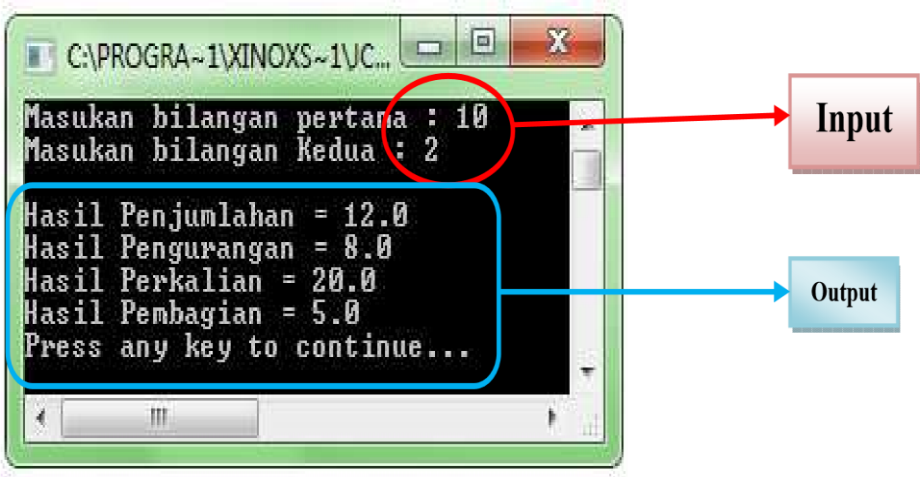


Adapun fungsi methodnya adalah sebagai berikut:

- Method **Penjumlahan()** digunakan untuk menjumlahkan dua buah bilangan, yakni Bil1 dan Bil2
- Method **Pengurangan()** digunakan untuk mengurangi dua buah bilangan, yakni Bil1 dan Bil2 Method **Perkalian()** digunakan untuk mengalikan dua buah bilangan, yakni Bil1 dan Bil2
- Method **Pembagian()** digunakan untuk membagi dua buah bilangan, yakni Bil1 dan Bil2

Tambahkan pula class Utama yang digunakan untuk memanggil class Kalkulator. Ketika class Utama dijalankan, hasilnya akan tampak seperti di bawah

ini:



Jawabannya adalah…

Setelah anda membaca soal tersebut dengan baik dan seksama, langkah pertama yang harus anda lakukan adalah mengidentifikasi class atau interface yang harus dibuat terlebih dahulu.

Langkah 1: interface Operator (ketikkan script berikut)

a. Membuat kerangka interface Operator

```
interface Operasi
{
    //deklarasi method
}
```

Setelah anda membuat interface Operasi, simpan file tersebut dengan nama **Operasi.java**. Di dalam interface Operasi, saya juga menyediakan tempat untuk mendeklarasikan method yang akan digunakan pada class Kalkulator.

b. Mendeklarasi method

```
interface Operasi
{
    //deklarasi method
    public void Penjumlahan();
    public void Pengurangan();
    public double Perkalian();
    public double Pembagian();
}
```

Dalam interface Operator, anda cukup mendeklarasikan method tanpa isi method (*body method*)

Langkah 2: class Kalkulator (ketikkan script berikut)

a. Membuat kerangka class Kalkulator

Setelah anda membuat class Kalkulator, simpan file tersebut dengan nama **Kalkulator.java**. Kelas Kalkulator merupakan hasil implementasi dari interface Operasi. Untuk itu, pada *line 1* terdapat keyword "implements". Khusus pada langkah ini, anda jangan merasa bingung apabila anda mendapat 1 error pada saat program di-*compile*.

```
1  class Kalkulator implements Operasi
2  {
3      //deklarasi variabel
4
5      //constructor
6
7      //getter
8
9      //implementasi method
10
11 };
```

Error yang berisi **Kalkulator is not abstract and does not override abstract method**

Pembagian() in Operasi menandakan bahwa method Penjumlahan(), Pengurangan(), Perkalian(), Pembagian() **harus dideklarasikan ulang (overriding method)** ke dalam class Kalkulator. Error ini akan terus ada sampai anda menyelesaikan poin (e).

b. Mendeklarasi variabel yang dibutuhkan

Setelah kita membuat kerangka class, maka diperlukan pendeklarasian variabel yang nantinya digunakan sebagai tempat menyimpan data yang bersifat sementara (*temporary*). Gambar di bawah ini menunjukkan

pendeklarasian variabel.

```
1  class Kalkulator implements Operasi
2  {
3      //deklarasi variabel
4      private double Bil1, Bil2;
5
6      //constructor
7
8      //getter
9
10     //implementasi method
11
12 };
```

c. Mendeklarasi constructor

Setelah membuat variabel yang dibutuhkan pada class Kalkulator, langkah selanjutnya anda membuat constructor Kalkulator. Constructor ini nantinya akan digunakan dalam class Utama. Gambar di bawah ini menunjukkan

deklarasi constructor.

```
class Kalkulator implements Operasi
{
    //deklarasi variabel
    private double Bil1, Bil2;

    //constructor
    Kalkulator ()
    {

    }

    Kalkulator(double Bil1, double Bil2)
    {
        this.Bil1=Bil1;
        this.Bil2=Bil2;
    }

    //getter

    //implementasi method

};
```

Pada gambar di atas, saya menggunakan Overloading Constructor (bagi yang lupa tentang Overloading Constructor, anda dapat melihat kembali pada modul 2 tentang Constructor. Constructor pertama digunakan untuk standard awal dalam melakukan *instance of class*. Sedangkan constructor kedua digunakan untuk mengeset data bilangan pertama dan bilangan kedua yang diperoleh dari kelas Utama.

d. Membuat method getter

Setelah membuat constructor, anda tinggal membuat method getter untuk mengambil nilai dari masing-masing variabel. Hal ini bertujuan untuk mengambil nilai dari variabel pada class Kalkulator yang nantinya akan kita kembalikan ke dalam class Utama. Perlu diketahui pula, bahwa dalam pembuatan variabel pada parameter di method getter, harus sesuai dengan tipe data pada variabel yang dideklarasikan di kelas. Gambar di bawah ini menunjukkan deklarasi getter.

```
class Kalkulator implements Operasi
{
    //deklarasi variabel
    private double Bil1, Bil2;

    //constructor
    Kalkulator ()
    {

    }

    Kalkulator(double Bil1, double Bil2)
    {
        this.Bil1=Bil1;
        this.Bil2=Bil2;
    }

    //getter
    public double getBil1()
    {
        return Bil1;
    };

    public double getBil2()
    {
        return Bil2;
    };

    //implementasi method

};
```

e. Implementasi method

Setelah membuat constructor, anda **wajib melakukan deklarasi ulang (overriding method)** ke dalam class Kalkulator seperti pada script di bawah

ini.

```
class Kalkulator implements Operasi
{
    //deklarasi variabel
    private double Bil1, Bil2;

    //constructor
    Kalkulator ()

    Kalkulator(double Bil1, double Bil2)

    //getter
    public double getBil1();

    public double getBil2();

    //implementasi method
    public void Penjumlahan()
    {
        System.out.println (Bil1+Bil2);
    };

    public void Pengurangan()
    {
        System.out.println (Bil1-Bil2);
    };

    public double Perkalian()
    {
        return Bil1*Bil2;
    };

    public double Pembagian()
    {
        return Bil1/Bil2;
    };
};
```

Perlu diketahui pula, bahwa method Penjumlahan() dan Pengurangan merupakan sub program berjenis prosedur. Sedangkan method Perkalian() dan Pembagian() merupakan sub program berjenis fungsi. Untuk itu, ada perbedaan cara memanggil method dalam kelas Utama.

Langkah 3: class Utama (ketikkan script berikut)

Berikut adalah script kelas utama kalkulator:

```
1  import java.io.*;
2  class Utama
3  {
4      public static void main (String [] args) throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //instance of class
9          Kalkulator k = new Kalkulator();
10
11         //input
12         System.out.print("Masukan bilangan pertama : ");
13         double a= Double.parseDouble(br.readLine());
14         System.out.print("Masukan bilangan Kedua : ");
15         double b= Double.parseDouble(br.readLine());
16
17         k= new Kalkulator (a,b);
18
19         System.out.println();
20
21         //output
22         System.out.print ("Hasil Penjumlahan = ");
23         k.Penjumlahan();
24
25         System.out.print ("Hasil Pengurangan = ");
26         k.Pengurangan();
27
28         System.out.println("Hasil Perkalian = "+k.Perkalian());
29
30         System.out.println("Hasil Pembagian = "+k.Pembagian());
31     }
32 }
```

Keterangan:

Line 9 = deklarasi instance of class, dimana variabel tersebut bertipe kelas Kalkulator, yang merupakan turunan dari kelas Operasi . Line 12-15 = inputan user, dimana bilangan 1 ditampung ke dalam variabel a dengan tipe data double. Sedangkan bilangan 2 ditampung ke dalam variabel b dengan tipe data double. Line 17 = mentransfer data pada variable a dan b ke dalam constructor Kalkulator. Line 19 = sebagai jarak antara isi input dan output ketika program dijalankan. Line 22-26 = cara memanggil method Penjumlahan dan Pengurangan yang merupakan sub program bertipe void. Karena di dalam isi void terdapat System.out.println, maka pemanggilan method dilakukan di luar kelas. Line 28-30 = cara memanggil method Penjumlahan dan Pengurangan yang merupakan sub program bertipe void. Karena di dalam isi function tidak terdapat System.out.println dan hanya mengembalikan return (nilai balik), maka pada class Utama, pemanggilan method dilakukan di dalam System.out.println().

9.5. Tugas

Diberikan oleh dosen pengampu

BAB 10
Pemrograman Berbasis GUI (1)

10.1 Capaian Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan:

1. Memahami cara penggunaan IDE Netbeans untuk membangun aplikasi berbasis GUI.
2. Memahami konsep pemrograman berbasis GUI.
3. Memahami fungsi dari komponen-komponen swing GUI pada Java.

10.2 Indikator

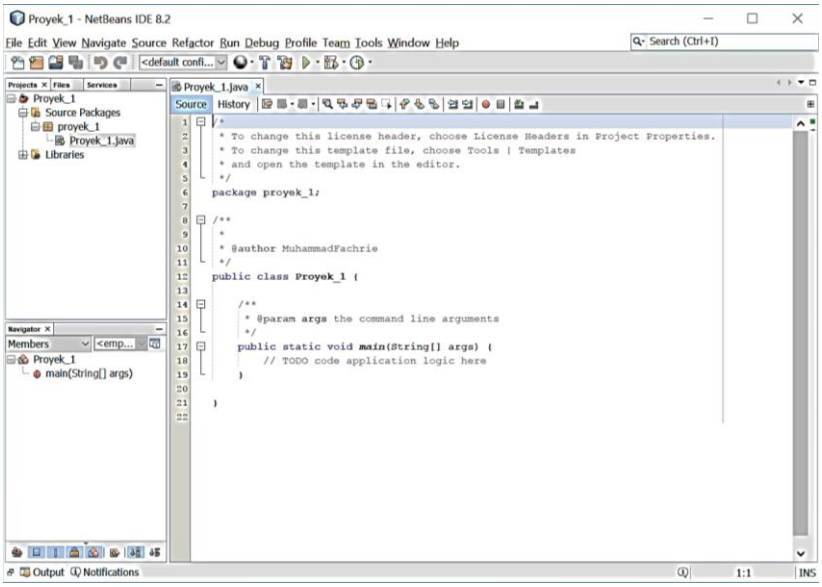
1. Mahasiswa mampu menggunakan IDE Netbeans untuk membangun aplikasi
2. Mahasiswa mampu menjelaskan konsep pemrograman berbasis GUI pada Java
3. Mahasiswa mampu menggunakan komponen-komponen swing GUI dalam membuat aplikasi

10.3 Uraian Materi

Pada bab ini, kita akan menggunakan IDE Netbeans untuk membantu mempermudah pembuatan program berbasi *Graphic User Interface* (GUI). Untuk itu, akan dibahas terlebih dahulu mengenai IDE Netbeans.

