Modul Praktikum

Pemrograman Berorientasi Objek

Oleh:

Edo Yonatan Koentjoro, S. Kom

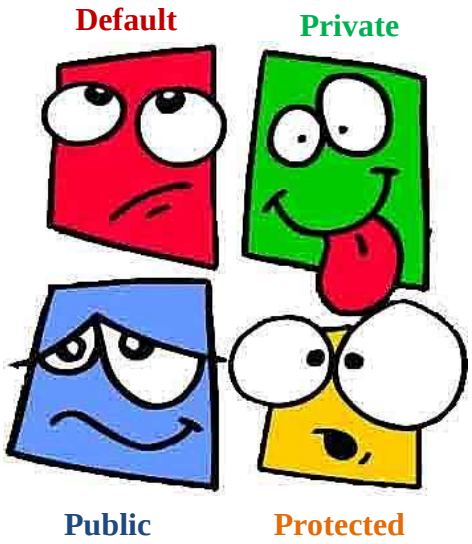
Ada orang lagi ngakses,
yang tersesat di dalam goa
Ini kunci menuju sukses,
dengan belajar dan berdoa
-EdTan-

Laboratorium Komputer STIKOM Surabaya | 2012

Daftar Isi

MODUL 1

PENGENALAN OOP



Tujuan:

Mahasiswa dapat mengenal dan memahami konsep class, objek, dan *access modifier*

Materi:

- ✓ Pengantar
- ✓ Class
- ✓ Atribut
- ✓ Method
- ✓ This
- ✓ *Access Modifier*
- ✓ *Instance of Class*
- ✓ Soal Latihan

Referensi:

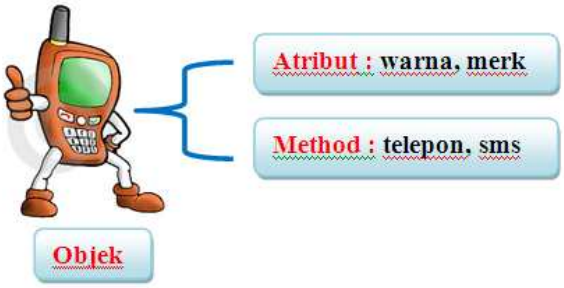
- ❖ Fikri, Rijalul. 2005. *Pemrograman Java*. Yogyakarta: Penerbit Andi
- ❖ Hermawan, Benny. 2004. *Menguasai Java 2 & Object Oriented Programming*. Yogyakarta: Penerbit Andi

1.1 Pengantar

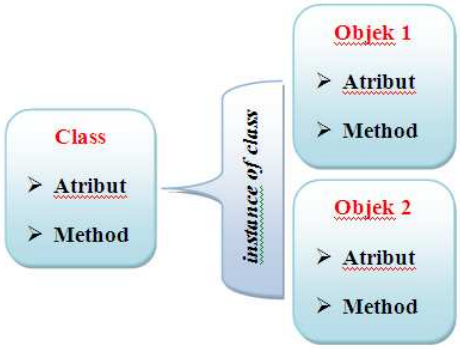
OOP? Apa tuh OOP? Apa sebuah komunitas yang terdiri dari "Orang-Orang Pandai (OOP)"? Agar tidak membuat anda bingung, simak penuturan saya lewat tulisan di bawah ini.

Sebelum kita berkenalan dengan OOP, saya ingin bertanya kepada anda tentang sebuah alat komunikasi bernama ***handphone***. Dan saya yakin bahwa saat ini anda juga membawa gadget tersebut. Sekarang coba anda keluarkan *handphone* anda dan coba perhatikan baik-baik! Bisakah anda menyebutkan ciri-ciri fisik yang terlihat pada *handphone* anda? Dan fungsi apa saja yang dapat digunakan pada *handphone* anda? Jika anda bisa menjawab dua pertanyaan di atas, berarti anda siap untuk membaca paragraf selanjutnya.

Merk dan warna merupakan ciri-ciri *handphone*. Sedangkan kemampuan berkomunikasi lewat telepon maupun SMS adalah fungsi yang dapat digunakan pada *handphone*. Seperti halnya pada OOP. **OOP** (atau biasa dikenal dengan nama ***Object Oriented Programming***) merupakan sebuah pemrograman berorientasi kepada obyek, dimana semua ciri-ciri (**atribut**) dan perilaku (***method***) dibungkus oleh kelas ataupun objek. Dalam modul ini, bahasa pemrograman yang digunakan untuk pembuatan OOP adalah bahasa Java. Pada OOP, *handphone* merupakan sebuah objek. Sedangkan ciri dan fungsi *handphone* merupakan sebuah atribut dan method. Gambar di bawah ini akan menunjukkan skema OOP pada *handphone*.



Objek yang memiliki kesamaan atribut dan method, dapat dikelompokkan menjadi sebuah *Class*. Dan objek-objek yang dibuat dari suatu *class* inilah yang disebut dengan *instance of class*. Untuk *instance of class* akan dijelaskan pada sub bab berikutnya. Berikut adalah gambar hubungan *class* dan objek



1.2 Class

Class merupakan cetak biru (*blue print*) dari objek dimana sebuah *class* menggambarkan ciri-ciri objek secara umum. Struktur pembuatan class, adalah sebagai berikut:

```
1  class Nama_Kelas
2  {
3      //isi dari kelas
4  }
```

Keterangan:

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

1.3 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";**

1.4 Method

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: **setWarna()**, **getWarna()**.

Secara umum, method ada dua macam, yaitu method yang mengembalikan nilai dan method yang tidak mengembalikan nilai. 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 getWarna()**

1.5 This

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.

1.6 Access Modifier

Seperti yang telah diberitahukan di atas, ***Access Modifier*** digunakan untuk untuk memberi batasan hak *class* maupun *method*. Terdapat 4 akses yang tersedia pada java, yakni *default, public, protected, private*. Berikut adalah kemampuan aksesabilitas pada masing-masing *access modifier*.

Tabel 1.1 Perbandingan hak *access modifier*

Aksesabilitas	*private*	*default*	*protected*	*public*
Dari class yang sama	Ya	Ya	Ya	Ya
Dari package yang sama	–	Ya	Ya	Ya
Dari package yang berbeda (subclass)	–	–	Ya	Ya
Dari package yang berbeda (nonsubclass)	–	–	–	Ya

1.7 Instance of Class

Instance of Class merupakan objek yang diinstan atau dibuat dari *class*. Berikut adalah contoh *syntax instance of class*:

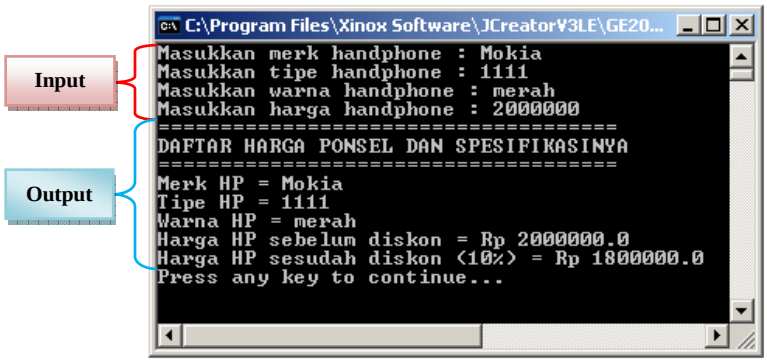
```
[nama_Class] [nama_obyek] = nama_Class (……)
```

Gambar 1.6 *instance of class*

Untuk contoh penggunaannya *instance of class*, dapat anda anda lihat pada soal latihan.

1.8 Soal 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:

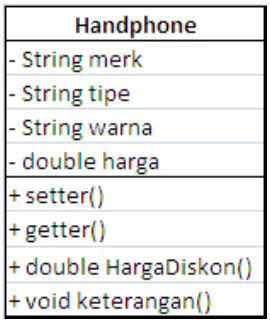
- ✓ Atribut berisi **merk, tipe, warna,** dan **harga**
- ✓ Terdapat method **setter** dan **getter** untuk mengeset dan mengambil nilai dari **merk, tipe, warna,** dan **harga**
- ✓ Terdapat method **HargaDiskon()** untuk menghitung harga handphone sesudah diskon. Diskon yang diperoleh adalah 10%
-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

Jawabannya adalah…

Setelah anda membaca soal tersebut dengan baik dan seksama, langkah pertama yang harus anda lakukan adalah menganalisa soal tersebut dan membuat skema diagram dari soal tersebut. Skema ini nantinya akan membantu anda pembuatan program. Berikut adalah skema diagramnya.

➢ 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

```
1  class Handphone
2  {
3      //deklarasi
4
5      //setter
6
7      //getter
8
9      //method tambahan
10
11 }
```

Setelah anda membuat class Handphone, simpan file tersebut dengan nama **Handphone.java**. Di dalam class Handphone, 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 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.

```
1  class Handphone
2  {
3      //deklarasi
4      private String merk, tipe, warna;
5      private double harga;
6
7      //setter
8      public void setMerk(String merk)
9      {
10         this.merk=merk;
11     }
12     public void setTipe(String tipe)
13     {
14         this.tipe=tipe;
15     }
16     public void setWarna(String colour)
17     {
18         warna=colour;
19     }
20     public void setHarga(double harga)
21     {
22         this.harga=harga;
23     }
24
25     //getter
26
27     //method tambahan
28
29 }
```

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.

```
1  class Handphone
2  {
3      //deklarasi
4      private String merk, tipe, warna;
5      private double harga;
6
7      //setter
8      public void setMerk(String merk)
9      {
10         this.merk=merk;
11     }
12     public void setTipe(String tipe)
13     {
14         this.tipe=tipe;
15     }
16     public void setWarna(String colour)
17     {
18         warna=colour;
19     }
20     public void setHarga(double harga)
21     {
22         this.harga=harga;
23     }
24
25     //getter
26
27     //method tambahan
28
29 }
```

Coba perhatikan kembali script yang telah anda buat seperti gambar di atas. Di dalam pembuatan method setter terdapat keyword ***this***. Penggunaan keyword ***this*** akan mengacu kepada variabel yang dideklarasikan pada class

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.

```
1  class Handphone
2  {
3      //deklarasi
4      private String merk, tipe, warna;
5      private double harga;
6
7      //setter
8      public void setMerk(String merk)...
12     public void setTipe(String tipe)...
16     public void setWarna(String colour)...
20     public void setHarga(double harga)...
24
25     //getter
26     public String getMerk()
27     {
28         return merk;
29     }
30     public String getTipe()
31     {
32         return tipe;
33     }
34     public String getWarna()
35     {
36         return warna;
37     }
38     public double getHarga()
39     {
40         return harga;
41     }
42
43
44     //method tambahan
45
46 }
```

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

e. 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

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

        //input

        //output

    }
}
```

Setelah anda membuat class Utama, simpan file tersebut dengan nama **Utama.java**. Di dalam class inilah, program anda akan dijalankan.

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.

```
1  class Utama
2  {
3      public static void main (String [] args)
4      {
5          //instance of class
6          Handphone hp = new Handphone();
7
8          //input
9
10         //output
11
12     }
13 }
```

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. 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
13         //output
14
15     }
16 }
```

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.

MODUL 2

Constructor

Tujuan:

Mahasiswa dapat mengenal dan memahami konsep constructor dan overloading constructor

Materi:

- ✓ Pengantar
- ✓ Constructor
- ✓ Overloading constructor
- ✓ Soal Latihan

Referensi:

- ❖ Fikri, Rijalul. 2005. *Pemrograman Java*. Yogyakarta: Penerbit Andi
- ❖ Hermawan, Benny. 2004. *Menguasai Java 2 & Object Oriented Programming*. Yogyakarta: Penerbit Andi

2.1 Pengantar

Kalau anda melihat gambar yang ada di cover pada modul 2, anda mungkin berpikiran bahwa constructor serupa dengan seorang yang membangun sebuah gedung, apartemen, maupun bangunan-bangunan tinggi. Memang sich, jawaban anda tidak salah untuk yang satu ini. Tapi jawaban yang anda kemukakan akan keliru ketika anda melihat constructor dari segi pemrograman. Lalu sebenarnya apa sich constructor itu? Baiklah akan saya jelaskan dengan bahasa yang mudah anda pahami melalui sebuah ilustrasi.

Setiap manusia pasti memiliki nama bukan? Coba anda bertanya kepada diri anda anda, “Siapakah aku? (atau biasa dikenal dengan *Who am I?)*?”. 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 anda tuh banyak dipakai juga pada orang lain? Bahkan nama lengkap saya sendiri pun juga ada yang menggunakannya meskipun ejaannya berbeda. (Berarti saya terkenal donk :p). 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. Untuk lebih jelas tentang konsep Constructor dan penerapannya, silahkan anda simak penjelasannya di bawah ini.