A. Pengenalan IDE Netbeans

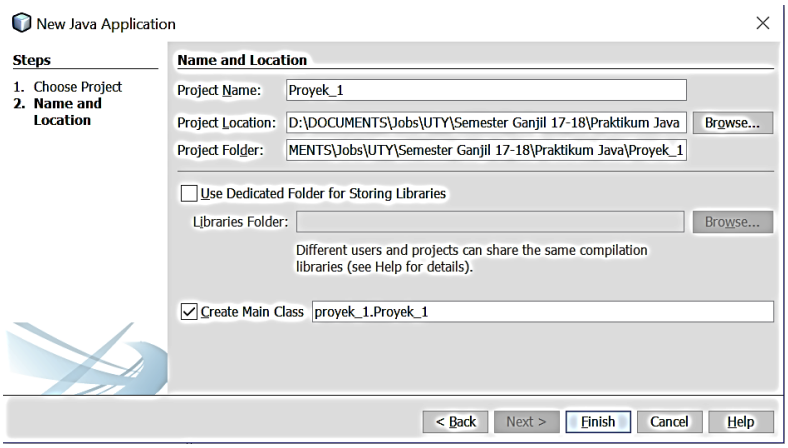
Netbeans adalah salah satu IDE yang popular yang sering digunakan untuk mengembangkan aplikasi berbasis Java. Meskipun demikian, Netbeans juga bisa digunakan untuk mengembangkan aplikasi berbasis bahasa pemrograman lain, seperti C++ atau PHP. Netbeans awalnya merupakan IDE yang dikembangkan oleh Sun Microsystems, namun saat ini telah diakuisisi oleh Oracle Corporation. Gambar di bawah ini menampilkan antarmuka dari IDE Netbeans.



Pemrograman di dalam Netbeans berbasis pada *project* yang disimpan dalam format *package*. Oleh sebab itu, setiap kali akan memulai pengembangan sebuah aplikasi, maka programmer diharuskan membuat *New Project* dengan nama tertentu, yang kemudian nama tersebut secara otomatis menjadi nama *file* dari kode program yang disimpan. Selain itu, pada baris paling atas dari kode program akan tampil tulisan `package <nama_proyek>`, misalnya `package proyek_1`, seperti yang terlihat pada gambar di atas.

B. Membuat *New Project*

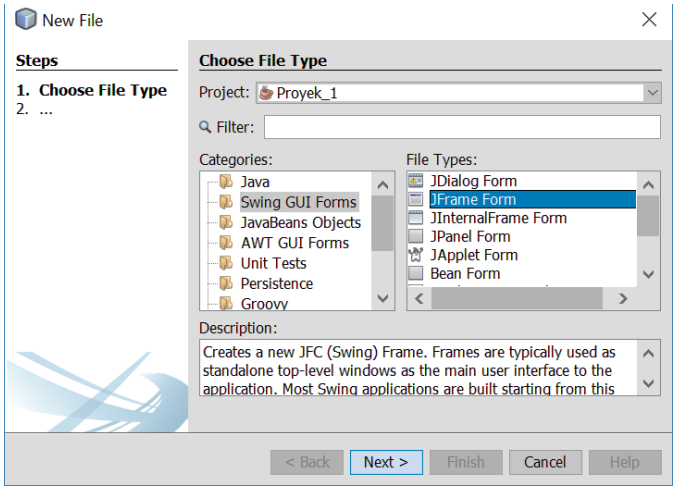
Untuk memulai membuat sebuah aplikasi berbasis Java menggunakan IDE Netbeans, maka pertama kali kita perlu membuat *New Project* melalui menu "File → New Project", kemudian pilih "Java" pada kolom *Categories* dan pilih "Java Application" pada kolom *Projects*. Setelah itu klik tombol *Next*, lalu ketikkan nama proyek yang akan dibuat pada bagian *Project Name*, misalnya "Proyek_1" dan tentukan direktori penyimpanan proyek pada bagian *Project Location*. Terakhir klik tombol *Finish*, maka akan tampil *file* kode program berkestensi *.java dari proyek yang kita buat seperti yang terlihat pada Gambar 1. Jika Anda perhatikan, nama file *.java tersebut akan sama dengan nama proyek yang kita buat tadi. *File* kode *.java tersebut disebut sebagai *main class*.



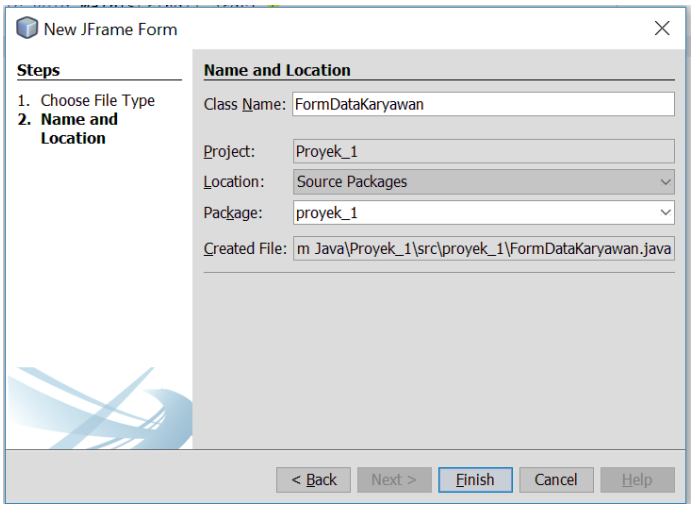
Setelah *new project* berhasil dibuat, maka kita akan masuk ke dalam kode program dengan nama `Proyek_1.java`. *File* inilah yang akan menjadi *main class* dari program kita.

C. Membuat Form

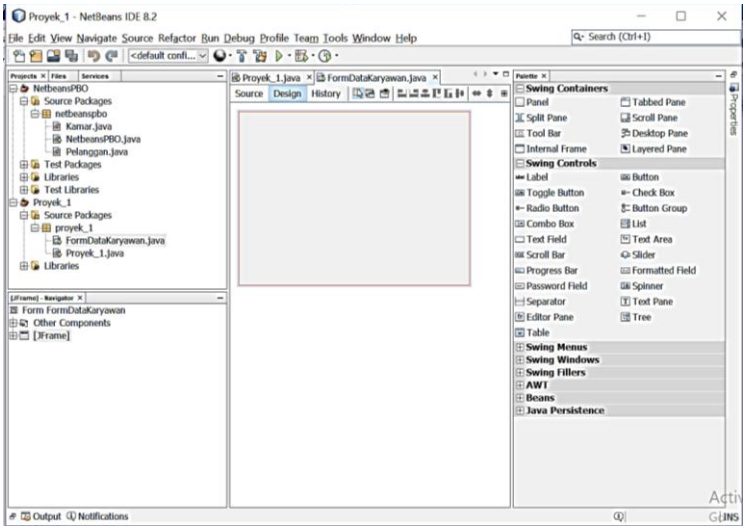
Untuk membuat sebuah *form* berbasis GUI, maka kita perlu membuat kelas baru melalui menu ”File → New File”, kemudian akan muncul jendela seperti gambar di bawah ini.



Pada jendela seperti tampak pada gambar di atas, pastikan Project yang kita pilih adalah Proyek_1, kemudian pilih ”Swing GUI Forms” pada bagian *Categories*, lalu pilih ”Jframe Form” pada bagian *File Types*. Setelah itu klik ”Next”. Kemudian, masukkan nama kelas yang ingin dibuat pada bagian *Class Name*. Sebagai contoh, coba ketikkan ”FormDataKaryawan” seperti terlihat pada gambar di bawah ini.



Jika sudah, klik tombol ”Finish”, maka Anda akan dibawa masuk ke dalam area desain GUI pada Netbeans seperti tampak pada gambar di bawah ini.

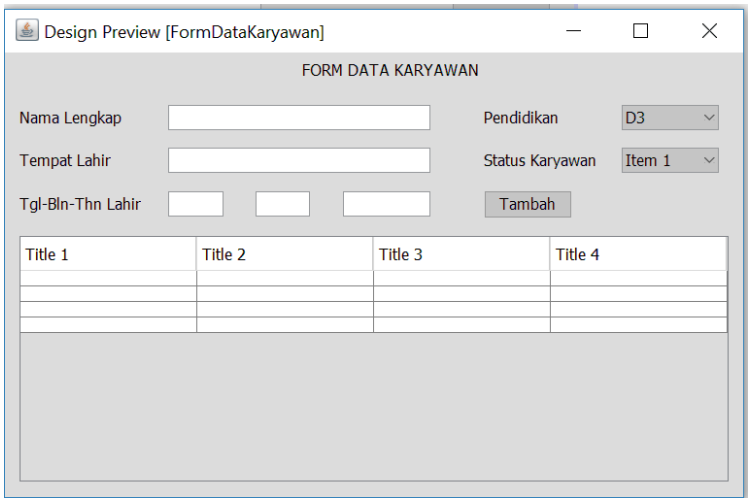


D. Meletakkan Komponen Swing pada Form

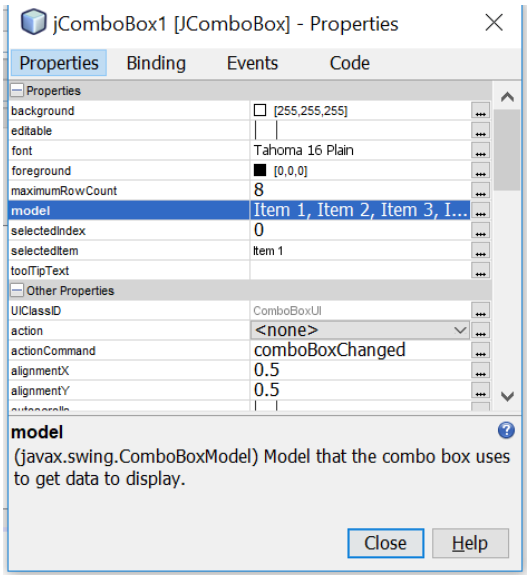
Pada area desain tersebut, Anda dapat mendesain secara mudah komponen-komponen yang ingin dimasukkan ke dalam form tersebut dengan cara *drag and drop* komponen-komponen yang ada pada tab Swing Containers atau Swing Controls yang berada pada sisi kanan jendela utama Netbeans. Sebagai contoh, kita akan membuat form isian data karyawan yang terdiri dari:

a. Nama Lengkap
b. Tempat Lahir
c. Tanggal – Bulan – Tahun Lahir
d. Pendidikan (D3, S1, S2, S3)
e. Status Karyawan (Tetap, Kontrak)

Untuk membuat form isian pada data Nama Lengkap Tempat Lahir, dan Tanggal – Bulan – Tahun Lahir, gunakan komponen `jTextField`. Kemudian untuk membuat form pemilihan pada data Pendidikan dan Status Karyawan menggunakan komponen `jComboBox`. Untuk membuat tombol ”Tambah” gunakan komponen `jButton`. Terakhir, pada bagian bawah form kita berikan komponen `jTable` untuk menampilkan setiap data karyawan yang ditambahkan ke dalam program.



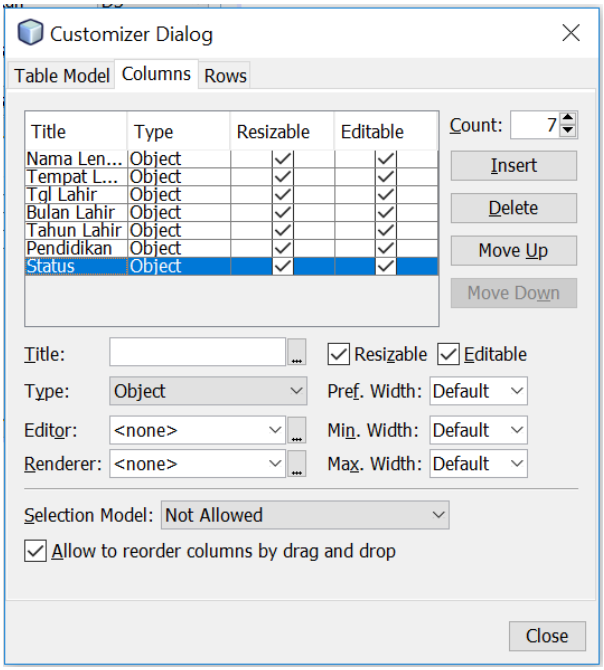
Setelah desain GUI selesai dibuat, selanjutnya kita lakukan pengaturan pada komponen jComboBox agar sesuai dengan masing-masing poin isian. Untuk itu, klik kanan pada komponen `jComboBox` ”Pendidikan”, kemudian pilih ”Properties” pada bagian paling bawah. Setelah tampil jendela Properties, maka lakukan pengeditan pada bagian ”model” dengan cara mengklik tanda titik-titik pada bagian kanannya.