2.2 Constructor

Constructor adalah method yang berfungsi untuk menginisialisasi variabel-variabel 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.

```
1  class Nama_Kelas
2  {
3      Nama_Kelas()
4      {
5          //isi constructor
6      }
7
8      // isi dari kelas
9  }
```

Contoh penggunaan constructor:

```
1  class Login
2  {
3      Login()
4      {
5          //isi constructor
6      }
7
8      // isi dari kelas
9  }
```


2.3 Karakteristik Constructor

Berikut ini adalah beberapa karakteristik yang dimiliki oleh constructor:

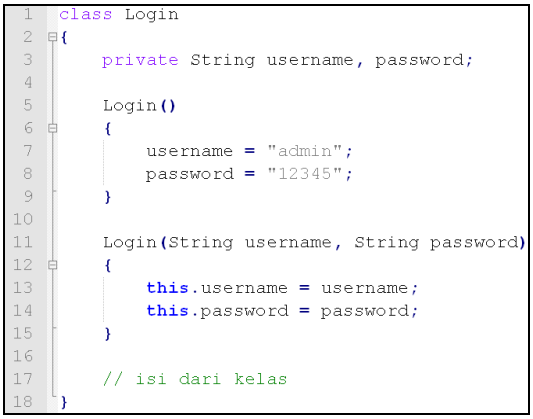
1. Method constructor harus memiliki nama yang sama dengan nama class
2. Tidak mengembalikan suatu nilai (tidak ada keyword *return*)
3. Satu kelas memiliki lebih dari constructor (*overloading constructor*)
4. Dapat ditambah *access modifier* public, private, protected maupun default
5. Suatu constructor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

2.4 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.

```
1  class Nama_Kelas
2  {
3      Nama_Kelas()
4      {
5          //isi constructor
6      }
7
8      Nama_Kelas(parameter)
9      {
10         //isi constructor
11     }
12
13     // isi dari kelas
14 }
```

Contoh penggunaan constructor:



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

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

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

    // isi dari kelas
}
```

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();
```

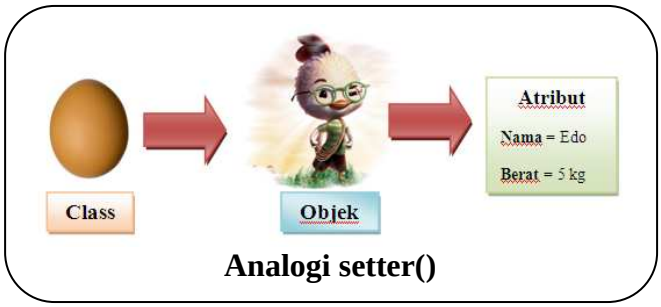
Pembuatan objek di atas akan memberikan nilai set default username dan password berupa admin dan 12345. Sedangkan jika anda ingin membuat user sendiri (misal: username=edo, password=pb0) melalui *instance of class* dengan method Login menggunakan parameter adalah sebagai berikut:

```
Login user1 = new Login("edo", "pb0");
```

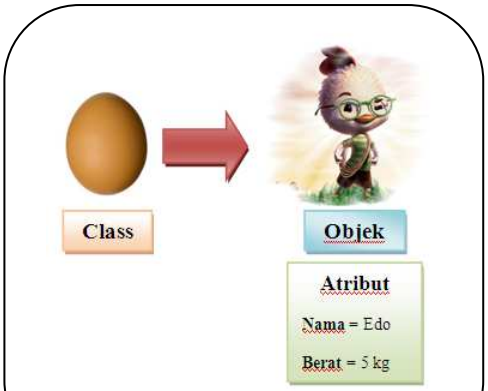
2.5 Perbedaan menggunakan Constructor dengan method Setter()

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()



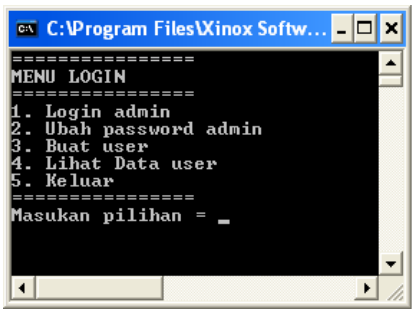
Analogi constructor

2.6 Soal Latihan

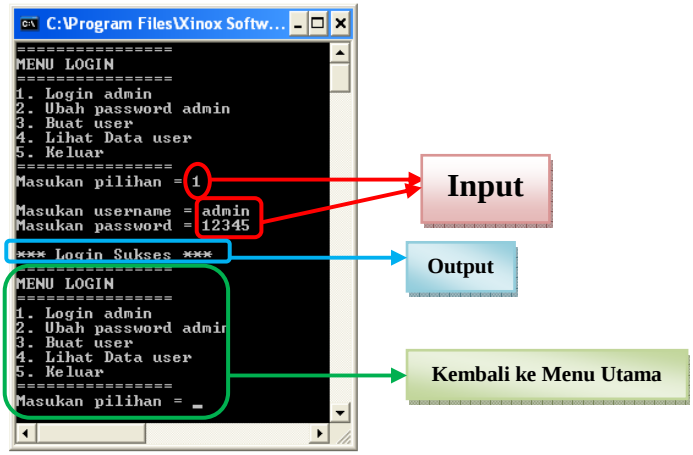
Berdasarkan contoh di atas tentang login, buatlah 2 buah class yang terdiri dari class Login dan class utama. Class Login harus memiliki beberapa ketentuan sebagai berikut:

- ✓ 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**

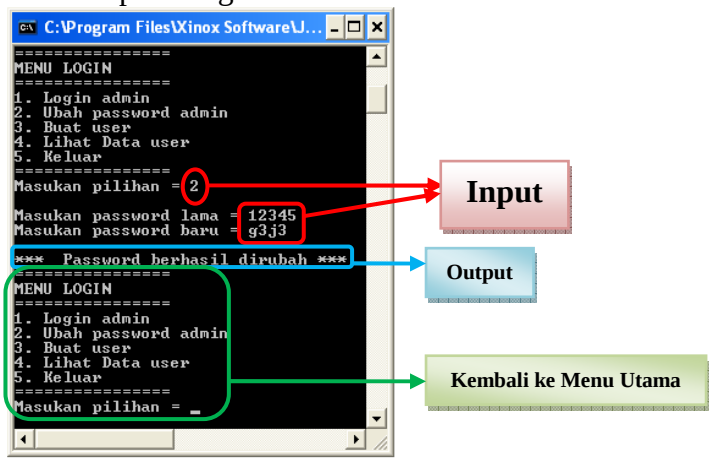
Sedangkan pada Class utama digunakan untuk memanggil class Login. Ketika class utama dijalankan, hasilnya akan tampak seperti di bawah ini:



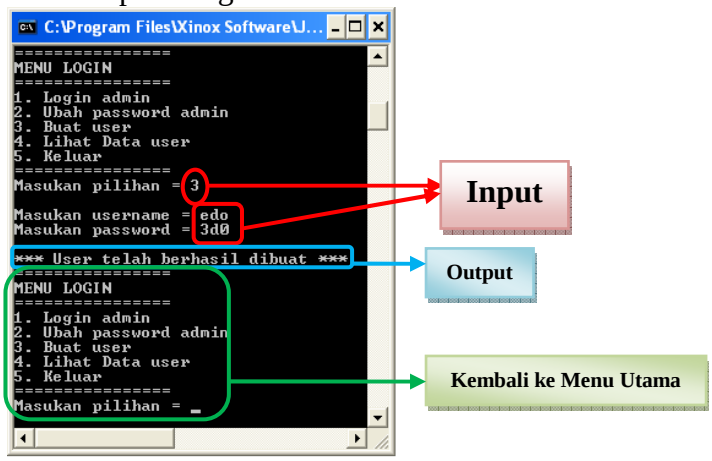
❖ Jika pilihan = 1, maka akan tampil sebagai berikut:



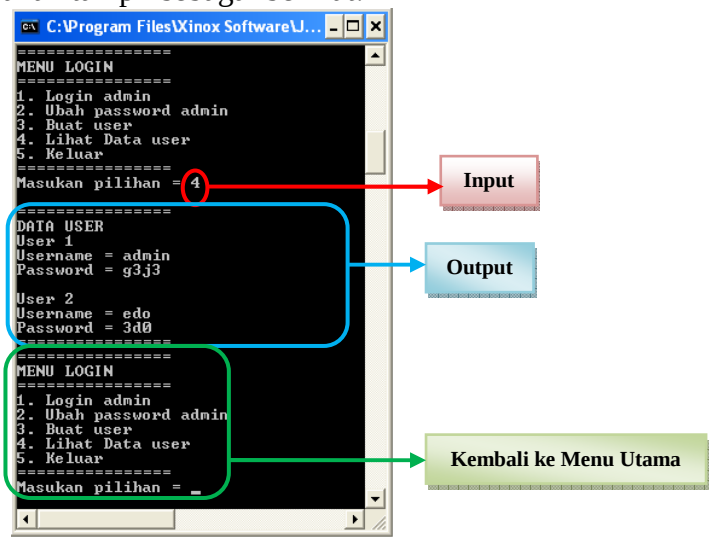
❖ Jika pilihan = 2, maka akan tampil sebagai berikut:



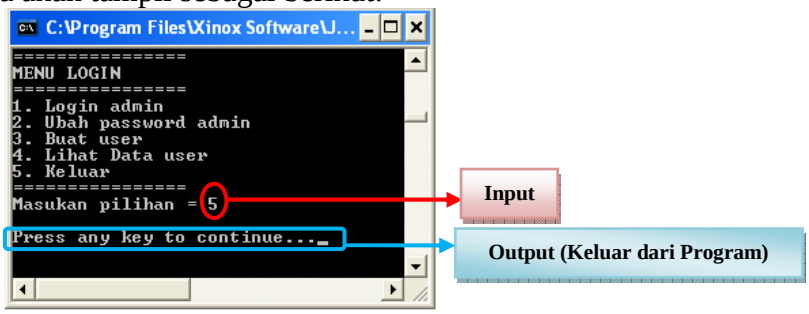
❖ Jika pilihan = 3, maka akan tampil sebagai berikut:



❖ Jika pilihan = 4, maka akan tampil sebagai berikut:



❖ Jika pilihan = 5, maka akan tampil sebagai berikut:



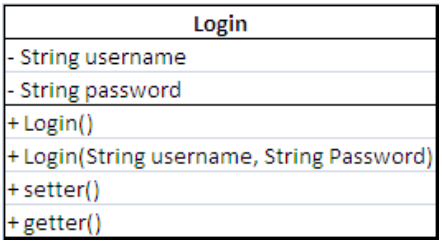
Jawabannya adalah…

Setelah anda membaca soal tersebut dengan baik dan seksama, langkah pertama yang harus anda lakukan adalah menganalisa soal tersebut dan membuat

skema diagram dari soal tersebut. Skema ini nantinya akan membantu anda pembuatan program. Berikut adalah skema diagramnya.

➢ 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 Login (ketikkan script berikut)

a. Membuat kerangka class Login

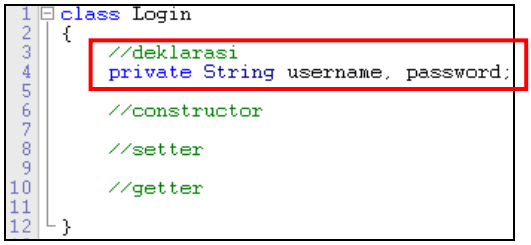
```
1  class Login
2  {
3      //deklarasi
4
5      //constructor
6
7      //setter
8
9      //getter
10
11 }
```

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

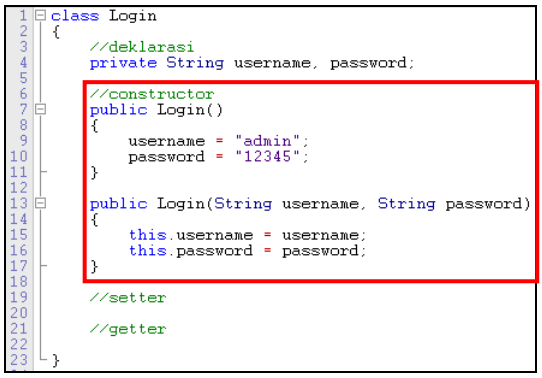
Setelah kita membuat kerangka class, maka diperlukan pendeklarasian variabel yang nantinya digunakan sebagai tempat menyimpan data yang

bersifat sementara (*temporary*). Gambar di bawah ini menunjukkan pendeklarasian variabel.



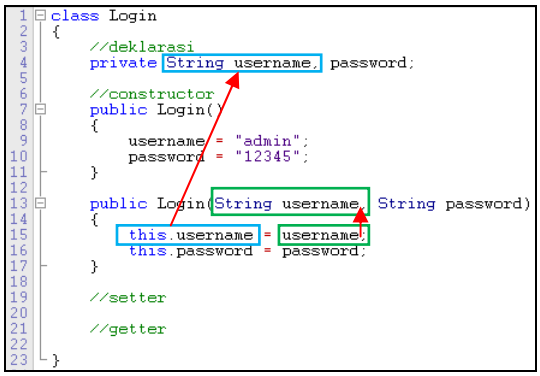
c. Mendeklarasi constructor

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.



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).



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;
```

h. Mengisi Case 3 (Buat User)

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

- Line 82-85: berisi permintaan inputan username dan password yang nantinya akan diisi oleh user
- Line 87: merupakan proses mentransfer data username pada variabel **create_user** dan password pada **create_password** ke dalam constructor Login. Setelah anda membuat username dan password baru, program akan mencetak tulisan ***** User telah berhasil dibuat ***** (line 91). 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**.

```
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;
```

i. Mengisi Case 4 (Lihat Data User)

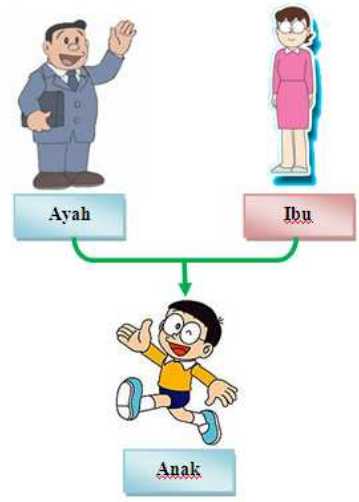
Ketika user memilih inputan menu no. 4, maka dilakukan beberapa proses sebagai berikut:

- Line 102-103: 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**.
- Line 108-109: mencetak isi data username dan password yang sudah anda buat pada **case 3** pada variabel **user 2**.