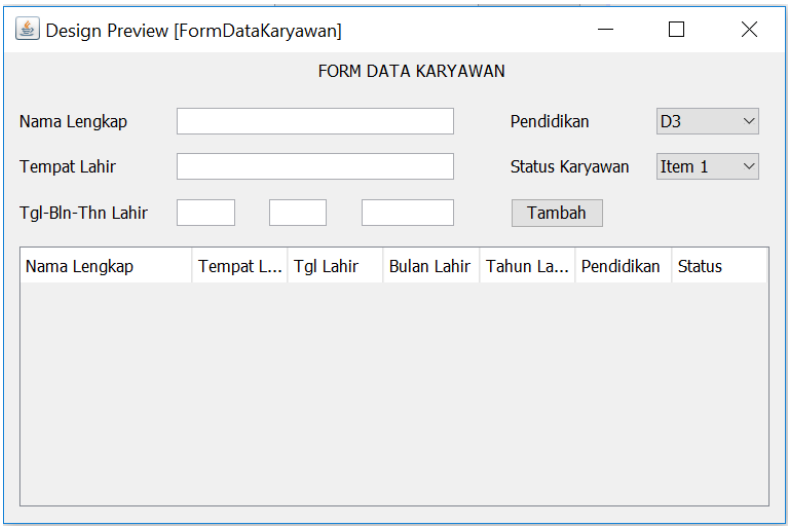
Setelah muncul jendela baru, isikanlah satu per satu kemungkinan nilai untuk komponen Pendidikan, yakni D3, S1, S2, dan S3. Jika sudah, klik tombol ”OK”. Lakukan hal yang sama untuk komponen `jComboBox` lainnya.

Selanjutnya, kita akan atur tampilan tabel agar sesuai dengan kebutuhan program untuk menampilkan setiap data karyawan yang ditambahkan ke dalam program. Untuk itu, klik kanan pada komponen jTable yang ada pada form, kemudian pilih ”Table Contents”. Pada tab ”Columns” kita bisa mengganti nama *header* untuk setiap kolom sesuai dengan

kebutuhan kita, serta kita juga dapat menambah jumlah kolom pada tabel. Selain itu kita juga bisa mengatur ukuran ”Prefered Size” untuk masing-masing kolom.



Lalu, pada tab ”Rows” ubah jumlah baris dari 4 menjadi 0. Jika sudah, klik tombol ”Close”, maka seketika tampilan `jTable` pada form GUI kita akan berubah seperti ini:



Karena kita menggunakan jTable, maka kita perlu mengimpor *library* `import javax.swing.table.DefaultTableModel` agar operasi pada jTable dapat dilakukan. Setelah itu, kita juga perlu membuat objek baru yang bertipe `DefaultTableModel`. Lihat gambar di bawah ini untuk lebih jelasnya.

```
Source  Design  History
1   package proyek_1;
2
3   import javax.swing.table.DefaultTableModel;
4
5   public class FormDataKaryawan extends javax.swing.JFrame {
6
7       DefaultTableModel model;
8
9       public FormDataKaryawan() {
10          initComponents();
11
12          //Membuat Table Model
13          model = (DefaultTableModel) jTable1.getModel();
14
15          //Menambahkan TableModel ke jTable1
16          jTable1.setModel(model);
17      }
```

E. Menulis Kode Aksi pada Tombol

Sebagaimana lazimnya sebuah tombol pada sebuah program komputer, ketika diklik maka sesuatu akan diproses oleh program tersebut. Dengan kata lain, program akan mengeksekusi sebuah aksi tertentu ketika tombol diklik oleh user. Pada contoh kali ini, aksi yang akan dilakukan ketika tombol ”Tambah” ditekan ada dua:

1. Membaca nilai yang diinputkan oleh user, baik pada `jTextField`, maupun pada `jComboBox`.
2. Menampilkan data yang telah diisi user ke dalam `jTable`.

Untuk itu, klik kanan pada tombol ”Tambah”, lalu pilih ”Events → Action → ActionPerformed”, maka secara otomatis *method* `jButton1ActionPerformed()` akan terbentuk pada area *source code*.

```
182
183     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
184         // TODO add your handling code here:
185     }
```

Di dalam *method* `jButton1ActionPerformed()` inilah kita akan mengetikkan kode program untuk membaca data yang diinputkan oleh user dan menampilkannya ke dalam tabel. Namun sebelumnya, kita perlu mendefinisikan variabel-variabel global yang akan digunakan untuk menampung data yang diinputkan oleh user. Perhatikan gambar berikut:

```
5   public class FormDataKaryawan extends javax.swing.JFrame {
6
7       DefaultTableModel model;
8
9       private String NamaLengkap;
10      private String TempatLahir;
11      private String Pendidikan;
12      private String Status;
13      private int tgl,bln,thn;
```

Setelah itu, kita ketikkan kode aksi pada *method* `jButton1ActionPerformed()` sebagai berikut:

```
187     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
188         //Membaca dan menyimpan data yang diinputkan oleh user ke dalam variabel
189         NamaLengkap = jTextField1.getText();
190         TempatLahir = jTextField2.getText();
191         tgl = Integer.parseInt(jTextField3.getText());
192         bln = Integer.parseInt(jTextField4.getText());
193         thn = Integer.parseInt(jTextField5.getText());
194         Pendidikan = jComboBox1.getSelectedItem().toString();
195         Status = jComboBox2.getSelectedItem().toString();
196
197         //Memasukkan data ke dalam jTable
198         Object [] data = {NamaLengkap, TempatLahir, tgl, bln, thn, Pendidikan, Status};
199         model.addRow(data);
200     }
```

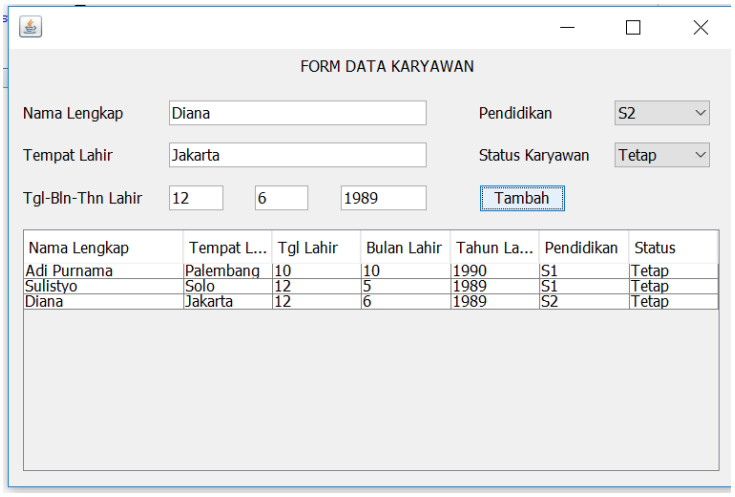
Pada kode program di atas bisa dilihat bahwa *method* getText() digunakan untuk membaca data yang diinputkan oleh user melalui jTextField. Data yang dibaca tersebut secara otomatis bertipe String. Tapi perhatikan pada baris ke-191 s.d. baris ke-193, di sana terdapat proses konversi dari tipe String menjadi tipe `int` menggunakan method `parseInt()` milik objek `Integer`. Hal ini dilakukan karena data dari field-field tersebut akan disimpan dalam variabel bertipe `int`. Kemudian, kita juga bisa melihat bahwa terdapat *method* `getSelectedItem()` untuk membaca *item* yang dipilih oleh user melalui komponen `jComboBox`.

F. Menjalankan Program Melalui *Main Class*

Kita telah berhasil membuat form berbasis GUI pada kelas FormDataKaryawan. Langkah selanjutnya adalah melakukan instansiasi untuk membuat sebuah objek yang merujuk pada kelas FormDataKaryawan. Perhatikan kode program di bawah ini:

```java
package proyek_1;

public class Proyek_1 {
    public static void main(String[] args) {
        FormDataKaryawan form = new FormDataKaryawan();
        form.setVisible(true);
    }
}
```

Jalankan program melalui menu ”Run → Run Project (Proyek_1)”.



BAB 11
Pengaksesan Basis Data

11.1. Capaian Belajar

Setelah mempelajari bab ini, mahasiswa diharapkan:
1. Memahami proses koneksi database dengan aplikasi
2. Memahami cara membuat database dengan MySQL
3. Memahami cara menyimpan dan mengambil data dari/ ke database

11.2. Indikator

1. Mahasiswa mampu mengimplementasikan koneksi database dengan aplikasi yang dibuat.
2. Mahasiswa mampu membuat database dengan MySQL
3. Mahasiswa mampu mengimplementasikan proses penyimpanan dan pembacaan data dari/ ke database.

11.3. Uraian Materi

Pada pembahasan di bab sebelumnya, kita telah berhasil membuat aplikasi berbasis GUI yang dapat menyimpan dan menampilkan data ke dalam tabel. Akan tetapi, masih ada kekurangan pada aplikasi tersebut, yakni data yang disimpan dalam aplikasi tersebut masih bersifat *volatile*, artinya ketika aplikasi ditutup, maka data yang tadinya disimpan pun hilang.

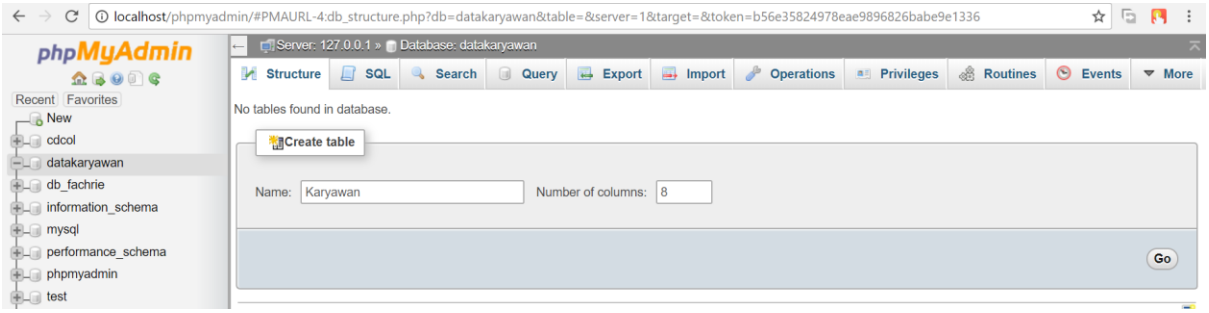
Pada praktiknya, sebuah aplikasi seringkali membutuhkan sistem penyimpanan data yang rapi dan terorganisasi dengan baik. Oleh karena itu, penggunaan basis data sangat diperlukan untuk dapat mengatasi hal tersebut. Integrasi antara aplikasi berbasis Java dengan DBMS (*Database Management System*) merupakan hal yang sudah lazim, dan dapat dilakukan menggunakan library jdbc yang biasanya sudah tersedia dalam IDE Netbeans. Pada bab ini, kita akan membahas mengenai proses integrasi antara aplikasi Java dengan DBMS MySQL.

A. Persiapan Basis Data

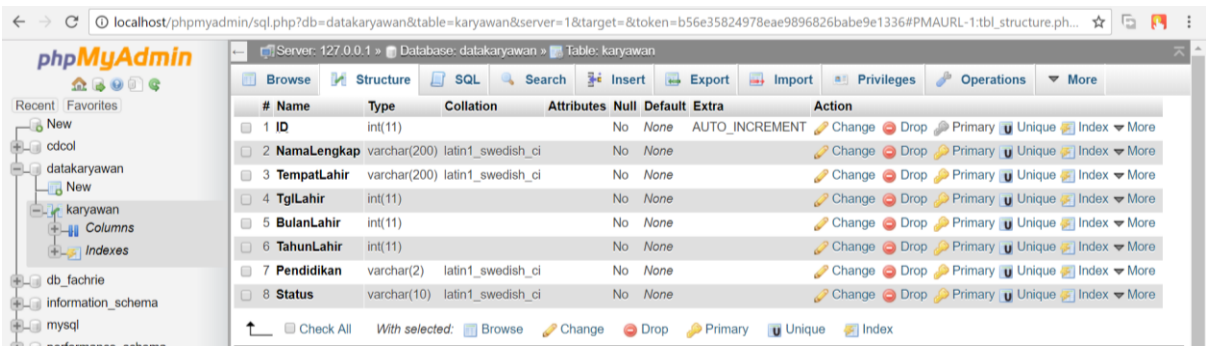
Untuk menggunakan basis data MySQL, salah satu alternatif yang mudah adalah dengan menginstall aplikasi XAMPP yang di dalamnya telah terintegrasi fitur phpMyAdmin yang merupakan *platform* MySQL berbasis GUI, sehingga memudahkan kita untuk merancang dan membuat basis data relasional. Anda dapat mengunduh aplikasi tersebut pada link: https://www.apachefriends.org/download.html.