```
94                  //jika pilih = 4
95                  case 4:
96                      //isi pilihan bernilai 4 ketika dijalankan
97
98                      System.out.println("================");
99                      System.out.println("DATA USER");
100
101                     System.out.println("User 1");
102                     System.out.println("Username = "+user1.getUsername());
103                     System.out.println("Password = "+user1.getPassword());
104
105                     System.out.println();
106
107                     System.out.println("User 2");
108                     System.out.println("Username = "+user2.getUsername());
109                     System.out.println("Password = "+user2.getPassword());
110
111                     System.out.println("================");
112                     break;
```


MODUL 3

Inheritance



Tujuan:

Mahasiswa dapat mengenal dan memahami konsep inheritance dan cara menerapkan inheritance dengan constructor

Materi:

- ✓ Pengantar
- ✓ Inheritance
- ✓ Manfaat Penggunaan Inherintance
- ✓ Keyword "*super*"
- ✓ Soal Latihan

Referensi:

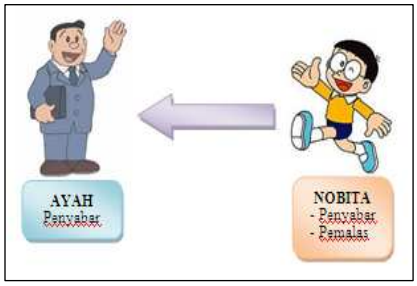
- ❖ Fikri, Rijalul. 2005. *Pemrograman Java*. Yogyakarta: Penerbit Andi
- ❖ Hermawan, Benny. 2004. *Menguasai Java 2 & Object Oriented Programming*. Yogyakarta: Penerbit Andi
- ❖ Purnama, Rangsang. 2003. *Tuntunan Pemrograman Java Jilid 2*. Surabaya: Prestasi Pustaka Publisher

3.1 Pengantar

"Duluan mana, telur atau ayam dulu?". Pertanyaan tersebut pasti *familiar* di telinga anda. Tapi kali ini, kita tidak membahas hal itu. "Trus kalau 'gak dibahas, buat apa ditulis di modul ini?". Jawabannya adalah sebagai berikut.

Sebuah telur yang sudah beberapa bulan dierami oleh induknya, maka akan menetas menjadi anak ayam. Anak ayam yang dirawat dan dipelihara baik-baik, pastilah akan sama seperti ibunya.

Demikian juga dalam kehidupan manusia. Kehidupan kita semua pasti memiliki karakter yang 'hampir' sama dengan orang tua kita. "Mengapa harus disebut 'hampir'?" Karena tidak mungkin kita mirip 100% dengan orang tua kita. Selain bentuk fisik yang memiliki keseragaman yang hampir sama, tentunya kita juga memiliki beberapa karakter yang hampir sama dengan kedua orang tua kita. Misal: Ayah Nobita adalah orang yang penyabar. Ada kemungkinan Nobita juga memiliki sikap penyabar. Tapi di balik semua itu, pasti Nobita punya sifat yang tidak dimiliki oleh ayahnya, yaitu sifat pemalas.



Dalam pemrograman, sifat orang tua yang diturunkan kepada anaknya dikenal dengan nama pewarisan (*inheritance*). Untuk lebih jelas mengenai konsep *inheritance*, saya akan menjelaskannya pada sub bab berikutnya.

3.2 Inheritance

Inheritance merupakan proses pewarisan data dan method dari suatu class yang telah ada kepada class baru. Class yang mewariskan disebut dengan **kelas super (super class)**, sedangkan kelas yang mendapat warisan tersebut atau class

yang diwariskan disebut dengan **subkelas (sub class)**. Ibarat contoh di atas, berarti ayah Nobita berperan sebagai super class, sedangkan Nobita berperan sebagai sub class.

Untuk menggunakan inheritance, maka dibutuhkan keyword **extends**. Cara penulisannya adalah sebagai berikut:

```
1  class namaSubClass extends namaSuperClass
2  {
3      //definisi kelas
4  }
```

Contoh:

Misalkan kita memiliki beberapa software di komputer. Software tersebut beraneka ragam. Ada software game, software edukasi, dan masih banyak lagi. Biarpun berbeda jenisnya, software tersebut pasti memiliki kesamaan dengan software sejenisnya. Dengan kata lain, apabila saya melihat dari segi pemrograman, maka **class Software** adalah **superclass**, sedengakan **class Game** adalah turunan dari class Software (**subclass**). Jadi penulisan script pada class Game adalah sebagai berikut:

```
1  class Game extends Software
2  {
3      //definisi kelas Game
4  }
```

3.3 Manfaat Penggunaan Inherintance

Berikut ini adalah beberapa menfaat apabila anda menggunakan konsep inheritance:

a. Bersifat Reusable

Bayangkan saja apabila anda memerlukan beberapa kelas yang berasal dari basis yang sama (data dan method yang sama), namun pada masing-masing kelas akan ditambahkan data atau method tambahan. Dengan menggunakan inheritance, anda cukup mengambil data atau method pada class induknya dan memberikan beberapa tambahan data atau method pada class anaknya apabila diperlukan.

b. Kemudahan dalam memanage kelas yang dimiliki data dan method yang sama

Bila anda ingin memodifikasi suatu data atau method pada semua subclass, anda tidak perlu melakukan perubahan pada masing-masing kelas pada subclass. Anda cukup melakukan perubahan data atau method pada kelas super (superclass) yang mewarisi subclass tersebut.

3.4 Keyword "*super*"

Keyword **super** digunakan oleh subclass untuk memanggil constructor aatau method yang ada pada superclassnya. Berikut adalah cara penulisan "super" pada subclass untuk memanggil constructor pada superclass.

super () atau **super (parameter)**

Sedangkan, cara penulisan "super" pada subclass untuk memanggil method pada superclass adalah sebagai berikut:

super.namaMethod() atau **super.namaMethod(parameter)**

Untuk contoh penggunaan keyword "super" akan dijelaskan pada contoh latihan.

3.5 Soal Latihan

Berdasarkan contoh di atas tentang software, buatlah 3 buah class yang terdiri dari class Software, class Game dan class Utama.

Class Software harus memiliki beberapa ketentuan sebagai berikut:

- ✓ Atribut berisi **kode, nama,** dan **lisensi**
- ✓ Terdapat 2 buah constructor Software
  - o Constructor pertama tidak memiliki parameter dan tidak ada isinya (kosongan)
  - o Sedangkan constructor kedua memiliki parameter untuk mengeset nilai **kode, nama,** dan **lisensi** (freeware/shareware) berdasarkan inputan user
- ✓ Terdapat method **setter** dan **getter** untuk mengeset/merubah dan mengambil nilai dari **kode, nama,** dan **lisensi** (freeware/shareware)

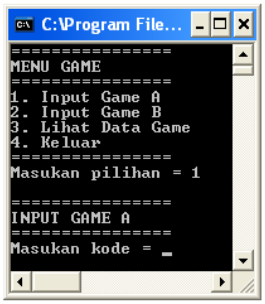
Class Game harus memiliki beberapa ketentuan sebagai berikut:

- ✓ Atribut berisi **jenis** dan **tipe** dan diberi nilai default “Unknown”
- ✓ Terdapat 2 buah constructor Game
  - o Constructor pertama memiliki parameter untuk mengeset nilai **kode, nama,** dan **lisensi** (freeware/shareware) berdasarkan inputan user. Data **kode, nama,** dan **lisensi** (freeware/shareware) diambil dari Constructor kedua pada class Software (gunakan keyword ***super***)
  - o Sedangkan constructor kedua memiliki parameter untuk mengeset nilai **kode, nama, lisensi** (freeware/shareware)**, jenis** (offline/online)**,** dan **tipe** (action/arcade/adventure/sport/puzzle) berdasarkan inputan user. Data **kode, nama,** dan **lisensi**

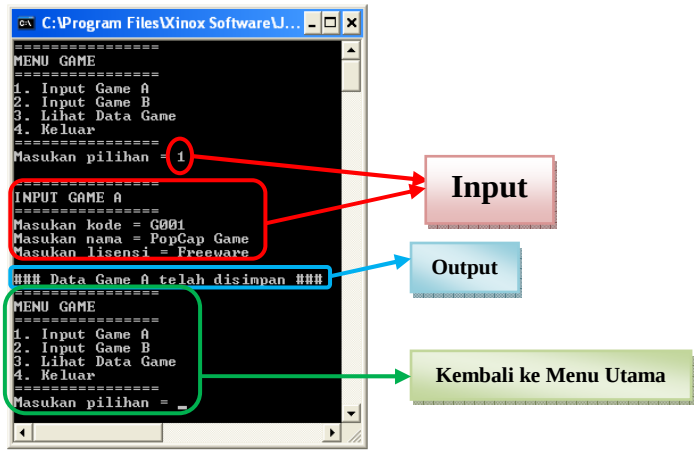
(freeware/shareware) diambil dari setter pada class Software (gunakan keyword ***super***)

- ✓ Terdapat method **getter** untuk mengambil nilai dari variabel **kode, nama,** dan **lisensi** (freeware/shareware) pada class Software serta mengambil nilai dari variabel **jenis** (offline/online) dan **tipe** (action/arcade/ adventure/sport/puzzle)

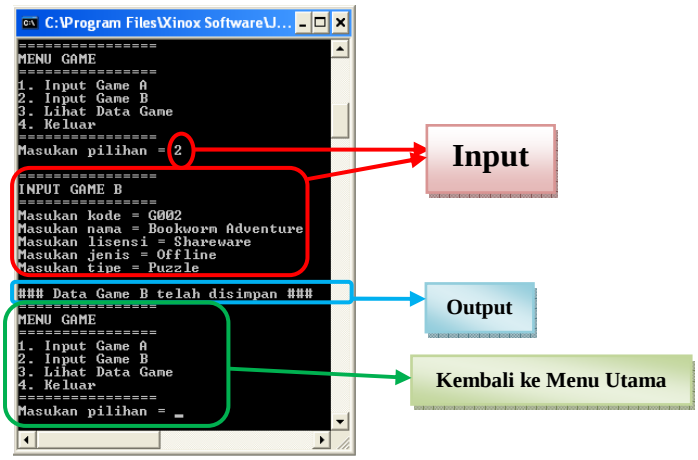
Sedangkan pada Class utama digunakan untuk memanggil class Game. Ketika class utama dijalankan, hasilnya akan tampak seperti di bawah ini:



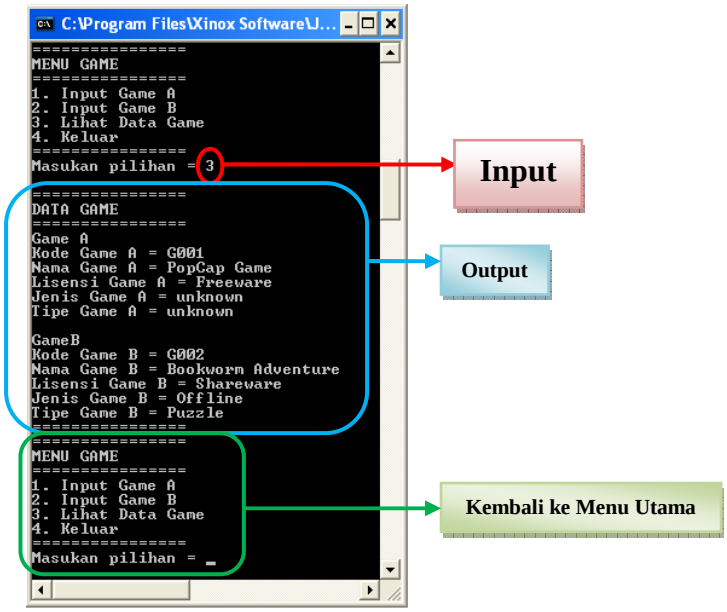
❖ Jika pilihan = 1, maka akan tampil sebagai berikut:



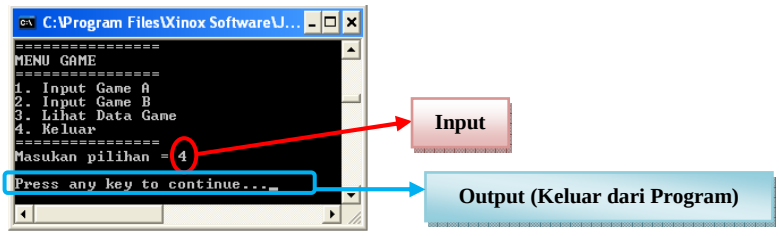
❖ Jika pilihan = 2, maka akan tampil sebagai berikut:



❖ Jika pilihan = 3, maka akan tampil sebagai berikut:



❖ Jika pilihan = 4, maka akan tampil sebagai berikut:



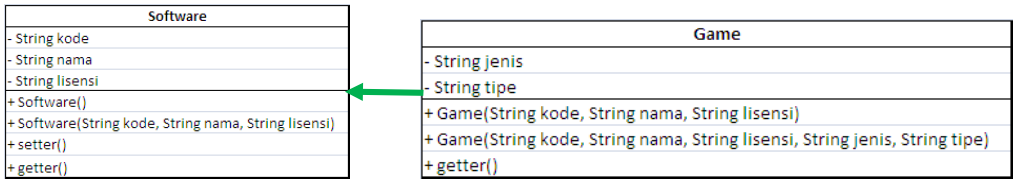
Jawabannya adalah…

Setelah anda membaca soal tersebut dengan baik dan seksama, langkah pertama yang harus anda lakukan adalah menganalisa soal tersebut dan membuat

skema diagram dari soal tersebut. Skema ini nantinya akan membantu anda pembuatan program. Berikut adalah skema diagramnya.

➢ 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.



Anak panah "←" menggambarkan konsep inherintance, dimana class Game merupakan turunan dari class Software. Sehingga variabel kode, nama, dan lisensi pada class Software tidak perlu dideklarasikan ulang.

➢ Langkah 2: class Software (ketikkan script berikut)

a. Membuat kerangka class Software

```
1  class Software
2  {
3      //deklarasi variabel
4
5      //constructor
6
7      //setter
8
9
10     //getter
11
12 }
```

Setelah anda membuat class Software, simpan file tersebut dengan nama **Software.java**. Di dalam class Login, saya juga menyediakan tempat untuk mendeklarasikan variabel, constructor, 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 Software
2  {
3      //deklarasi variabel
4      private String kode, nama, lisensi;
5
6      //constructor
7
8      //setter
9
10
11     //getter
12
13 }
```

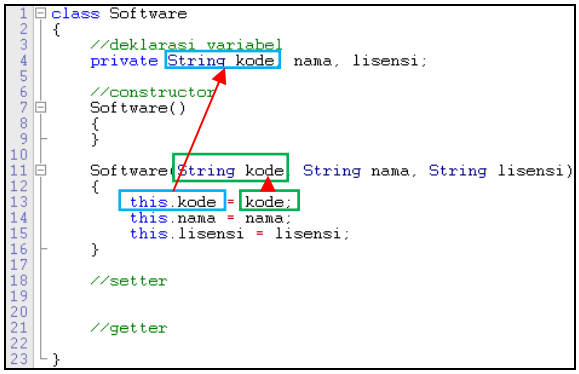
c. Mendeklarasi constructor

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

```
1  class Software
2  {
3      //deklarasi variabel
4      private String kode, nama, lisensi;
5
6      //constructor
7      Software()
8      {
9      }
10
11     Software(String kode, String nama, String lisensi
12     {
13         this.kode = kode;
14         this.nama = nama;
15         this.lisensi = lisensi;
16     }
17
18     //setter
19
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 Software dengan parameter, anda juga dapat menggunakan keyword ***this***. Penggunaan keyword ***this*** akan mengacu kepada variabel yang dideklarasikan pada class Software (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 Software (untuk lebih jelas mengenai keyword ***this***, anda dapat melihat modul 1).



d. Membuat method setter

Setelah membuat variabel yang dibutuhkan pada class Software, langkah selanjutnya anda buat method setter untuk masing-masing variabel. Hal ini bertujuan untuk mengeset atau merubah nilai variabel kode, nama dan lisensi apabila diperlukan sesuai dengan constructor kedua pada Class Game. 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 Software
{
    //deklarasi variabel
    private String kode, nama, lisensi;

    //constructor
    Software()
    {
    }

    Software(String kode, String nama, String lisensi)
    {
        this.kode = kode;
        this.nama = nama;
        this.lisensi = lisensi;
    }

    //setter
    public void setKode(String k)
    {
        kode = k;
    }
    public void setNama(String n)
    {
        nama = n;
    }
    public void setLisensi(String l)
    {
        lisensi = l;
    }

    //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 Software yang nantinya akan kita gunakan ke dalam class Game. 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 Software
{
    //deklarasi variabel
    private String kode, nama, lisensi;

    //constructor
    Software()...

    Software(String kode, String nama, String lisensi)...

    //setter
    public void setKode(String k)...
    public void setNama(String n)...
    public void setLisensi(String l)...

    //getter
    public String getKode()
    {
        return kode;
    }

    public String getNama()
    {
        return nama;
    }

    public String getLisensi()
    {
        return lisensi;
    }
}
```

➢ Langkah 2: class Game (ketikkan script berikut)

a. Membuat kerangka class Game

```
1  class Game extends Software
2  {
3      //deklarasi variabel
4
5      //constructor
6
7      //getter
8
9  }
```

Setelah anda membuat class Login, simpan file tersebut dengan nama G**ame.java**. Di dalam class Game, saya juga menyediakan tempat untuk mendeklarasikan variabel, constructor dan getter. Penggunaan **extends** menunjukkan bahwa class game merupakan turunan dari class Software

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.

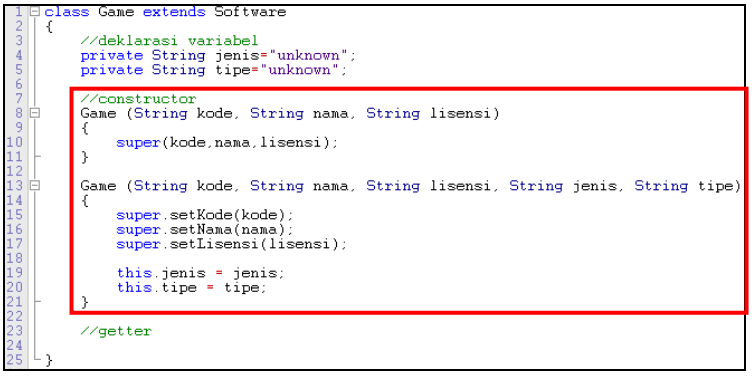
```
1  class Game extends Software
2  {
3      //deklarasi variabel
4      private String jenis="unknown";
5      private String tipe="unknown";
6
7      //constructor
8
9      //getter
10
11 }
```

Perlu diketahui pula, bahwa dalam pembuatan class Game tidak diperlukan pendeklarasian variabel kode, nama, dan lisensi.

c. Mendeklarasi constructor

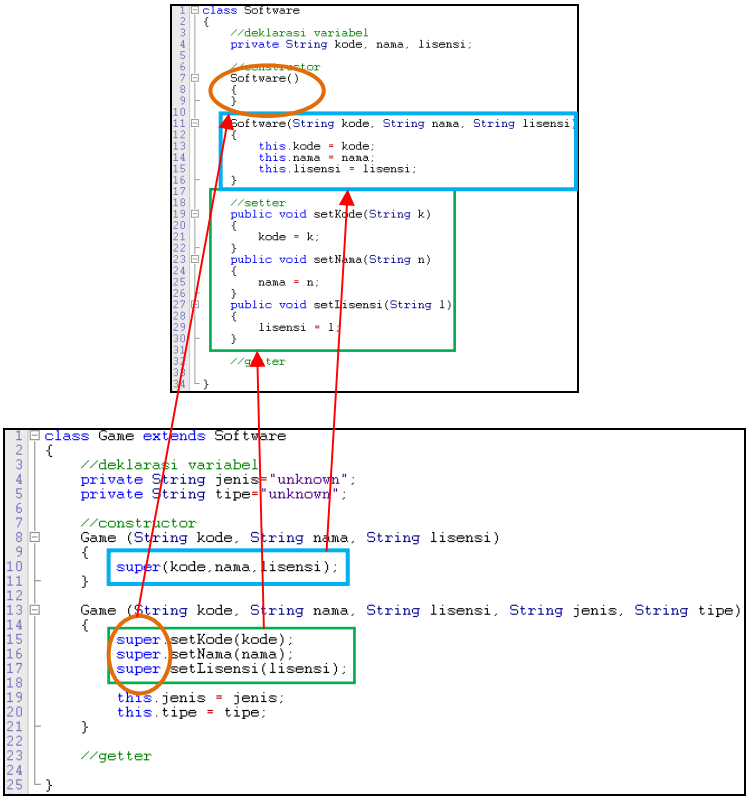
Setelah membuat variabel yang dibutuhkan pada class Game, langkah selanjutnya anda membuat constructor Game. Constructor ini nantinya akan

digunakan dalam class Utama. Gambar di bawah ini menunjukkan deklarasi constructor.



```
class Game extends Software
{
    //deklarasi variabel
    private String jenis="unknown";
    private String tipe="unknown";

    //constructor
    Game (String kode, String nama, String lisensi)
    {
        super(kode,nama,lisensi);
    }

    Game (String kode, String nama, String lisensi, String jenis, String tipe)
    {
        super.setKode(kode);
        super.setNama(nama);
        super.setLisensi(lisensi);

        this.jenis = jenis;
        this.tipe = tipe;
    }

    //getter

}
```

Coba perhatikan kembali script yang telah anda buat seperti gambar di atas. Pada constructor pertama, terdapat keyword "***super***". Keyword ini akan memanggil constructor kedua (sesuai isi parameter) pada class induknya, yakni class Software (lihat *script* yang diberi kotak berwarna biru pada gambar di bawah).



```
class Software
{
    //deklarasi variabel
    private String kode, nama, lisensi;

    //constructor
    Software()
    {
    }

    Software(String kode, String nama, String lisensi)
    {
        this.kode = kode;
        this.nama = nama;
        this.lisensi = lisensi;
    }

    //setter
    public void setKode(String k)
    {
        kode = k;
    }
    public void setNama(String n)
    {
        nama = n;
    }
    public void setLisensi(String l)
    {
        lisensi = l;
    }

    //getter
}
```

```
class Game extends Software
{
    //deklarasi variabel
    private String jenis="unknown";
    private String tipe="unknown";

    //constructor
    Game (String kode, String nama, String lisensi)
    {
        super(kode,nama,lisensi);
    }

    Game (String kode, String nama, String lisensi, String jenis, String tipe)
    {
        super.setKode(kode);
        super.setNama(nama);
        super.setLisensi(lisensi);

        this.jenis = jenis;
        this.tipe = tipe;
    }

    //getter

}
```

Sedangkan jika anda ingin memanggil setter/getter pada class induknya, anda dapat menggunakan keyword "***super***" yang kemudian dilanjutkan dengan nama method yang dipanggil seperti constructor kedua pada class Game (lihat *script* yang diberi kotak berwarna biru pada gambar di atas). Artinya penggunaan keyword "***super***" akan mengarah kepada constructor class induknya (lihat *script* yang diberi kotak berwarna oranye pada gambar di atas).

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 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 tersebut maupun di kelas induknya. Gambar di bawah ini menunjukkan deklarasi getter.

```
1  class Game extends Software
2  {
3      //deklarasi variabel
4      private String jenis="unknown";
5      private String tipe="unknown";
6
7      //constructor
8      Game (String kode, String nama, String lisensi)
12
13     Game (String kode, String nama, String lisensi, String jenis, String tipe)
22
23     //getter
24     public String getKode()
25     {
26         return super.getKode();
27     }
28
29     public String getNama()
30     {
31         return super.getNama();
32     }
33
34     public String getLisensi()
35     {
36         return super.getLisensi();
37     }
38
39     public String getJenis()
40     {
41         return jenis;
42     }
43     public String getTipe()
44     {
45         return tipe;
46     }
47 }
```


Seperti halnya penggunaan Constructor kedua pada class Game yang menggunakan super.[nama_method], maka untuk method getKode(), getNama(), dan getLisensi() menggunakan keywoard "***super***" dikarenakan tidak dideklarasikan pada class Game

- **Langkah 3: class Utama (ketikkan script berikut)**

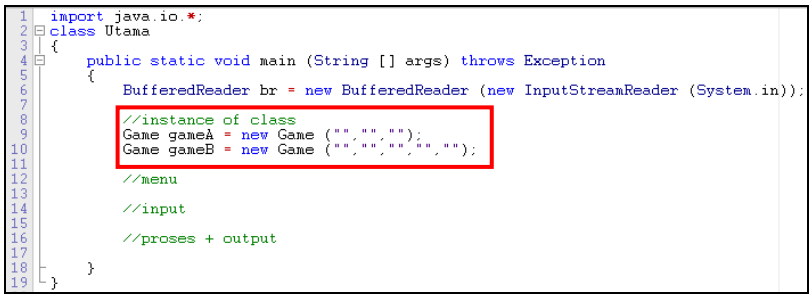
a. Membuat kerangka class Utama

```
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 **gameA** dan **gameB**, dimana **gameA** menggunakan constructor pertama pada class Game dan **gameB** menggunakan constructor kedua pada class Game.



```
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          Game gameA = new Game ("","","");
10         Game gameB = new Game ("","","","","");
11
12         //menu
13
14         //input
15
16         //proses + output
17
18     }
19 }
```

Coba perhatikan kembali pembuatan *instance of class*. Pada *line 9*, objek "gameA" yang telah terbentuk akan mereferens ke constructor pertama pada class Game. Sedangkan pada *line 10*, kita mendeklarasikan objek "gameB" yang akan mereferens ke constructor kedua pada class Game, dimana kedua variable tersebut (gameA dan gameB) masih belum diketahui nilainya. Karena tipe data username dan password bertipe *String*, maka saya menggunakan tanda petik ganda (" ") untuk memberi nilai awal berupa kosong.

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          Game gameA = new Game ("","","");
10         Game gameB = new Game ("","","","","");
11
12         //menu
13         while(true)
14         {
15             //menu
16             System.out.println("================");
17             System.out.println("MENU GAME");
18             System.out.println("================");
19             System.out.println("1. Input Game A");
20             System.out.println("2. Input Game B");
21             System.out.println("3. Lihat Data Game");
22             System.out.println("4. Keluar");
23             System.out.println("================");
24
25             //input
26
27             //proses + output
28
29         }
30     }
31 }
```

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 (e)**.