Setelah Anda berhasil mengunduh dan melakukan instalasi XAMPP pada komputer Anda, maka buka XAMPP-Control Panel, lalu tekan tombol ”Start” pada fitur ”Apache” dan ”MySQL” seperti yang terlihat pada gambar di bawah.

Setelah itu, buka aplikasi *web browser* pada komputer Anda, lalu ketik URL: localhost/phpmyadmin. Apabila Anda masuk ke halaman mesin pencari Google, coba nonaktifkan dahulu sementara jaringan internet Anda. Jika sudah, Anda bisa mulai membuat basis data baru dan membuat tabel baru seperti di bawah ini. Pada contoh ini, kita membuat basis data dengan nama ”DataKaryawan” dan membuat sebuah tabel bernama ”Karyawan” yang terdiri dari delapan buah kolom. Perhatikan gambar di bawah.

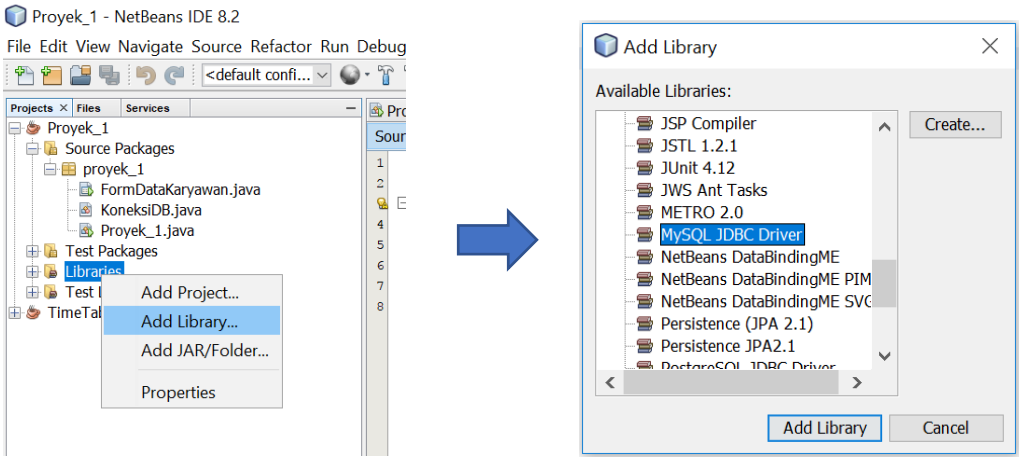


Kemudian buatlah kedelapan kolom pada tabel tersebut yang terdiri dari: ID, NamaLengkap, TempatLahir, TglLahir, BulanLahir, TahunLahir, Pendidikan, dan Status. Jadikan kolom/ atribut ID sebagai *primary key* dan set *Auto increment*. Jika sudah, maka selanjutnya kita lakukan pengetikan kode di dalam Java.

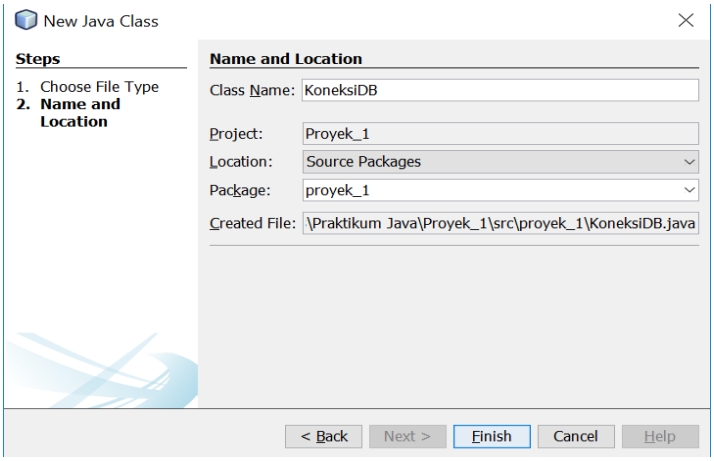


B. Membangun Koneksi dengan JDBC

Pada package Proyek_1 yang telah kita buat pada bab sebelumnya, Lihat tab ”Projects” pada bagian kiri Netbeans, lalu klik kanan pada ”Libraries”, kemudian pilih ”Add Library”. Setelah itu, pilih ”MySQL JDBC Driver” pada kolom ”Available Libraries”.



Berikutnya, buka menu ”File → New File”, lalu pilih ”Java” pada bagian *Categories* dan pilih “Java Class” pada bagian *File Types*, lalu klik tombol “Next”. Setelah itu, pada bagian *Class Name,* isi dengan “KoneksiDB”.



Pada kode program dari kelas KoneksiDB, kita tambahkan kode program seperti di bawah ini:

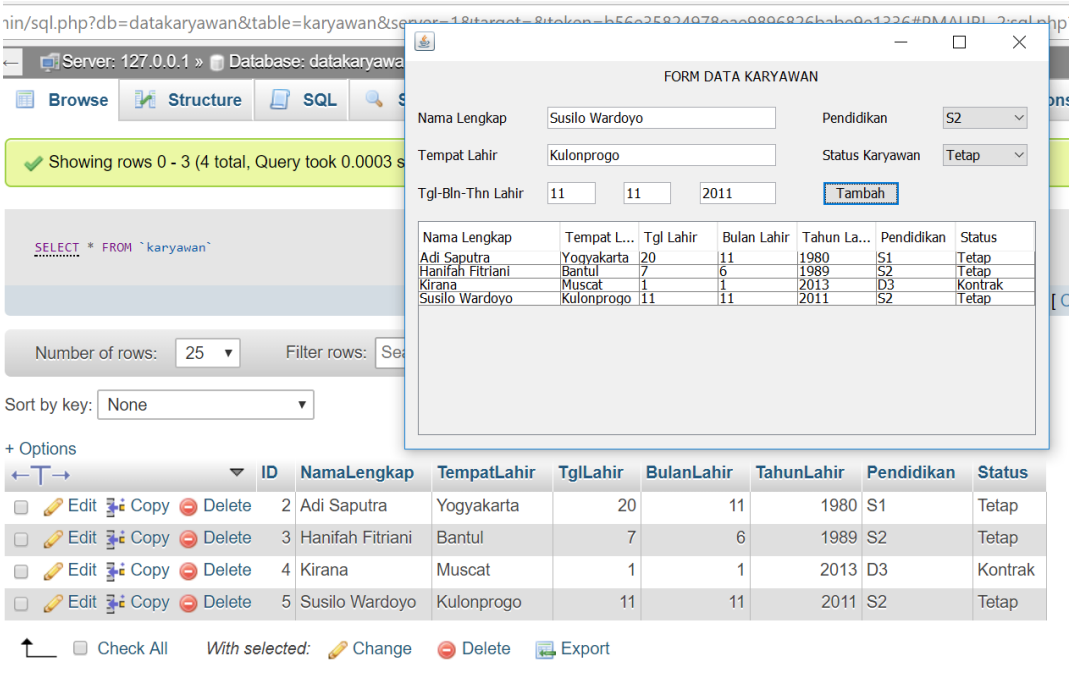
```
Source  History
1  package proyek_1;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.SQLException;
6
7  public class KoneksiDB {
8      //Membuat variabel bertipe Connection
9      private static Connection koneksi;
10
11     public static Connection getKoneksi(){
12         //method ini berfungsi untuk membuat koneksi ke MySQL
13         if (koneksi == null){
14
15             try{
               String url = "jdbc:mysql://localhost:3306/datakaryawan";
17                 String username = "root";
18                 String password = "";
19
20                 DriverManager.registerDriver(new com.mysql.jdbc.Driver());
21
22                 koneksi = DriverManager.getConnection(url,username,password);
23             } catch (SQLException e){
24                 System.out.println("Gagal membuat koneksi");
25             }
26         }
27
28         return koneksi;
29     }
30 }
```

Pada kode program di atas, kita deklarasikan sebuah variabel bertipe Connection. Variabel ini kita beri nama ”koneksi”. Kemudian kita buat sebuah *method* bernama `getKoneksi()` yang berfungsi untuk membuat koneksi antara aplikasi Java yang kita buat dengan DBMS MySQL pada phpMyAdmin. Pada *method* tersebut, kita menginisialisasi tiga buah variabel, yakni `url`, `username`, dan `password`. Variabel `url` berfungsi menyimpan lokasi penyimpanan database di dalam localhost. Variabel `username` digunakan untuk menyimpan username dari DBMS yang kita gunakan, dan variabel `password` digunakan untuk menyimpan password dari DBMS yang kita gunakan. Apabila tidak ada password yang diset, maka variabel `password` cukup diisi dengan karakter kosong (`""`).

Selanjutnya, kita tambahkan baris kode program di bawah ini untuk menyimpan data ke database pada *method* `jButton1ActionPerformed()`. Letakkan kode di bawah ini di bawah kode program yang sudah ada pada *method* `jButton1ActionPerformed()`.

```
204         //Simpan ke dalam database MySQL
205         try{
206             Connection conn = KoneksiDB.getKoneksi();
207 
208             String sql = "INSERT INTO KARYAWAN (NamaLengkap,TempatLahir,TglLahir,"
209                     + "BulanLahir,TahunLahir,Pendidikan,Status) VALUES (?,?,?,?,?,?,?)";
210 
212             PreparedStatement p = conn.prepareStatement(sql);
213 
214             p.setString(1, NamaLengkap);    //kolom ke-1 diset auto increment
215             p.setString(2, TempatLahir);
216             p.setInt(3, tgl);
217             p.setInt(4, bln);
218             p.setInt(5, thn);
219             p.setString(6, Pendidikan);
220             p.setString(7, Status);
221 
222             p.executeUpdate();
223             p.close();
224         } catch(SQLException e){
225             System.out.println("Terjadi error");
        }
```

Apabila koneksi database berhasil dibangun dan proses INSERT query berhasil dilakukan, maka seharusnya data yang diinputkan melalui aplikasi Java yang kita buat juga tersimpan di dalam database MySQL. Perhatikan gambar di bawah ini.



Namun, jika Anda cermati, ternyata ada sedikit masalah pada aplikasi yang telah kita bangun sejauh ini. Ketika Anda menutup aplikasi tersebut, lalu membukanya lagi, ternyata data karyawan yang telah kita simpan sebelumnya tidak tampil pada tabel. Kenapa hal ini bisa terjadi? Karena kita belum melakukan *load data* dari database setiap kali aplikasi dibuka. Oleh karena itu, kita perlu membuat sebuah *method* baru lagi di dalam kelas FormDataKaryawan yang berfungsi untuk melakukan *load data* dari database ke dalam tabel.

```
33      public void loadData(){
34          try{
35              //Membuat koneksi
36              Connection conn = KoneksiDB.getKoneksi();
                Statement s = conn.createStatement();
38
39              //Membuat query SELECT
40              String sql = "SELECT * FROM KARYAWAN";
41              ResultSet r = s.executeQuery(sql);
42
43              //Membaca data di dalam database baris per baris
44              while(r.next()){
45                  //Membuat objek 'obj' untuk menampung data yang dibaca dari DB
46                  Object [] obj = new Object[7];
47                  obj[0] = r.getString("NamaLengkap");
48                  obj[1] = r.getString("TempatLahir");
49                  obj[2] = r.getInt("TglLahir");
50                  obj[3] = r.getInt("BulanLahir");
51                  obj[4] = r.getInt("TahunLahir");
52                  obj[5] = r.getString("Pendidikan");
53                  obj[6] = r.getString("Status");
54
55                  //tampilkan satu baris data ke dalam tabel
56                  model.addRow(obj);
57              }
58
59              //menutup hasil penelusuran dan koneksi
60              r.close();
61              s.close();
62          } catch(SQLException e){
63              System.out.println("Terjadi error");
64          }
65      }
```

Setelah itu, kita panggil *method* `loadData()` tersebut di dalam *constructor* FormDataKaryawan.

```
20      public FormDataKaryawan() {
21          initComponents();
22
23          //Membuat Table Model
24          model = (DefaultTableModel) jTable1.getModel();
25
26          //Menambahkan TableModel ke jTable1
27          jTable1.setModel(model);
28
29          //Melakukan load data dari database
            loadData();
31      }
```

Maka, kini setiap kali kita membuka aplikasi FormDataKaryawan, secara otomatis semua data yang tersimpan di dalam database akan dibaca dan ditampilkan di dalam tabel.

BAB 12
Membaca File dari Ms. Excel

12.1. Capaian Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan:
1. Memahami cara untuk membaca dan menulis data ke dalam *file* Excel
2. Memahami cara menerapkan *library* jxl dalam program

12.2. Indikator

1. Mahasiswa mampu mengimplementasikan proses pembacaan dan penulisan data dari/ ke dalam *file* Excel.
2. Mahasiswa mampu menerapkan *library* jxl dalam program

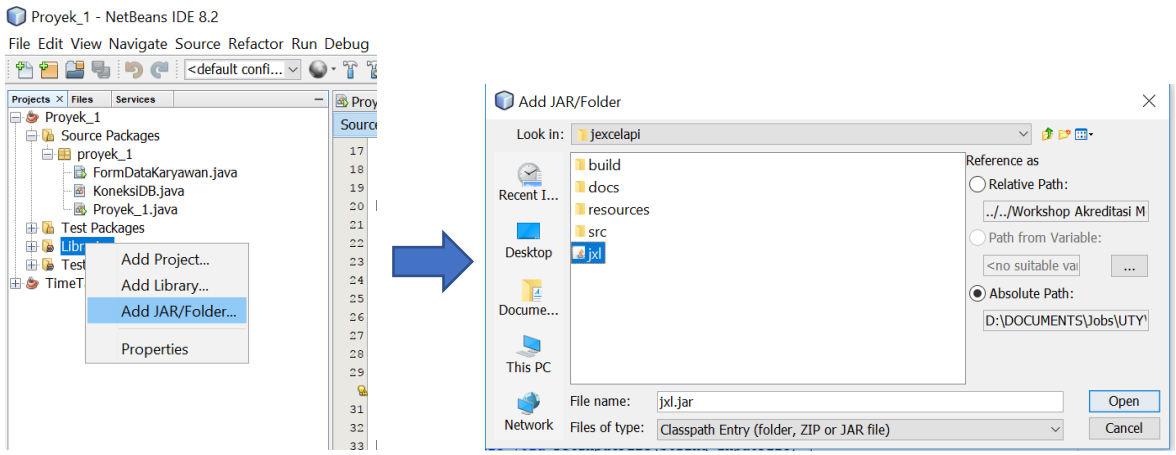
12.3. Uraian Materi

Salah satu keunggulan Java adalah kaya akan *library* yang sangat berguna dalam membangun aplikasi. Salah satu fitur yang akan dibahas pada bab ini adalah *library* jxl yang merupakan salah satu *library* yang cukup banyak dipakai untuk membaca data pada dokumen berekstensi **.xls**.

A. Persiapan Library jxl

Library jExcelAPI (jxl) dapat secara mudah dan gratis untuk Anda unduh pada link: https://sourceforge.net/projects/jexcelapi/files/. *Library* tersebut memang termasuk library yang telah cukup lama dikembangkan, namun *library* ini adalah salah satu yang paling mudah digunakan dan ringan. *Library* jxl mampu membaca *file* Excel versi 95, 97, 2000, XP, dan 2003. Meskipun tidak dapat membaca format *file* Excel terbaru saat ini, namun masih banyak yang menggunakan *library* ini.

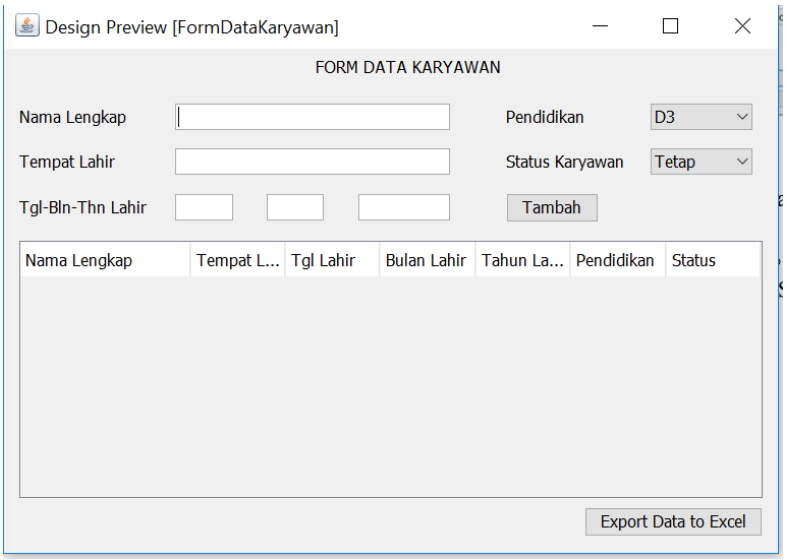
Pada bab ini, kita akan menggunakan kembali *package* Proyek_1 yang telah kita buat sebelumnya. Setelah Anda berhasil mengunduh *library* jxl, maka lakukan *extract file* tersebut sehingga kita bisa menemukan folder yang di dalamnya terdapat aplikasi **jxl.jar**. Setelah itu, pada tab ”Projects” di Netbeans, klik kanan pada bagian ”Libraries”, lalu pilih ”Add JAR/Folder”. Setelah itu kita *load* file jxl.jar yang telah kita unduh sebelumnya.



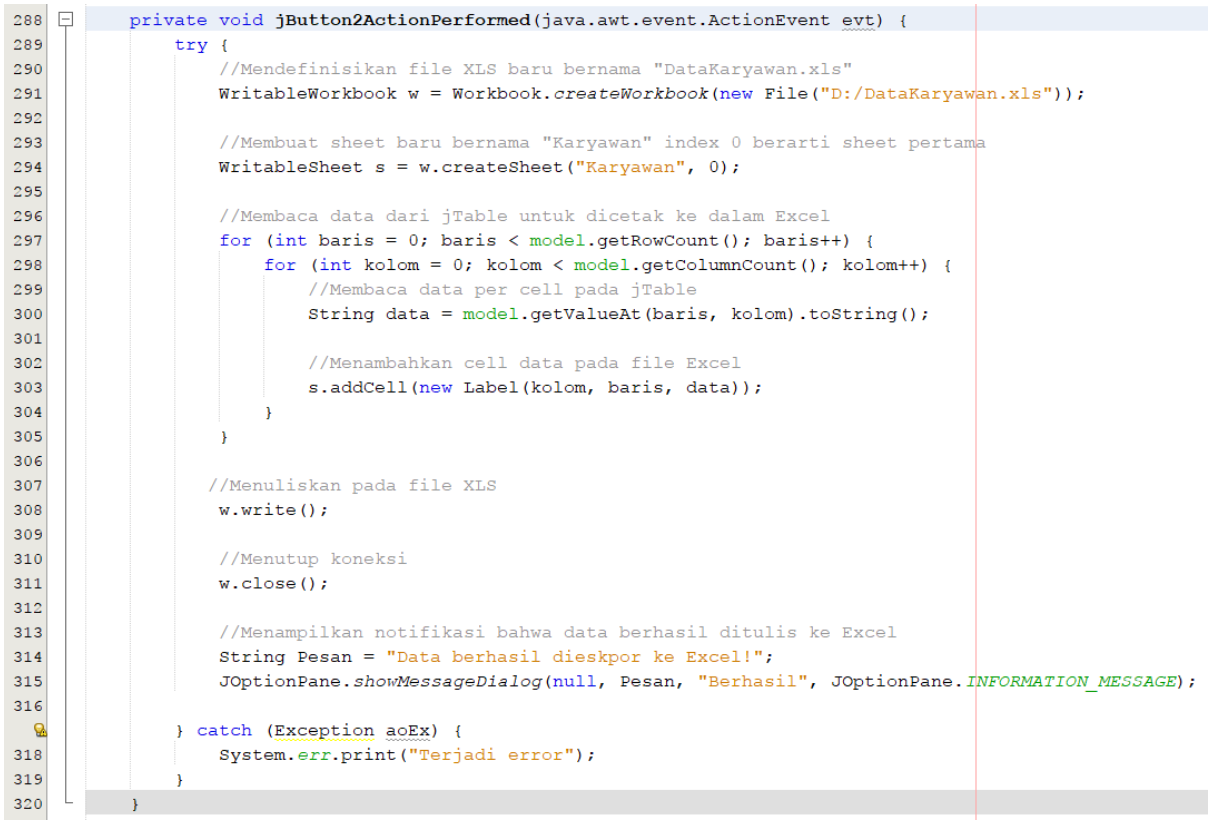
B. Mencetak *file* Excel

Pada dasarnya, proses mencetak atau menulis data ke dalam *file* excel adalah dengan cara menempatkan data satu per satu ke dalam *cell* yang ada di dalam Excel. Penempatan data tersebut ditentukan oleh koordinat *cell* yang dituju. Misalnya, kode `s.addCell(new Label(0, 0, "Hello World"))` akan meletakkan kalimat ”Hello World” pada *cell* baris ke-0 dan kolom ke-0, atau sama saja dengan *cell* A1.

Pada pembahasan ini, kita akan mempraktikkan bagaimana mencetak data yang ada pada jTable agar tersimpan dalam *file* .xls. Sebelumnya, kita tambahkan dulu sebuah tombol untuk melakukan aksi tersebut, kemudian kita akses *method* `jButton2ActionPerformed()` dari tombol tersebut untuk kemudian kita isi kode program untuk mencetak *file* .xls. Sebagai contoh, pada modul ini, tombol ”Export to Excel” kita letakkan pada bagian bawah sebelah kanan dari jTable. Perhatikan gambar berikut ini untuk lebih jelasnya.



Kode program yang kita tuliskan di dalam method jButton2ActionPerformed() dapat dilihat pada gambar di bawah ini.



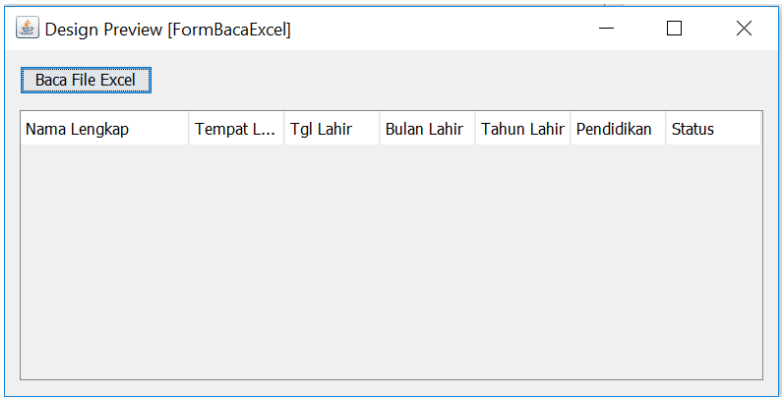
```
288     private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
289         try {
290             //Mendefinisikan file XLS baru bernama "DataKaryawan.xls"
291             WritableWorkbook w = Workbook.createWorkbook(new File("D:/DataKaryawan.xls"));
292 
293             //Membuat sheet baru bernama "Karyawan" index 0 berarti sheet pertama
294             WritableSheet s = w.createSheet("Karyawan", 0);
295 
296             //Membaca data dari jTable untuk dicetak ke dalam Excel
297             for (int baris = 0; baris < model.getRowCount(); baris++) {
298                 for (int kolom = 0; kolom < model.getColumnCount(); kolom++) {
299                     //Membaca data per cell pada jTable
300                     String data = model.getValueAt(baris, kolom).toString();
301 
302                     //Menambahkan cell data pada file Excel
303                     s.addCell(new Label(kolom, baris, data));
304                 }
305             }
306 
307            //Menuliskan pada file XLS
308             w.write();
309 
310             //Menutup koneksi
311             w.close();
312 
313             //Menampilkan notifikasi bahwa data berhasil ditulis ke Excel
314             String Pesan = "Data berhasil dieskpor ke Excel!";
315             JOptionPane.showMessageDialog(null, Pesan, "Berhasil", JOptionPane.INFORMATION_MESSAGE);
316 
317         } catch (Exception aoEx) {
318             System.err.print("Terjadi error");
319         }
320     }
```

Pada gambar di atas kita membuat sebuah file baru pada direktori D: yang kita beri nama DataKaryawan.xls (baris ke-291). Setelah itu, kita membuat pula sebuah *sheet* baru bernama ”Karyawan” (baris ke-294). Pada baris ke-297 s.d. baris ke-305, kita membaca data pada jTable secara *cell* per *cell*, kemudian mengalokasikan *cell* pada *file* DataKaryawan.xls yang nantikan akan digunakan untuk menulis data yang dibaca tersebut. Selanjutnya, pada baris ke-314 s.d. baris ke-315, kita menampilkan *dialog box* untuk menginformasikan kepada *user* bahwa data telah berhasil ditulis dalam *file* Excel.