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          Game gameA = new Game ("","","");
10         Game gameB = new Game ("","","","","");
11
12         //menu
13         while(true)
14         {
15             //menu
16             System.out.println("================");
17             System.out.println("MENU GAME");
18             System.out.println("================");
19             System.out.println("1. Input Game A");
20             System.out.println("2. Input Game B");
21             System.out.println("3. Lihat Data Game");
22             System.out.println("4. 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              System.out.println();
30
31              //proses + output
32              switch(pilih)
33              {
34                  //jika pilih = 1
35                  case 1:
36                      //isi pilihan bernilai 1 ketika dijalankan
37                      break;
38
39                  //jika pilih = 2
40                  case 2:
41                      //isi pilihan bernilai 1 ketika dijalankan
42                      break;
43
44                  //jika pilih = 3
45                  case 3:
46                      //isi pilihan bernilai 3 ketika dijalankan
47                      break;
48
49                  //jika nilai pilih yang dimasukkan bukan 1,2 atau 3, maka program akan keluar secara otomatis
50                  default:
51                      System.exit(0);
52              }
53          }
54      }
55  }
```

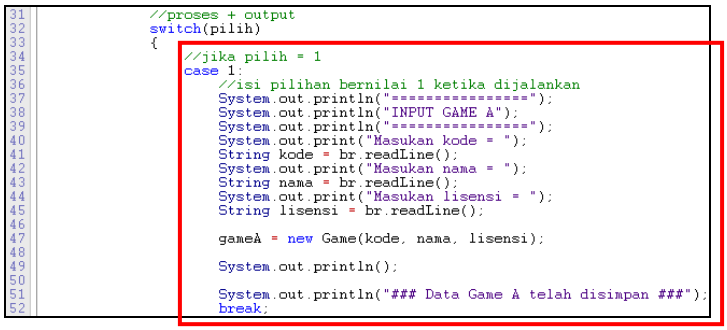
Sekedar tambahan, dalam setiap case jangan lupa menambahkan **break** yang bertugas untuk menghentikan proses pengecekan menu apabila salah satu case sudah terpenuhi dan telah dieksekusi. Penggunaan **default** ditujukan apabila pilihan 1,2, atau 3 tidak sesuai dengan inputan user. Jika anda perhatikan baik-baik, “System.exit(0);” pada line 51 bertujuan untuk keluar dari menu dan mengakhiri program.

f. Mengisi Case 1 (Input Game A)

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

- Line 37-45: berisi permintaan inputan kode, nama, dan lisensi yang nantinya akan diisi oleh user

- Line 47: merupakan proses mentransfer data kode, nama, dan lisensi pada tiap variabel dalam constructor pertama class Game. Setelah berhasil, program akan mencetak tulisan **### Data Game A telah disimpan ###** (line 51).

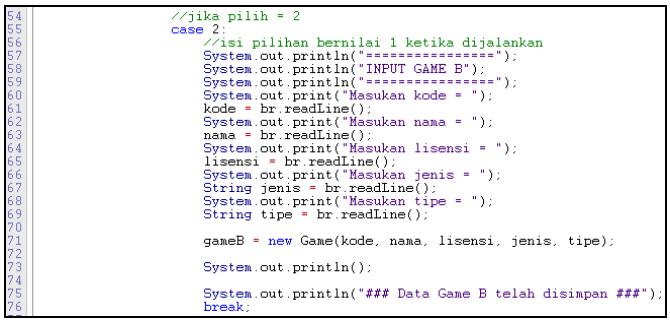


```
31              //proses + output
32              switch(pilih)
33              {
34                  //jika pilih = 1
35                  case 1:
36                      //isi pilihan bernilai 1 ketika dijalankan
37                      System.out.println("================");
38                      System.out.println("INPUT GAME A");
39                      System.out.println("================");
40                      System.out.print("Masukan kode = ");
41                      String kode = br.readLine();
42                      System.out.print("Masukan nama = ");
43                      String nama = br.readLine();
44                      System.out.print("Masukan lisensi = ");
45                      String lisensi = br.readLine();
46
47                      gameA = new Game(kode, nama, lisensi);
48
49                      System.out.println();
50
51                      System.out.println("### Data Game A telah disimpan ###");
52                      break;
```

g. Mengisi Case 2 (Input Game B)

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

- Line 57-69: berisi permintaan inputan kode, nama, dan lisensi, jenis, dan tipe yang nantinya akan diisi oleh user
- Line 71: merupakan proses mentransfer data kode, nama, dan lisensi, jenis, dan tipe pada tiap variabel dalam constructor kedua class Game. Setelah berhasil, program akan mencetak tulisan **### Data Game B telah disimpan ###** (line 75).



```
54                  //jika pilih = 2
55                  case 2:
56                      //isi pilihan bernilai 1 ketika dijalankan
57                      System.out.println("================");
58                      System.out.println("INPUT GAME B");
59                      System.out.println("================");
60                      System.out.print("Masukan kode = ");
61                      kode = br.readLine();
62                      System.out.print("Masukan nama = ");
63                      nama = br.readLine();
64                      System.out.print("Masukan lisensi = ");
65                      lisensi = br.readLine();
66                      System.out.print("Masukan jenis = ");
67                      String jenis = br.readLine();
68                      System.out.print("Masukan tipe = ");
69                      String tipe = br.readLine();
70
71                      gameB = new Game(kode, nama, lisensi, jenis, tipe);
72
73                      System.out.println();
74
75                      System.out.println("### Data Game B telah disimpan ###");
76                      break;
```

h. Mengisi Case 3 (Lihat Data Game)

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

- Line 86-90: mencetak isi data kode, nama, dan lisensi sudah ada pada variabel **gameA**
- Line 95-99: mencetak isi data kode, nama, dan lisensi, jenis, dan tipe sudah ada pada variabel **gameB**

```
78                  //jika pilih = 3
79                  case 3:
80                      //isi pilihan bernilai 3 ketika dijalankan
81                      System.out.println("================");
82                      System.out.println("DATA GAME");
83                      System.out.println("================");
84
85                      System.out.println("Game A");
86                      System.out.println("Kode Game A = "+gameA.getKode());
87                      System.out.println("Nama Game A = "+gameA.getNama());
88                      System.out.println("Lisensi Game A = "+gameA.getLisensi());
89                      System.out.println("Jenis Game A = "+gameA.getJenis());
90                      System.out.println("Tipe Game A = "+gameA.getTipe());
91
92                      System.out.println();
93
94                      System.out.println("GameB");
95                      System.out.println("Kode Game B = "+gameB.getKode());
96                      System.out.println("Nama Game B = "+gameB.getNama());
97                      System.out.println("Lisensi Game B = "+gameB.getLisensi());
98                      System.out.println("Jenis Game B = "+gameB.getJenis());
99                      System.out.println("Tipe Game B = "+gameB.getTipe());
100
101                     System.out.println("================");
102                     break;
```


MODUL 4

Polymorphism

Tujuan:

Mahasiswa dapat mengenal dan memahami konsep polymorphism dan overriding method serta penerapannya dalam konsep OOP

Materi:

- ✓ Pengantar
- ✓ Polymorphism
- ✓ Kriteria penggunaan Polymorphism
- ✓ Overriding Method
- ✓ Soal Latihan

Referensi:

- ❖ Fikri, Rijalul. 2005. *Pemrograman Java*. Yogyakarta: Penerbit Andi
- ❖ Hermawan, Benny. 2004. *Menguasai Java 2 & Object Oriented Programming*. Yogyakarta: Penerbit Andi

4.1 Pengantar

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

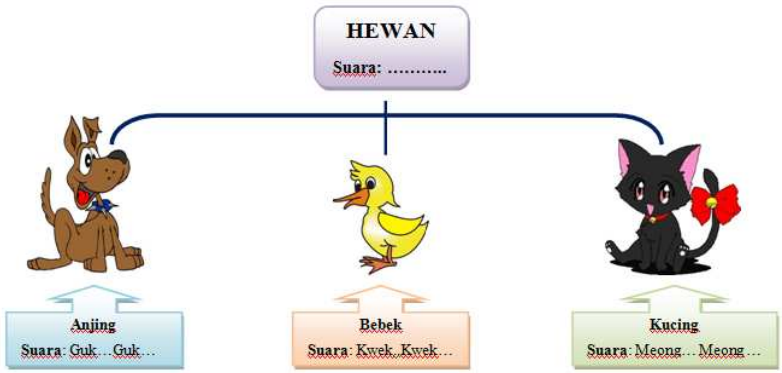
- ✓ Praktikan **bisa** memahami konsep Polymorphism dalam Praktikum Pemrograman Berorientasi Objek
- ✓ Ular itu mempunyai **bisa** yang sangat berbahaya

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.

4.2 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.

4.3 Kriteria penggunaan Polymorphism

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

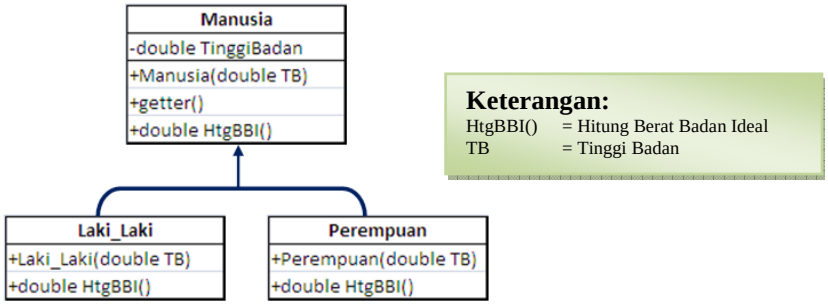
- ✓ Method dari kelas turunan yang akan dieksekusi, dipanggil oleh objek dari kelas induk
- ✓ Nama method yang digunakan pada kelas induk harus ditulis ulang (*overriding method*) pada kelas turunannya, dengan asumsi nama dan tipe data method harus sama

4.4 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.

4.5 Soal Latihan

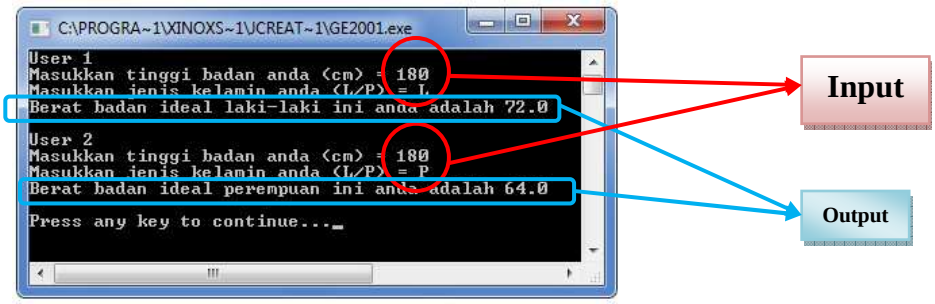
Perhatikan gambar di bawah ini!



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.

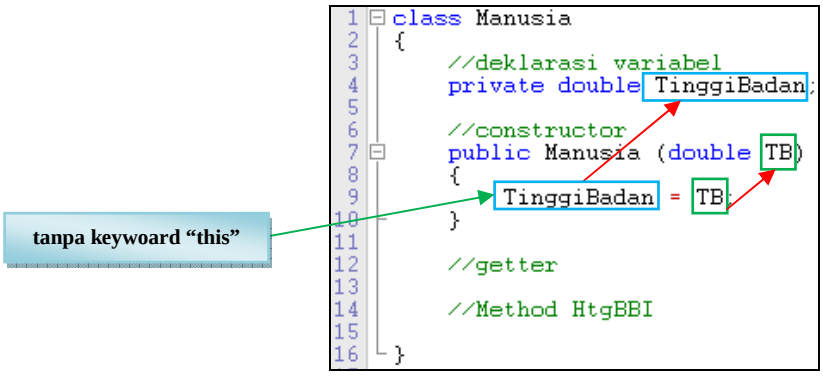
```
1  class Manusia
2  {
3      //deklarasi variabel
4      private double TinggiBadan;
5
6      //constructor
7
8      //getter
9
10     //Method HtgBBI
11
12 }
```

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.

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

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

    //getter

    //Method HtgBBI

}
```

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.


```
1  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     public double getTB()
14     {
15         return TinggiBadan;
16     }
17
18     //Method HtgBBI
19
20 }
```

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().

```
1  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     public double getTB()
14     {
15         return TinggiBadan;
16     }
17
18     //Method HtgBBI
19     public double HtgBBI()
20     {
21         return 0.0;
22     }
23 }
```

➢ **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)**

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

    //Method HtgBBI() merupakan method overidding dari superclass-nya
    public 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().

➢ 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

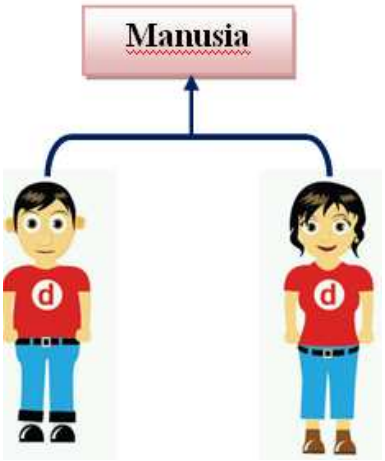
Setelah membuat inputan user, maka dilakukan pengecekan apakah jenis kelamin yang dipilih adalah laki-laki atau perempuan. Hal ini akan menentukan kelas turunan mana yang akan diakses.