C. Membaca File Excel

Pada pembahasan ini, kita akan membuat sebuah form GUI baru untuk menampilkan data yang dibaca dari file Excel. Mula-mula kita buat ”New Project” dari menu ”File → New Project”, lalu pilih ”Java” pada bagian *Categories*, kemudian pilih ”Java Application” pada bagian *Projects*. Beri nama ”BacaExcel” pada bagian *Project Name*. Lakukan import *library* jxl ke dalam *project* BacaExcel ini dengan cara seperti yang telah dijelaskan sebelumnya.

Setelah itu, kita buat lagi sebuah kelas baru melalui menu ”File → New File”, lalu pilih ”Swing GUI Forms” pada bagian *Categories*, lalu pilih ”JFrame Form” pada bagian *File Types*. Beri nama *file* tersebut dengan nama ”FormBacaExcel”. Selanjutnya kita desain form GUI seperti gambar di bawah ini:



Pada tombol ”Baca File Excel”, kita klik dua kali sehingga secara otomatis Netbeans akan membuatkan *method* `jButton1ActionPerformed()`. Pada *method* tersebut kita isikan kode program di bawah ini:

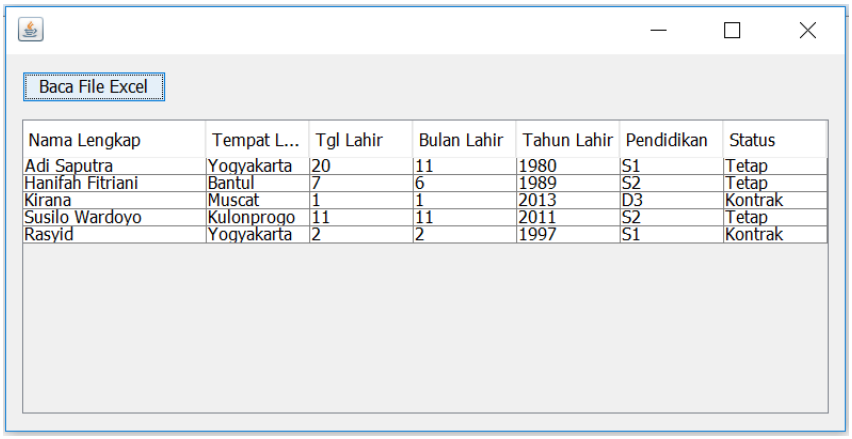
```
95      private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
96          //mengambil direktori file excel yang ingin dibaca
97          String direktori = "D:/DataKaryawan.xls";
98
99          //Membaca isi file
100         File inputWorkbook = new File(direktori);
101         Workbook w;
102         try {
103             w = Workbook.getWorkbook(inputWorkbook);
104
105             // Ambil sheet pertama, nomer 0 menandakan sheet ke 1
106             Sheet sheet = w.getSheet(0);
107
108             // Looping sebanyak baris dan kolom yang ada
109             Object [] obj = new Object[7];
110             for (int i = 0; i < sheet.getRows(); i++) {
111                 for (int j = 0; j < sheet.getColumns(); j++) {
112                     //Membaca data pada cell kolom ke-j, baris ke-i
113                     Cell cell = sheet.getCell(j, i);
114
115                     //Menyimpan data sementara pada obj
116                     obj[j] = cell.getContents();
117                 }
118                 //Menampilkan data obj ke dalam tabel baris per baris
119                 model.addRow(obj);
120             }
121         } catch (BiffException e) {
122             e.printStackTrace();
123         } catch (IOException ex) {
124             Logger.getLogger(FormBacaExcel.class.getName()).log(Level.SEVERE, null, ex);
125         }
126     }
```

*Catatan: Anda harus terlebih dahulu membuat *table model* untuk jTable yang Anda buat pada FormBacaExcel. Cara membuat *table model* telah dijelaskan pada Bab 10.

Selanjutnya, jangan lupa melakukan instansiasi objek FormBacaExcel pada *main class*, yakni *file* BacaExcel.java.

```
1   package bacaexcel;
2
3   public class BacaExcel {
4
5       public static void main(String[] args) {
6           FormBacaExcel form = new FormBacaExcel();
7           form.setVisible(true);
8       }
9
10  }
```

Ketika program dijalankan dan tombol ”Baca File Excel” diklik, maka aplikasi akan membaca *file* Excel yang telah disiapkan sebelumnya dan menampilkannnya ke dalam tabel.



Bab 13
Java Class Library

13.1. Capaian Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan:
1. Mengenal Java Library
2. Memahami penggunaan Java Library
3. Memahami cara pembuatan Java Library.

13.2. Indikator

1. mahasiswa mampu memanfaatkan library pada Java
2. melakukan kompilasi kode program Java menjadi aplikasi .jar.
3. Mahasisw mampu membangun aplikasi berbasis GUI sederhana untuk pemesanan tiket pesawat terbang.
4. mahasiswa mampu mengintegrasikan beberapa *library* dalam satu program

13.3. Uraian Materi

A. Library

Library merupakan kumpulan perintah yang dapat digunakan oleh aplikasi maupun platform lain. Pada dasarnya library sama dengan sebuah aplikasi. Library dapat menampung kelas-kelas yang berfungsi untuk melakukan tugas tertentu dan dipanggil dari aplikasi lain. Dalam pemrograman Java, library di-compile menjadi Java Archive (JAR).
Library akan berguna untuk situasi sebagai berikut

a. Ketika membangun banyak aplikasi yang memilki komponen sama seperti services maupun method.
b. Ketika membangun aplikasi yang bervariasi, seperti versi trial dan versi paid dimana keduanya membutuhkan komponen inti yang sama.

Untuk membuat library, hanya perlu memindahkan kelas atau file yang ingin digunakan kembali pada aplikasi lain ke sebuah package tersendiri dan di-compile menjadi sebuah .jar
Pemanggilan kelas maupun method yang ada pada library sama seperti pemanggilan kelas atau method dalam project, yaitu dengan melakukan import pada kelas yang akan memanggil method dari library.

B. Distribusi aplikasi

Aplikasi yang telah kita buat menggunakan Java dapat didistribusikan menjadi sebuah aplikasi independen dengan cara mengkompilasi kode program Java tersebut menjadi *file* berekstensi **.jar**. Jika kita menggunakan IDE Netbeans dalam membangun aplikasi, maka proses kompilasi *file* .jar ini sangat mudah. Anda cukup masuk ke menu ”Run → Clean and Build Project”, maka secara otomatis Netbeans akan membuatkan *file* .jar pada direktori ”dist” pada *project* yang Anda buat.

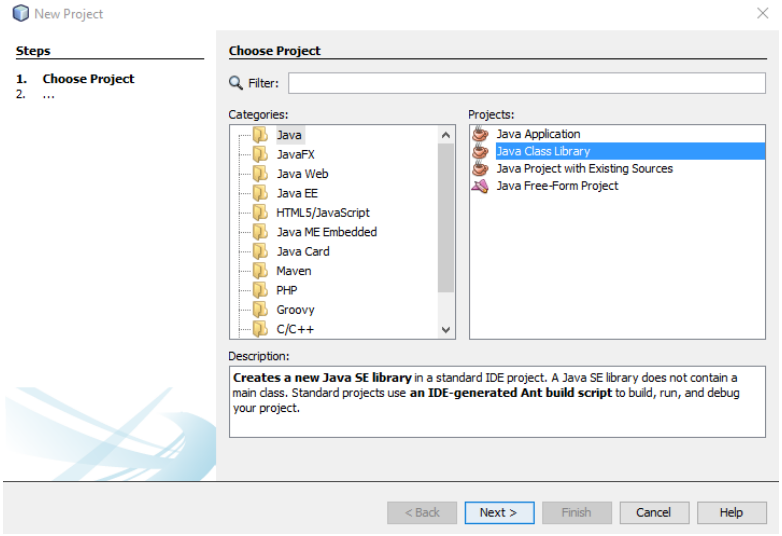
Perlu diingat, apabila Anda membuat aplikasi yang terkoneksi dengan database MySQL, maka Anda harus mengaktifkan dulu server database-nya, baru dapat menjalankan aplikasi .jar. Jika Anda merasa kurang praktis menggunakan database yang tidak terintegrasi di dalam aplikasi yang Anda buat, maka ada beberapa alternatif DBMS lain yang sifatnya terintegrasi dengan aplikasi yang dibuat, misalnya: Java DB, Apache Derby, H2, atau HSQLDB. DBMS tersebut memungkinkan kita membuat aplikasi yang terintegrasi dengan DBMS di dalamnya, sehingga ketika aplikasi dijalankan, secara otomatis DBMS juga diaktifkan.

13.4. Latihan

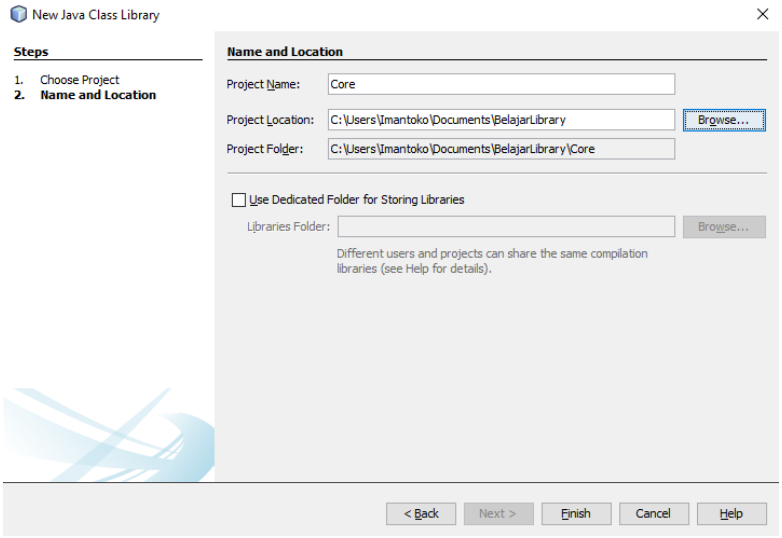
A. PEMBUATAN LIBRARY

Langkah-langkah pembuatan java library sebagai berikut

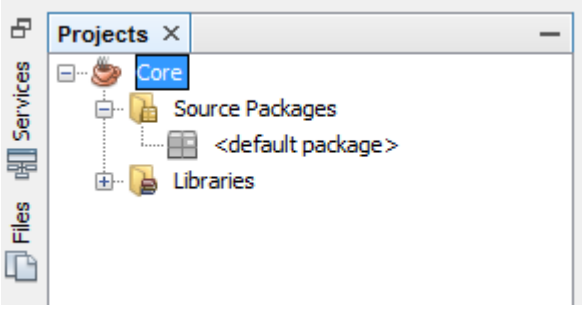
a. Klik File → New Project → Pilih Categories: Java, Project: Java Class Library.