- 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 }
```

MODUL 5

Abstract Class



Tujuan:

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

Materi:

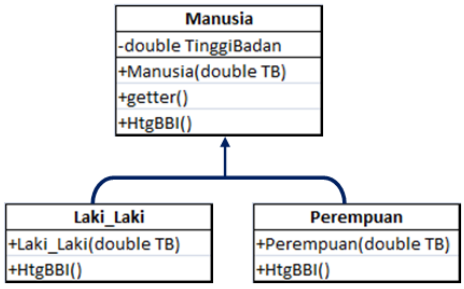
- ✓ Pengantar
- ✓ Abstract Class
- ✓ Keyword "*final*"
- ✓ Soal Latihan

Referensi:

- ❖ Fikri, Rijalul. 2005. *Pemrograman Java*. Yogyakarta: Penerbit Andi
- ❖ Hermawan, Benny. 2004. *Menguasai Java 2 & Object Oriented Programming*. Yogyakarta: Penerbit Andi

5.1 Pengantar

Anda pasti masih ingat contoh program yang menghitung berat badan ideal berdasarkan tinggi badan (bagi yang tidak ingat, bisa membuka kembali soal latihan pada modul 4). 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.

5.2 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:

```
1  abstract class Nama_Kelas
2  {
3      //isi abstract class
4  }
```

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

5.3 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];
    }
}
```

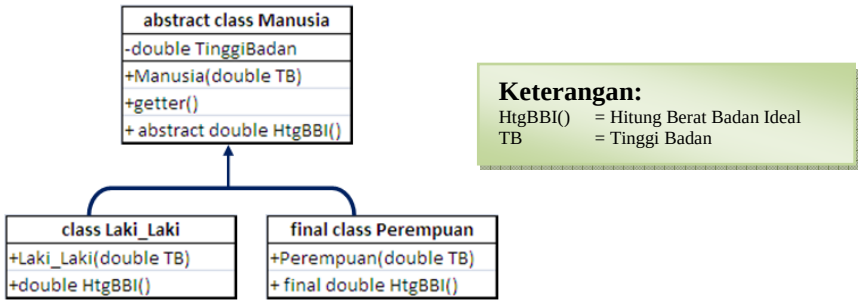
Contoh

```
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

5.4 Soal Latihan

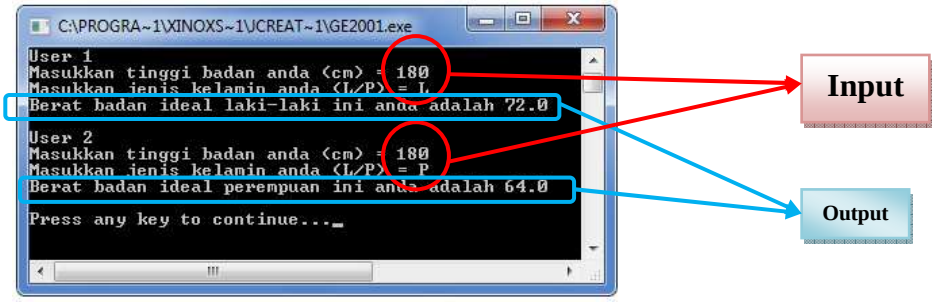
Seperti halnya dengan soal latihan pada modul 4 tentang “Polymorphism”, soal latihan pada modul 5 tidak berbeda jauh. Perhatikan gambar di bawah ini!



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

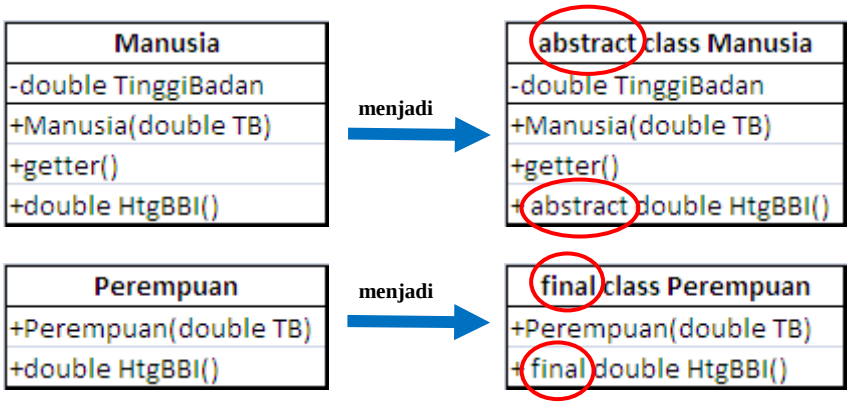
- ❖ Laki-Laki = (tinggi badan(cm)-100) kg x 90%
- ❖ Untuk Perempuan = (tinggi badan(cm)-100) kg x 80%

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 5 dengan soal latihan modul 4, 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.

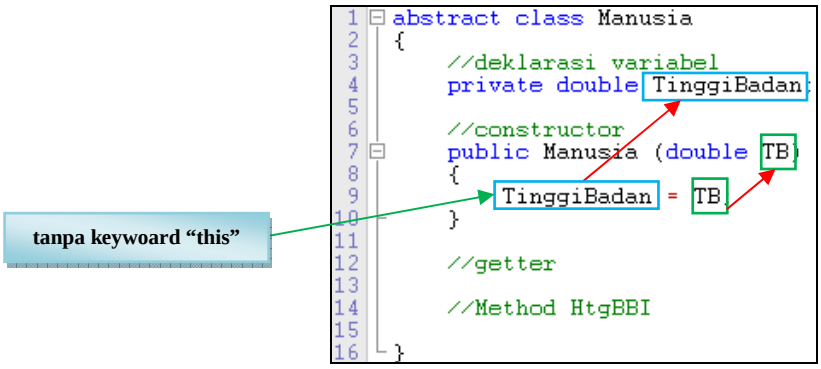
```
abstract class Manusia
{
    //deklarasi variabel
    private double TinggiBadan;

    //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)

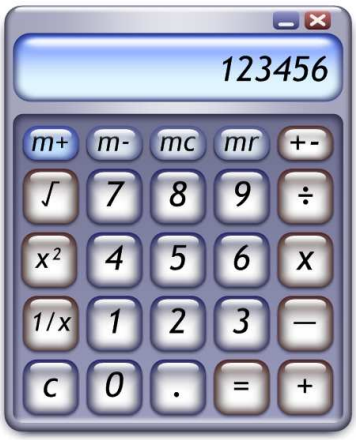
Karena class Utama pada soal latihan 4 = soal latihan 5, maka script ini tidak mengalai perubahan. Adapun scriptnya adalah sebagai berikut:

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

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

MODUL 6

Interface



Tujuan:

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

Materi:

3.6	Pengantar	3.8	Implementasi Interface
3.7	Interface	3.9	Soal Latihan

Referensi:

- Fikri, Rijalul. 2005. *Pemrograman Java*. Yogyakarta: Penerbit Andi
- Hermawan, Benny. 2004. *Menguasai Java 2 & Object Oriented Programming*. Yogyakarta: Penerbit Andi

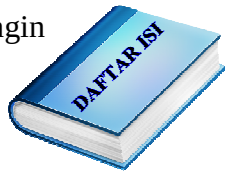
6.1 Pengantar

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. Untuk lebih jelas mengenai konsep interface, simak penjelasannya di bawah ini.

6.2 Interface

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:

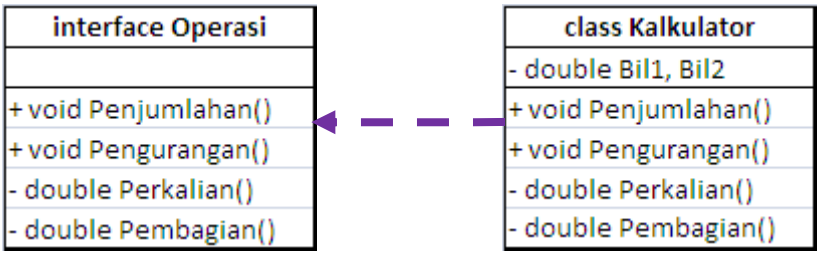
```
interface Operasi
{
    //deklarasi variabel dan/atau method
    public void Penjumlahan();
    public void Pengurangan();
    public double Perkalian();
    public double Pembagian();
}
```

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.**

6.3 Implementasi Interface

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
}
```

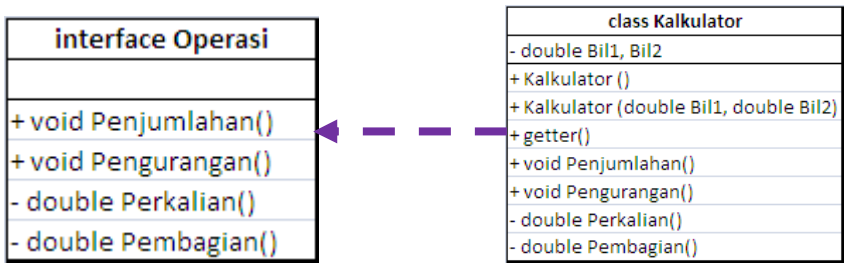
Contoh:

```
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.

6.4 Soal Latihan

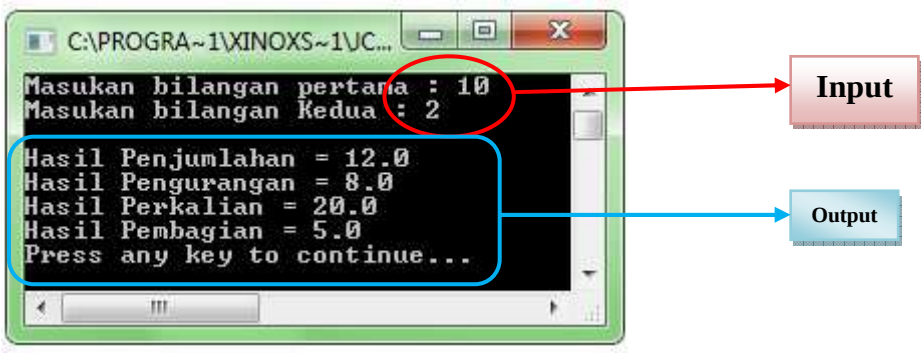
Berdasarkan contoh di atas tentang kalkulator, buatlah program sesuai dengan skema di bawah ini:



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

```
1  interface Operasi
2  {
3      //deklarasi method
4  }
```

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

```
1  interface Operasi
2  {
3      //deklarasi method
4      public void Penjumlahan();
5      public void Pengurangan();
6      public double Perkalian();
7      public double Pembagian();
8  }
```

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

➢ Langkah 2: class Kalkulator (ketikkan script berikut)

a. Membuat kerangka class Kalkulator

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

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*. 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.


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

    //constructor

    //getter

    //implementasi method

};
```

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.

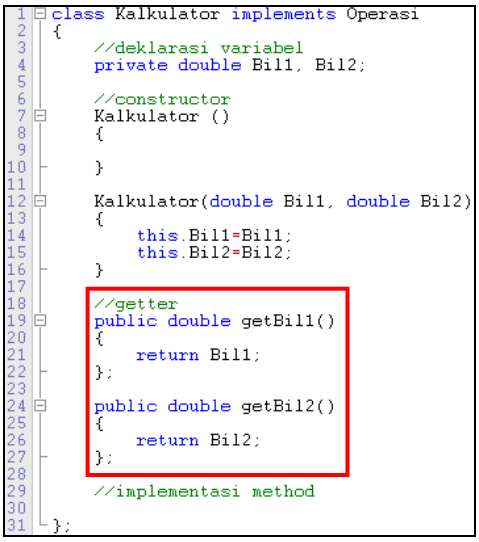
```
class Kalkulator implements Operasi
{
    //deklarasi variabel
    private double Bil1, Bil2;

    //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

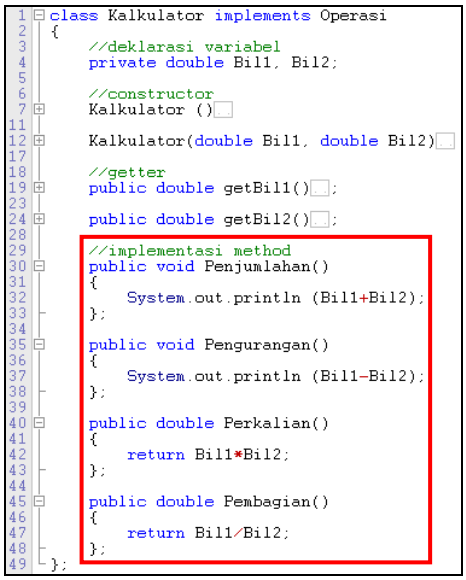
kembalikan ke dalam class Utama. Perlu diketahui pula, bahwa dalam pembuatan variabel pada parameter di method getter, harus sesuai dengan tipe data pada variabel yang dideklarasikan di kelas. Gambar di bawah ini menunjukkan deklarasi getter.



```
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().

MODUL 7

I/O Stream

Tujuan:

Mahasiswa dapat mengenal dan memahami konsep I/O Stream, serta penerapan I/O Stream dalam konsep OOP

Materi:

- ✓ Pengantar
- ✓ *OutputStream*
- ✓ *OutputStream*
- ✓ Soal Latihan

Referensi:

- ❖ Fikri, Rijalul. 2005. *Pemrograman Java*. Yogyakarta: Penerbit Andi
- ❖ Hermawan, Benny. 2004. *Menguasai Java 2 & Object Oriented Programming*. Yogyakarta: Penerbit Andi

7.1 Pengantar

Ilustrasi 1

Langkah – langkah apa saja yang akan anda lakukan ketika anda disuruh menghapalkan 3 buah kalimat sama persis seperti yang diucapkan oleh dosen, tanpa ada pengurangan maupun penambahan kalimat?

Ilustrasi 2

Menurut anda, apa kelebihan CD-RW/DVD-RW dibandingkan dengan CD-R/DVD-R?

Pada ilustrasi 1, **menulis** pada sebuah kertas dan **membaca** kembali isi pada kertas kemudian dihapalkan merupakan salah satu cara termudah dalam meniru perkataan dosen. Sedangkan pada ilustrasi 2, penggunaan CD-RW/DVD-RW merupakan salah satu penyimpanan yang lebih baik dibandingkan dengan CD-R/DVD-R. Hal ini dikarenakan pada CD-RW dapat melakukan proses **read** dan **write** berulang kali, meskipun antara CD-R dengan CD-RW memiliki ukuran dan kecepatan menulis yang sama. Demikian juga untuk membandingkan antara DVD-R dengan DVD-RW.

Read atau/dan Write yang tampak pada ilustrasi 1 dan ilustrasi 2, merupakan salah satu konsep Java yang biasa dikenal dengan nama I/O Stream. Di dalam Java, penerapan Read menggunakan Class InputStream, sedangkan penerapan Write menggunakan Class OutputStream. Penggunaan read/write digunakan untuk membantu anda dalam menyimpan hasil keluaran (output) program yang telah anda inputkan ke dalam komputer pada sebuah file. Untuk lebih jelas mengenai I/O Stream, simak penjelasannya di bawah ini.

7.2 OutputStream

OutputStream merupakan class induk yang digunakan untuk menangani operasi output. **Class ini merupakan kelas abstrak**, dimana kelas ini tidak dapat digunakan secara langsung ke dalam kelas utama, melainkan harus diturunkan terlebih dahulu ke kelas turunannya. Berikut beberapa class turunan dari class OutputStream yang dapat digunakan:

- ✓ ByteArrayOutputStream
- ✓ FileOutputStream
- ✓ FilterOutputStream
- ✓ ObjectOutputStream
- ✓ PipedOutputStream

Algoritma dalam penulisan data ke dalam file:

- Koneksi OutputStream ke dalam file
- Tulis data
- Tutup file

Contoh:

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

        //input data
        System.out.print("Masukan nama : ");
        String nama = br.readLine();
        System.out.print("Masukan umur: ");
        int umur = Integer.parseInt(br.readLine());
        System.out.print("Masukan IPK : ");
        double ipk = Double.parseDouble(br.readLine());

        System.out.println();

        //akses FileOutputStream

        //buat file dangan nama "Biodata.txt"
        FileOutputStream fos = new FileOutputStream("Biodata.txt");
        //melakukan proses "write" dengan DataOutputStream
        DataOutputStream dos = new DataOutputStream (fos);

        //tampung data inputan ke dalam file "Biodata.txt"
        dos.writeUTF(nama);
        dos.writeInt(umur);
        dos.writeDouble(ipk);

        //menutup DataOutputStream
        dos.close();

        //cetak keterangan
        System.out.println("Data berhasil disimpan ke dalam file \"Biodata.txt\"");
    }
}
```

Keterangan:

Line 9-14 = inputan user yang ditampung ke dalam variabel

Line 21 = membuat file “Biodata.txt” dengan menggunakan kelas FileOutputStream

Line 23 = memberikan kemampuan kepada file tersebut dalam menulis data (*write*) ke dalam file menggunakan kelas DataOutputStream

Line 26 = memasukkan data yang bertipe **String** ke dalam file “Biodata.txt” menggunakan method **writeUTF(nama_file)**

Line 27 = memasukkan data yang bertipe **Integer** ke dalam file “Biodata.txt” menggunakan method **writeInt(nama_file)**

Line 28 = memasukkan data yang bertipe **Double** ke dalam file “Biodata.txt” menggunakan method **writeDouble(nama_file)**

Line 31 = menutup file “Biodata.txt” sehingga tidak dapat dilakukan proses “write” kembali

Line 34 = mencetak keterangan

Hasilnya adalah sebagai berikut:

Untuk menambah file yang telah ada isinya, maka diperlukan diperlukan nilai true pada parameter FileOutputStream, sehingga baris pada class TulisFile.java:

```
FileOutputStream fos = new FileOutputStream("Biodata.txt");
```

Diubah menjadi:

```
FileOutputStream fos = new FileOutputStream("Biodata.txt", true);
```

7.3 InputStream

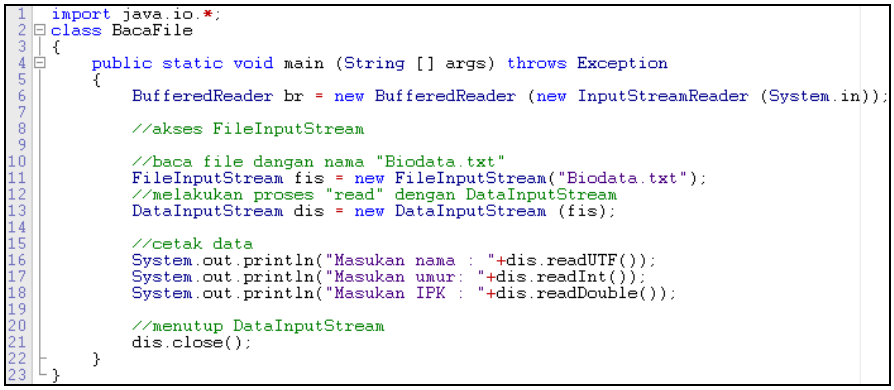
InputStream merupakan class induk yang digunakan untuk menangani operasi input. **Class ini merupakan kelas abstrak**, dimana kelas ini tidak dapat digunakan secara langsung ke dalam kelas utama, melainkan harus diturunkan terlebih dahulu ke kelas turunannya. Berikut beberapa class turunan dari class InputStream yang dapat digunakan:

- ✓ ByteArrayInputStream
- ✓ FileInputStream
- ✓ FilterInputStream
- ✓ ObjectInputStream
- ✓ PipedInputStream

Algoritma dalam membaca data dalam sebuah file:

- Koneksi InputStream ke dalam file
- Baca data
- Tutup file

Contoh:



```
1  import java.io.*;
2  class BacaFile
3  {
4      public static void main (String [] args) throws Exception
5      {
6          BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
7
8          //akses FileInputStream
9
10         //baca file dangan nama "Biodata.txt"
11         FileInputStream fis = new FileInputStream("Biodata.txt");
12         //melakukan proses "read" dengan DataInputStream
13         DataInputStream dis = new DataInputStream (fis);
14
15         //cetak data
16         System.out.println("Masukan nama : "+dis.readUTF());
17         System.out.println("Masukan umur: "+dis.readInt());
18         System.out.println("Masukan IPK : "+dis.readDouble());
19
20         //menutup DataInputStream
21         dis.close();
22     }
23 }
```

Keterangan:

Line 11 = mencari file “Biodata.txt” dengan menggunakan kelas FileInputStream

Line 13 = memberikan kemampuan kepada file tersebut dalam membaca data (*read*) ke dalam file menggunakan kelas DataInputStream

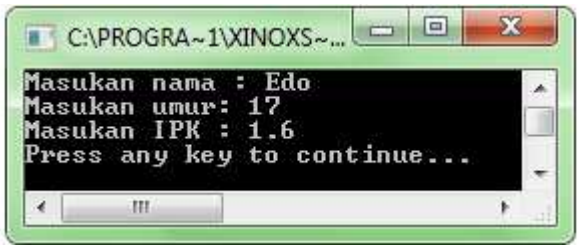
Line 16 = mencetak data yang bertipe **String** pada file “Biodata.txt” menggunakan method **readUTF(nama_file)**

Line 17 = mencetak data yang bertipe **Integer** pada file “Biodata.txt” menggunakan method **readInt(nama_file)**

Line 18 = mencetak data yang bertipe **Double** pada file “Biodata.txt” menggunakan method **readDouble(nama_file)**

Line 20 = menutup file “Biodata.txt” sehingga tidak dapat dilakukan proses “read” kembali

Hasilnya adalah sebagai berikut:



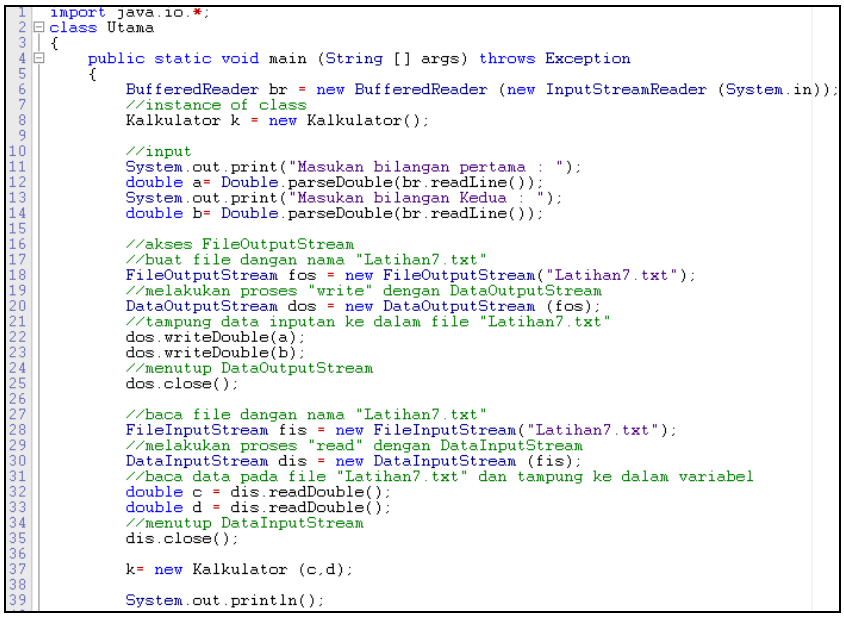
Lalu bagaimana jika anda ingin membaca data dalam file yang terdapat 2 record atau lebih? Cobalah untuk bereskperimen sendiri… ☺

7.4 Soal Latihan

- Seperti pada soal latihan modul 6, buatlah inputan user untuk memasukkan bilangan 1 dan bilangan 2 pada class Utama.
- Data bilangan 1 dan bilangan 2 kemudian ditampung ke dalam file bernama "latihan7.txt".
- Lakukan pembacaan file tersebut sehingga data tersebut dapat digunakan sebagai variable untuk class Kalkulator sehingga dapat diketahui hasil penjumlahan, pengurangan, perkalian, dan pembagian.

Jawabannya…

Berikut adalah script 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          //instance of class
8          Kalkulator k = new Kalkulator();
9
10         //input
11         System.out.print("Masukan bilangan pertama : ");
12         double a= Double.parseDouble(br.readLine());
13         System.out.print("Masukan bilangan Kedua : ");
14         double b= Double.parseDouble(br.readLine());
15
16         //akses FileOutputStream
17         //buat file dangan nama "Latihan7.txt"
18         FileOutputStream fos = new FileOutputStream("Latihan7.txt");
19         //melakukan proses "write" dengan DataOutputStream
20         DataOutputStream dos = new DataOutputStream (fos);
21         //tampung data inputan ke dalam file "Latihan7.txt"
22         dos.writeDouble(a);
23         dos.writeDouble(b);
24         //menutup DataOutputStream
25         dos.close();
26
27         //baca file dangan nama "Latihan7.txt"
28         FileInputStream fis = new FileInputStream("Latihan7.txt");
29         //melakukan proses "read" dengan DataInputStream
30         DataInputStream dis = new DataInputStream (fis);
31         //baca data pada file "Latihan7.txt" dan tampung ke dalam variabel
32         double c = dis.readDouble();
33         double d = dis.readDouble();
34         //menutup DataInputStream
35         dis.close();
36
37         k= new Kalkulator (c,d);
38
39         System.out.println();
```

Gambar *script* pada bagian atas

Keterangan:

Line 11-14 = inputan user yang ditampung ke dalam variabel

Line 18 = membuat file “Latihan7.txt” dengan menggunakan kelas FileOutputStream

Line 20 = memberikan kemampuan kepada file tersebut dalam menulis data (*write*) ke dalam file menggunakan kelas DataOutputStream

Line 22-23 = memasukkan data yang bertipe **Double** ke dalam file “Latihan7.txt” menggunakan method **writeDouble(nama_file)**

Line 25 = menutup file “Latihan7.txt” sehingga tidak dapat dilakukan proses “write” kembali

Line 28 = mencari file “Latihan7.txt” dengan menggunakan kelas FileInputStream

Line 30 = memberikan kemampuan kepada file tersebut dalam

membaca data (*read*) ke dalam file menggunakan kelas DataInputStream

Line 32-33 = membaca data yang bertipe **Double** pada file "Latihan7.txt" menggunakan method **readDouble(nama_file)** dan ditampung ke dalam variabel

Line 35 = menutup file "Biodata.txt" sehingga tidak dapat dilakukan proses "read" kembali

Line 37 = mentransfer isi data pada variabel **c** dan **d** (yang diperoleh dari "Latihan7.txt") ke dalam constructor Kalkulator