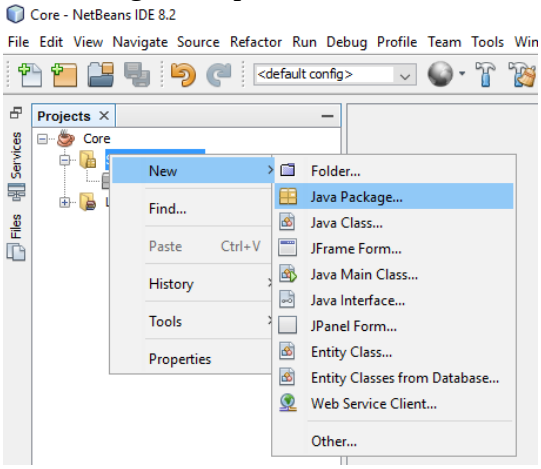
b. Klik Next, kemudian berikan nama project dan pilih lokasi penyimpanan. Klik Finish



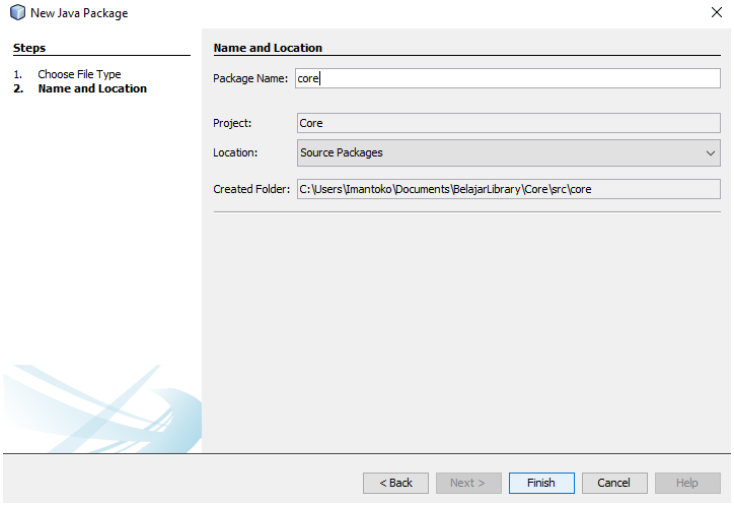
c. Setelah klik finish, maka pada Project Browser akan muncul seperti pada gambar berikut



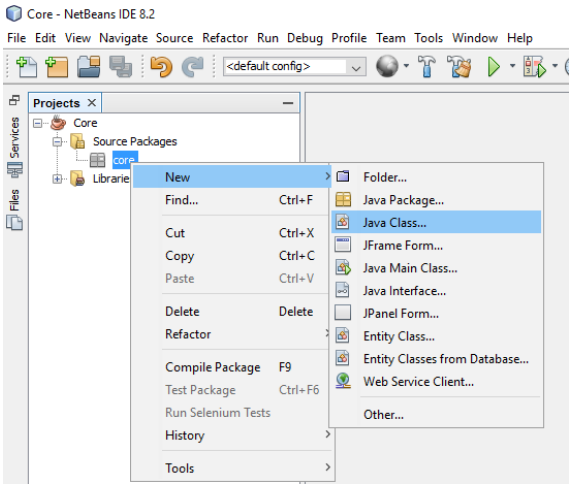
d. Klik kanan pada Source Packages dan pilih New → Java Packages…



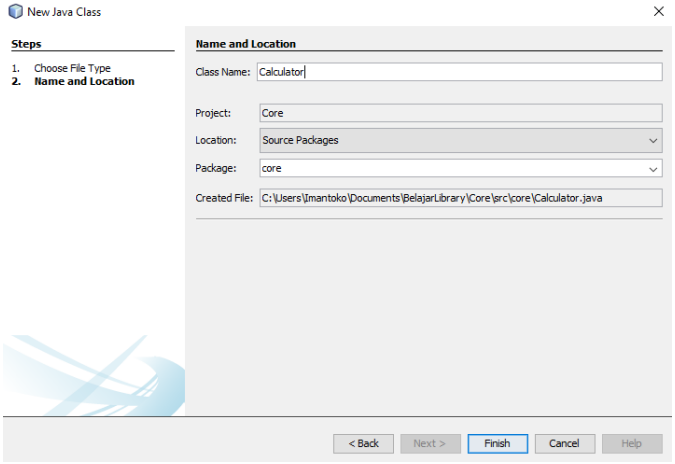
e. Beri nama package dan klik finish.



f. Klik kanan pada package yang telah diberi nama tadi dan pilih New → Java Class…



g. Beri nama class dan klik finish

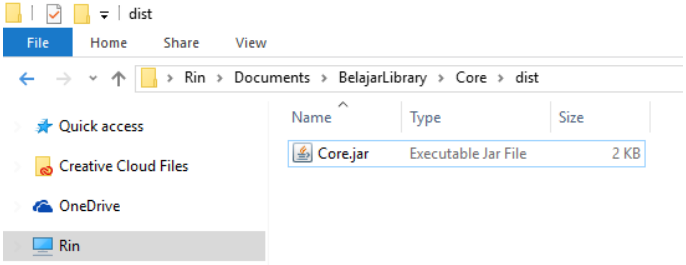


h. Tulliskan syntax berikut pada Calculator.class

```
public class Calculator {
    //PENJUMLAHAN
    public static double sum(double firstNumber, double secondNumber){
        return firstNumber + secondNumber;
    }
    //PENGURANGAN
    public static double subtract(double firstNumber, double secondNumber){
        return firstNumber - secondNumber;
    }
    //PERKALIAN
    public static double multiple(double firstNumber, double secondNumber){
        return firstNumber * secondNumber;
    }
    //PEMBAGIAN
    public static double divide(double firstNumber, double secondNumber){
        return firstNumber / secondNumber;
    }
}
```

i. Klik Run → Build Project.

Library yang sudah dibuild akan berada di folder /dist.

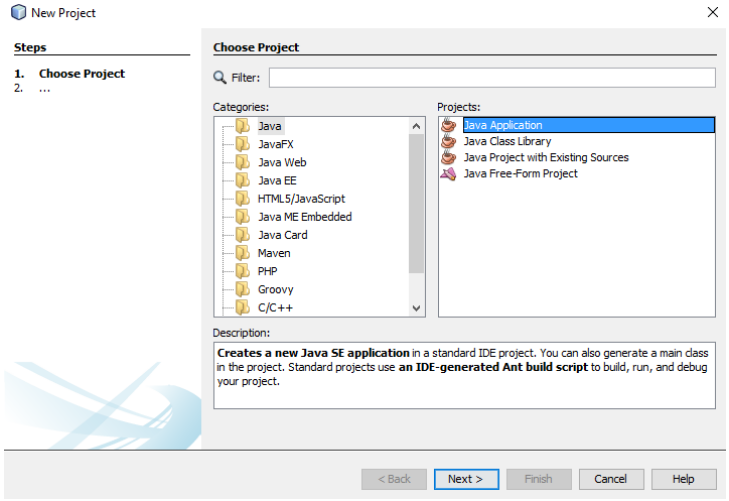


Library Core.jar ini sudah dapat digunakan oleh aplikasi lainnya.

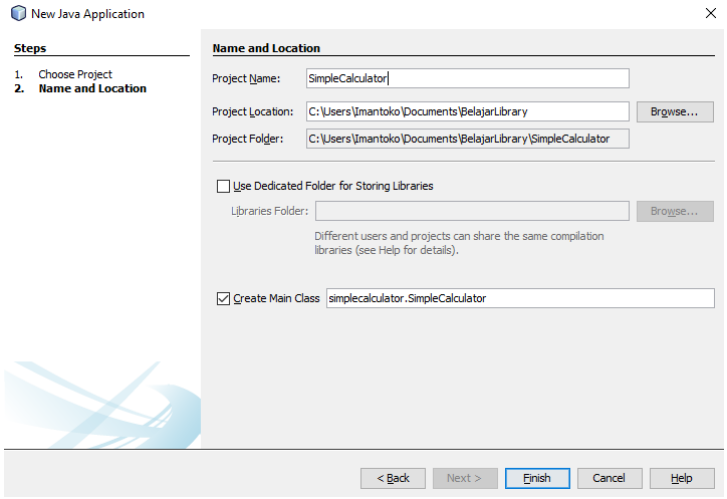
B. IMPLEMENTASI LIBRARY DALAM APLIKASI

a. Menambah Library ke aplikasi

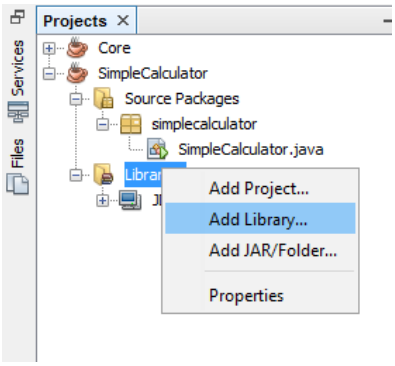
Klik File → New Project → Pilih Categories: Java, Project: Java Application



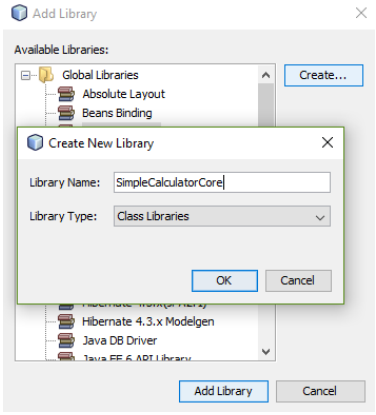
Klik Next, kemudian berikan nama project dan pilih lokasi penyimpanan. Klik Finish.



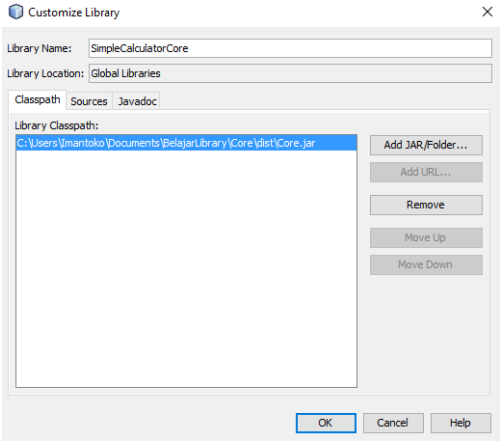
Project baru akan muncul di project browser. Klik kanan pada Libraries yang ada di project baru, kemudian pilih Add Library…



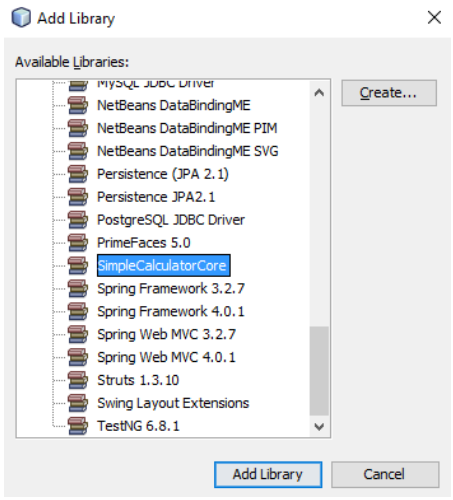
Klik Create Library…
Berikan nama library dan tipe library. Lalu tekan Ok



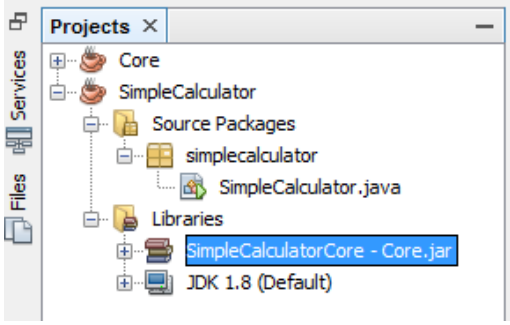
Pada Classpath, klik Add Jar/Folder…
Kemudian arahkan ke lokasi Core.jar yang dibuat sebelumnya. Lakukan hal yang sama pada tab Sources. Jika sudah, tekan OK.



Pilih library yang sudah dibuat dan tekan Add Library

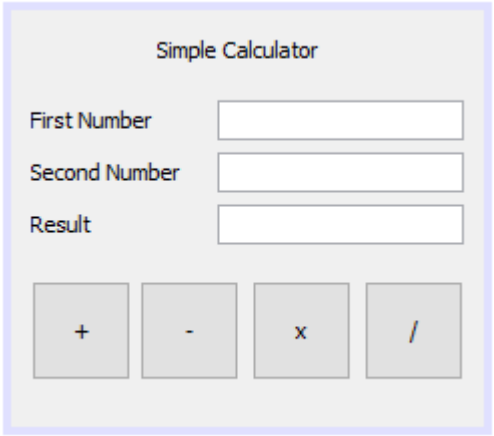


Maka pada project browser akan terlihat seperti gambar berikut



b. Membuat Aplikasi

Buatlah form baru seperti pada gambar berikut



Form terdiri dari tiga TextField, empat Label, dan empat Button.

TextField

- txtFirstNumber, untuk input angka pertama.
- txtSecondNumber, untuk input angka kedua.
- txtResult, untuk menampilkan hasil operasi.

Button

- btnSum, untuk penjumlahan.
- btnSubtract, untuk pengurangan.
- btnMultiple, untuk perkalian.
- btnDivide, untuk pembagian.

Agar form yang dibuat dapat tampil ketika program dijalankan, tuliskan sintaks berikut pada main class project.

```
public class SimpleCalculator {
    public static void main(String[] args) {
    SimpleCalculatorFrame frame = new SimpleCalculatorFrame();
    frame.setVisible(true);
  }
}
```

Tuliskan sintaks berikut pada class form yang sudah dibuat.
Buat variable global untuk pesan error.

```
private final String errorTitle = "Error";
private final String errorMessage = "First Number or Second Number must be
filled with number";
```

Pada constructor class form tuliskan sintaks berikut untuk inisiasi.

```
public SimpleCalculatorFrame() {
   initComponents();
   txtFirstNumber.setHorizontalAlignment(SwingConstants.RIGHT);
   txtSecondNumber.setHorizontalAlignment(SwingConstants.RIGHT);
   txtResult.setHorizontalAlignment(SwingConstants.RIGHT);
   txtResult.setEnabled(false);
}
```

Pada btnSum tuliskan sintaks berikut

```
private void btnSumActionPerformed(java.awt.event.ActionEvent evt) {
   String firstNumber = txtFirstNumber.getText();
   String secondNumber = txtSecondNumber.getText();
   if (firstNumber.equals("") || secondNumber.equals("")) {
      JOptionPane.showMessageDialog(null,
                    errorMessage,
                    errorTitle,
                    JOptionPane.ERROR_MESSAGE);
   }else{
      double first = Double.parseDouble(firstNumber);
      double second = Double.parseDouble(secondNumber);
      double result = Calculator.sum(first, second); // Penjumlahan
      txtResult.setText(String.valueOf(result));
   }
}
```

Pada btnSubtract tuliskan sintaks berikut

```
private void btnSubtractActionPerformed(java.awt.event.ActionEvent evt) {
   String firstNumber = txtFirstNumber.getText();
   String secondNumber = txtSecondNumber.getText();
   if (firstNumber.equals("") || secondNumber.equals("")) {
      JOptionPane.showMessageDialog(null,
```

```
                    errorMessage,
                    errorTitle,
                JOptionPane.ERROR_MESSAGE);
   }else{
      double first = Double.parseDouble(firstNumber);
      double second = Double.parseDouble(secondNumber);
      double result = Calculator.subtract(first, second); // Pengurangan
      txtResult.setText(String.valueOf(result));
   }
}
```

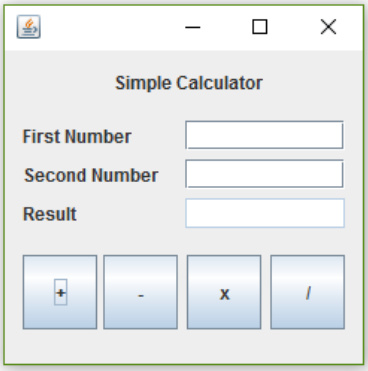
Pada btnDivide tuliskan sintaks berikut

```
private void btnDivideActionPerformed(java.awt.event.ActionEvent evt) {
   String firstNumber = txtFirstNumber.getText();
   String secondNumber = txtSecondNumber.getText();
   if (firstNumber.equals("") || secondNumber.equals("")) {
      JOptionPane.showMessageDialog(null,
                    errorMessage,
                    errorTitle,
                    JOptionPane.ERROR_MESSAGE);
   }else{
      double first = Double.parseDouble(firstNumber);
      double second = Double.parseDouble(secondNumber);
      double result = Calculator.divide(first, second); // Pembagian
      txtResult.setText(String.valueOf(result));
   }
}
```

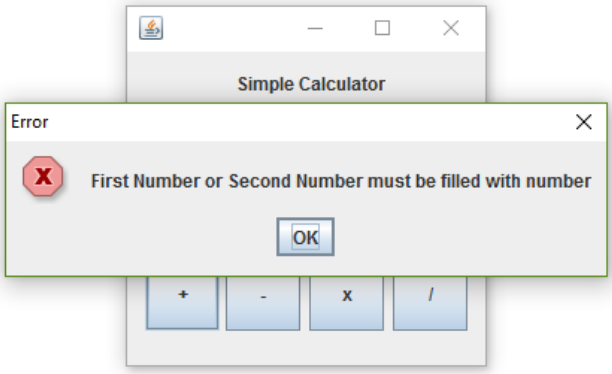
Pada btnMultiple tuliskan sintaks berikut

```
private void btnMultipleActionPerformed(java.awt.event.ActionEvent evt) {
   String firstNumber = txtFirstNumber.getText();
   String secondNumber = txtSecondNumber.getText();
   if (firstNumber.equals("") || secondNumber.equals("")) {
      JOptionPane.showMessageDialog(null,
                    errorMessage,
                    errorTitle,
                    JOptionPane.ERROR_MESSAGE);
   }else{
      double first = Double.parseDouble(firstNumber);
      double second = Double.parseDouble(secondNumber);
      double result = Calculator.multiple(first, second); // Penjumlahan
      txtResult.setText(String.valueOf(result));
   }
}
```

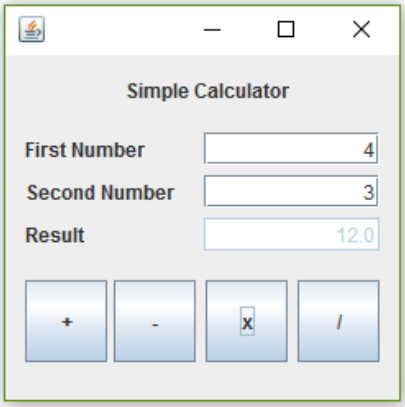
Hasil running aplikasi sebagai berikut



Running awal



Handling error ketika input kosong



Hasil perkalian

BAB 14
Mengakses Port pada Hardware Komputer

14.1. Capaian Pembelajaran

Setelah mempelajari bab ini, mahasiswa diharapkan:
1. Mengetahui cara menerapkan *library* rxtx dalam program
2. Mengetahui cara komunikasi serial pada comm port

14.2. Indikator

1. Mahasiswa mampu mengintegrasikan *library* rxtx dalam program
2. Mahasiswa mampu mengimplementasikan kode program untuk melakukan komunikasi pada comm port.

14.3. Uraian Materi

Sebuah aplikasi yang melibatkan komunikasi dengan *hardware* komputer dapat dilakukan dalam Java, salah satunya dengan menggunakan *library* Java Comm yang bisa Anda unduh dari link berikut ini: http://rxtx.qbang.org/wiki/index.php/Download. Sebagai informasi, jika Anda menggunakan IDE Netbeans, maka sebaiknya Anda mengunduh rxtx versi *binary file*.

A. Mencari COMM Port yang Aktif

Untuk mencari comm port yang sedang aktif menggunakan *library* rxtx, kita bisa menggunakan *method* di bawah ini:

```
1  public static HashSet<CommPortIdentifier> getAvailableSerialPorts() {
2      HashSet<CommPortIdentifier> h = new HashSet<CommPortIdentifier>();
3      Enumeration thePorts = CommPortIdentifier.getPortIdentifiers();
4      while (thePorts.hasMoreElements()) {
5          CommPortIdentifier com = (CommPortIdentifier) thePorts.nextElement();
6          switch (com.getPortType()) {
7          case CommPortIdentifier.PORT_SERIAL:
8              try {
9                  CommPort thePort = com.open("CommUtil", 50);
10                 thePort.close();
11                 h.add(com);
12             } catch (PortInUseException e) {
13                 System.out.println("Port, "  + com.getName() + ", is in use.");
14             } catch (Exception e) {
15                 System.err.println("Failed to open port " +  com.getName());
16                 e.printStackTrace();
17             }
18         }
19     }
20     return h;
21 }
```

B. Mencari Semua COMM Port yang Ada

Untuk mencari semua comm port, baik yang aktif ataupun yang tidak menggunakan *library* rxtx, kita bisa menggunakan *method* di bawah ini:

```
import gnu.io.*;
static void listPorts()
{
    java.util.Enumeration<CommPortIdentifier> portEnum = CommPortIdentifier.getPortIdentifiers();
    while ( portEnum.hasMoreElements() )
    {
        CommPortIdentifier portIdentifier = portEnum.nextElement();
        System.out.println(portIdentifier.getName()  +  " - " +  getPortTypeName(portIdentifier.getPortType()) );
    }
}

static String getPortTypeName ( int portType )
{
    switch ( portType )
    {
        case CommPortIdentifier.PORT_I2C:
            return "I2C";
        case CommPortIdentifier.PORT_PARALLEL:
            return "Parallel";
        case CommPortIdentifier.PORT_RAW:
            return "Raw";
        case CommPortIdentifier.PORT_RS485:
            return "RS485";
        case CommPortIdentifier.PORT_SERIAL:
            return "Serial";
        default:
            return "unknown type";
    }
}
```

C. Komunikasi Serial

Berikut adalah contoh kode untuk melakukan komunikasi seriap pada comm port menggunakan *library* rxtx.

```
import gnu.io.CommPort;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;

import java.io.FileDescriptor;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

public class TwoWaySerialComm
{
    public TwoWaySerialComm()
    {
        super();
    }

    void connect ( String portName ) throws Exception
    {
        CommPortIdentifier portIdentifier = CommPortIdentifier.getPortIdentifier(portName);
        if ( portIdentifier.isCurrentlyOwned() )
        {
            System.out.println("Error: Port is currently in use");
        }
        else
        {
            CommPort commPort = portIdentifier.open(this.getClass().getName(),2000);

            if ( commPort instanceof SerialPort )
            {
                SerialPort serialPort = (SerialPort) commPort;

serialPort.setSerialPortParams(57600,SerialPort.DATABITS_8,SerialPort.STOPBITS_1,SerialPort.PA
RITY_NONE);
```

```
            InputStream in = serialPort.getInputStream();
            OutputStream out = serialPort.getOutputStream();

            (new Thread(new SerialReader(in))).start();
            (new Thread(new SerialWriter(out))).start();

        }
        else
        {
            System.out.println("Error: Only serial ports are handled by this example.");
        }
    }
}

/** */
public static class SerialReader implements Runnable
{
    InputStream in;

    public SerialReader ( InputStream in )
    {
        this.in = in;
    }

    public void run ()
    {
        byte[] buffer = new byte[1024];
        int len = -1;
        try
        {
            while ( ( len = this.in.read(buffer)) > -1 )
            {
                System.out.print(new String(buffer,0,len));
            }
        }
        catch ( IOException e )
        {
            e.printStackTrace();
        }
    }
}

/** */
public static class SerialWriter implements Runnable
{
    OutputStream out;

    public SerialWriter ( OutputStream out )
    {
        this.out = out;
    }

    public void run ()
    {
        try
        {
            int c = 0;
            while ( ( c = System.in.read()) > -1 )
            {
                this.out.write(c);
            }
        }
        catch ( IOException e )
        {
            e.printStackTrace();
        }
    }
}

public static void main ( String[] args )
{
    try
    {
        (new TwoWaySerialComm()).connect("COM3");
    }
```

```
        catch ( Exception e )
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Daftar Pustaka

Mohan, P., Fundamentals of Object Oriented Programming in Java, he University of the West Indie

Koentjoro, E.Y., 2016, Modul Praktikum Pemrograman Berorientasi Objek, STIKOM Surabaya

Khannedy, E. K., 2011. *Belajar Java Dasar*. Bandung. Available online: http://vokasi.uho.ac.id/statistika/assets/download/151227152252Belajar-Java-Dasar.pdf.

Rahardjo, B., Heryanto, I., Haryono, I. 2012. *Mudah Belajar Java: Revisi Kedua*. Bandung: Penerbit Informatika.

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

RESPONSI	Nama Lengkap		
	NIM		
Diperiksa tanggal:		Paraf dosen/ asisten:	

Diterbitkan oleh :
Universitas Teknologi Yogyakarta
Jl. Siliwangi, Jombor, Sleman, Yogyakarta
Email : publikasi@uty.ac.id
Website : uty.ac.id

ISBN 978-623-92626-5-5