```
40              //output
41              System.out.print ("Hasil Penjumlahan = ");
42              k.Penjumlahan();
43              System.out.print ("Hasil Pengurangan = ");
44              k.Pengurangan();
45              System.out.println("Hasil Perkalian = "+k.Perkalian());
46              System.out.println("Hasil Pembagian = "+k.Pembagian());
47          }
48      }
```

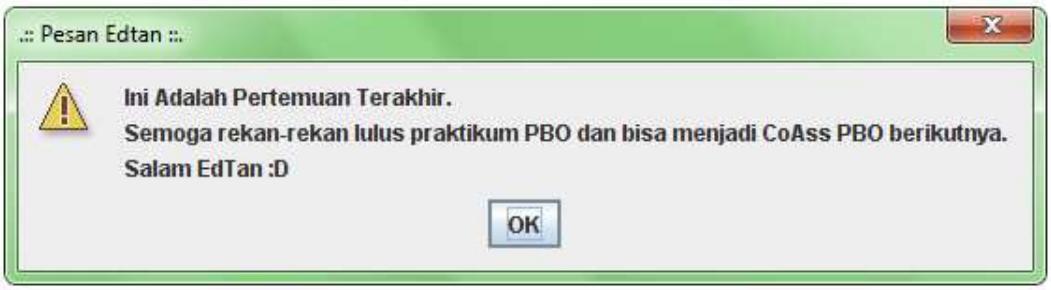
Gambar *script* pada bagian bawah (lanjutan)

Keterangan:

Line 41-46 = mencetak hasil penjumlahan, pengurangan, perkalian, dan pembagian dengan memanggil method yang terdapat pada class Kalkulator

MODUL 8

JOptionPane



Tujuan:

Mahasiswa dapat menggunakan JOptionPane sebagai salah satu GUI dalam dalam konsep OOP

Materi:

- ✓ Pengantar
- ✓ *JOptionPane*
- ✓ *Method-Method JOptionPane*
- ✓ Soal Latihan

Referensi:

- ❖ http://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html
- ❖ Tutorial Java API

8.1 Pengantar

Selama pertemuan 1 s/d pertemuan 7, anda telah mempelajari beberapa hal yang berkaitan tentang OOP, mulai dari pengenalan OOP, constructor, inheritance, polymorphism, abstract class, interface, hingga I/O Stream. Namun, pernahkah anda berpikir untuk mengganti tampilan program anda menjadi bentuk GUI. Salah satu yang bisa kita gunakan adalah JoptionPane. Biar tidak penasaran, simak cerita saya di bawah ini.

8.2 JOptionPane

JOptionPane merupakan salah satu class yang dapat digunakan dalam membuat dialog box berisi inputan, pesan, maupun konfirmasi yang dapat dilakukan oleh user. Yang perlu anda ketahui dalam mengakses class JOptionPane, anda harus mengimport **javax.swing**. Berikut adalah cara deklarasi JOptionPane.

```
JOptionPane [nama_varibel] = new JOptionPane()
```

Contoh:

```
JOptionPane jop = new JOptionPane()
```

8.3 Method-Method JOptionPane

Terdapat 4 method yang umumnya sering digunakan dalam class JOptionPane, yaitu:

8.3.1 showMessageDialog

Merupakan dialog box yang digunakan untuk menampilkan pesan atau hasil output yang akan dilaporkan kepada user. Syntax umum yang digunakan pada method ini adalah sebagai berikut:

```
showConfirmDialog(Component parentComponent,Object message,String title,int optionType)
```

atau

```
showConfirmDialog(Component parentComponent,Object message,String title,int messageType)
```

Keterangan:

- parentComponent : mendefinisikan component yang menjadi parent dari dialog box. Jika diisi dengan **null**, maka secara default Frame akan menjadi parent component
- message : berisi pesan yang ditampilkan pada dialog box
- title : judul pada dialog box
- optionType : berisi pilihan yang tombol yang dapat dipilih oleh user. Pilihan bisa berupa DEFAULT_OPTION, YES_NO_OPTION, YES_NO_CANCEL_OPTION, OK_CANCEL_OPTION
- messageType : menampilkan tipe pesan dengan memberi icon pada dialog box. Tipe pesan dapat berupa ERROR_MESSAGE,INFORMATION_MESSAGE, WARNING_MESSAGE, QUESTION_MESSAGE, PLAIN_MESSAGE

Contoh penggunaan messageType pada deklarasi syntax yang kedua:

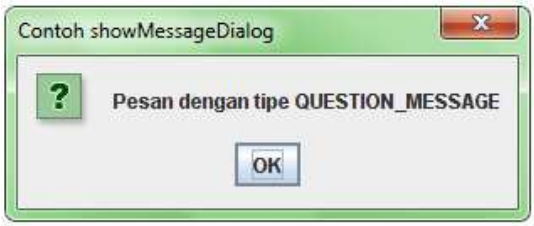
```
1  import javax.swing.*;
2  class Contoh_showMessageDialog
3  {
4      public static void main (String [] args)
5      {
6          JOptionPane jop  = new JOptionPane ();
7
8          jop.showMessageDialog(null,"Pesan dengan tipe ERROR_MESSAGE","Contoh showMessageDialog",jop.ERROR_MESSAGE);
9          jop.showMessageDialog(null,"Pesan dengan tipe INFORMATION_MESSAGE","Contoh showMessageDialog",jop.INFORMATION_MESSAGE);
10         jop.showMessageDialog(null,"Pesan dengan tipe WARNING_MESSAGE","Contoh showMessageDialog",jop.WARNING_MESSAGE);
11         jop.showMessageDialog(null,"Pesan dengan tipe QUESTION_MESSAGE","Contoh showMessageDialog",jop.QUESTION_MESSAGE);
12         jop.showMessageDialog(null,"Pesan dengan tipe PLAIN_MESSAGE","Contoh showMessageDialog",jop.PLAIN_MESSAGE);
13     }
14 }
```

Hasilnya adalah sebagai berikut:

Hasil dari script pada *line* 8

Hasil dari script pada *line* 9

Hasil dari script pada *line* 10



Hasil dari script pada *line* 11

Hasil dari script pada *line* 12

8.3.2 showInputDialog

Merupakan *prompt* yang digunakan untuk menerima inputan user. Nilai yang dimasukan user ke dalam prompt bertipe Object, sehingga inputan bisa dimasukkan ke dalam variable bertipe String. Apabila anda ingin memasukkan inputan ke dalam variable bertipe int maupun double, anda terlebih dahulu harus melakukan parsing. Syntax umum yang digunakan pada method ini adalah sebagai berikut:

showInputDialog(Component parentComponent, Object message)

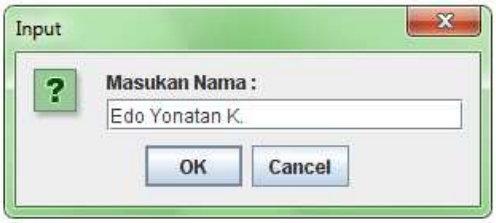
Keterangan:

- parentComponent : mendefinisikan component yang menjadi parent dari dialog box. Jika diisi dengan **null**, maka secara default Frame akan menjadi parent component
- message : berisi pesan yang ditampilkan pada dialog box

Contoh penghitungan berat badan ideal:
(asumsi: rumus yang digunakan hanya berlaku untuk jenis kelamin laki-laki)

```
1  import javax.swing.*;
2  class Contoh_showInputDialog
3  {
4      public static void main (String [] args)
5      {
6          JOptionPane jop  = new JOptionPane ();
7
8          String nama = JOptionPane.showInputDialog(null,"Masukan Nama : ");
9          double tb = Double.parseDouble(JOptionPane.showInputDialog(null, "Masukan Tinggi Badan : "));
10
11         double bbi = (tb-100)*0.9;
12
13         String cetak = "Data User\nNama : "+nama+"\nTinggi Badan : "+tb+" cm\nBerat Badan Ideal : "+bbi+" kg";
14
15         jop.showMessageDialog(null,cetak,"Hasil Berat Badan Ideal",jop.INFORMATION_MESSAGE);
16     }
17 }
```

Hasilnya adalah sebagai berikut:



Hasil dari script pada *line* 8

Hasil dari script pada *line* 9

Hasil dari script pada *line* 13 dan *line* 15

8.3.3 showConfirmDialog

Merupakan dialog box yang digunakan untuk menanyakan dan memberikan pilihan kepada user dalam melakukan tindakan/aksi berikutnya. Karena hasil yang diperoleh dari method ini adalah bertipe integer, maka anda tidak perlu melakukan parsing. Syntax umum yang digunakan pada method ini adalah sebagai berikut:

```
showConfirmDialog(Component parentComponent, Object message, String title, int optionType)
```

Keterangan:

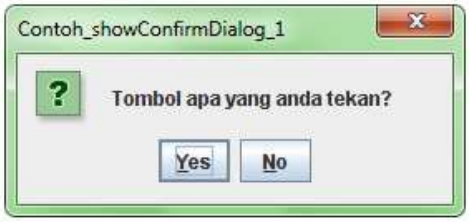
- parentComponent : mendefinisikan component yang menjadi parent dari dialog box. Jika diisi dengan **null**, maka secara default Frame akan menjadi parent component
- message : berisi pesan yang ditampilkan pada dialog box

- title : judul pada dialog box
- optionType : berisi pilihan yang tombol yang dapat dipilih oleh user. Pilihan bisa berupa DEFAULT_OPTION, YES_NO_OPTION, YES_NO_CANCEL_OPTION, OK_CANCEL_OPTION

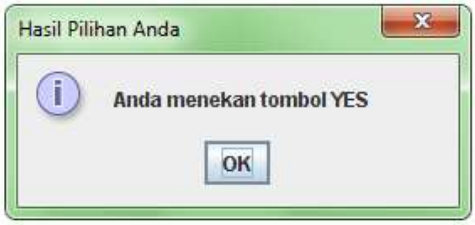
Contoh:

```
1  import javax.swing.*;
2  class Contoh_showConfirmDialog_1
3  {
4      public static void main (String [] args)
5      {
6          JOptionPane jop  = new JOptionPane ();
7
8          int pilih = jop.showConfirmDialog(null, "Tombol apa yang anda tekan?", "Contoh_showConfirmDialog_1", jop.YES_NO_OPTION);
9
10         if (pilih == 0)
11         {
12             jop.showMessageDialog(null,"Anda menekan tombol YES","Hasil Pilihan Anda",jop.INFORMATION_MESSAGE);
13         }
14         else
15         {
16             jop.showMessageDialog(null,"Anda menekan tombol NO","Hasil Pilihan Anda",jop.INFORMATION_MESSAGE);
17         }
18     }
19 }
```

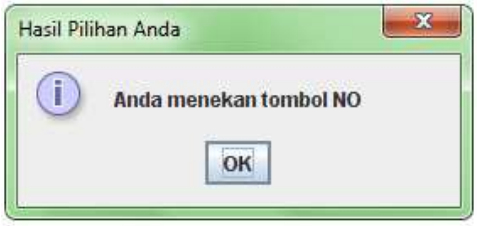
Hasilnya adalah sebagai berikut:



Hasil dari script pada *line* 8



Hasil dari script pada *line* 12, apabila pilihan = 0



Hasil dari script pada *line* 16, , apabila pilihan = 1

Jika anda perhatikan baik-baik, pilihan Yes dianggap apabila pilihan=0, sedangkan pilihan No dianggap apabila pilihan=1. Lalu bisakah anda

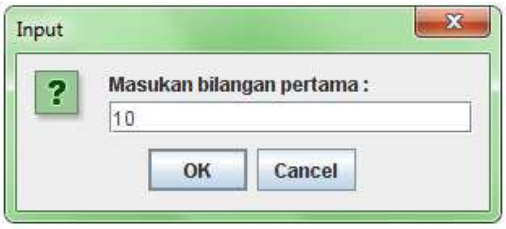
membuat program seperti di atas apabila **optionType** menggunakan YES_NO_CANCEL_OPTION? Cobalah untuk bereskperimen sendiri… ☺

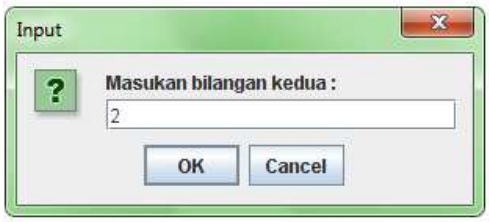
8.3.4 showOptionDialog

Fungsi showOptionDialog hampir serupa dengan method showConfirmDialog. Perbedaannya, pada method showConfirmDialog terdapat 4 varian method yang berbeda-beda pada parameternya, sedangkan method showOptionDialog hanya memiliki 1 varian method saja.

8.4 Soal Latihan

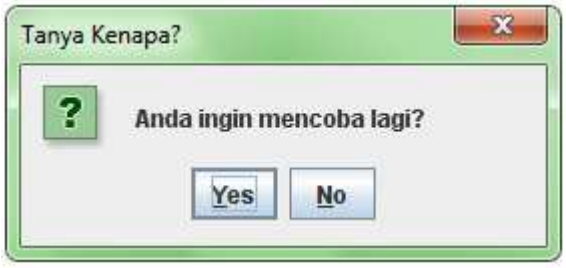
Seperti pada soal latihan modul 6, ubahlah menu class Utama yang sebelumnya tampil dalam bentuk command prompt menjadi tampilan dalam bentuk dialog box seperti contoh di bawah ini:



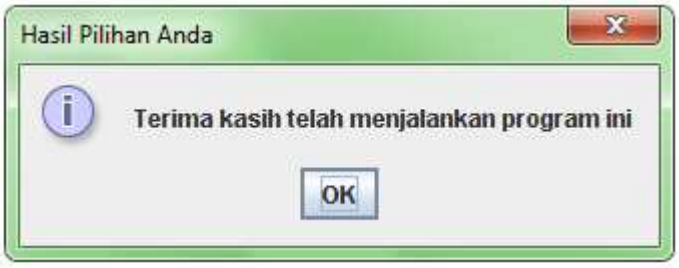


Apabila inputan sudah dilakukan, maka hasil Penjumlahan dan hasil Pengurangan akan masuk ke dalam command prompt. Sedangan hasil Perkalian dan hasil Pembagian akan masuk ke dalam dialog box.

Apabila anda telah menekan tombol "OK", maka akan tampil dialog box seperti di bawah ini:

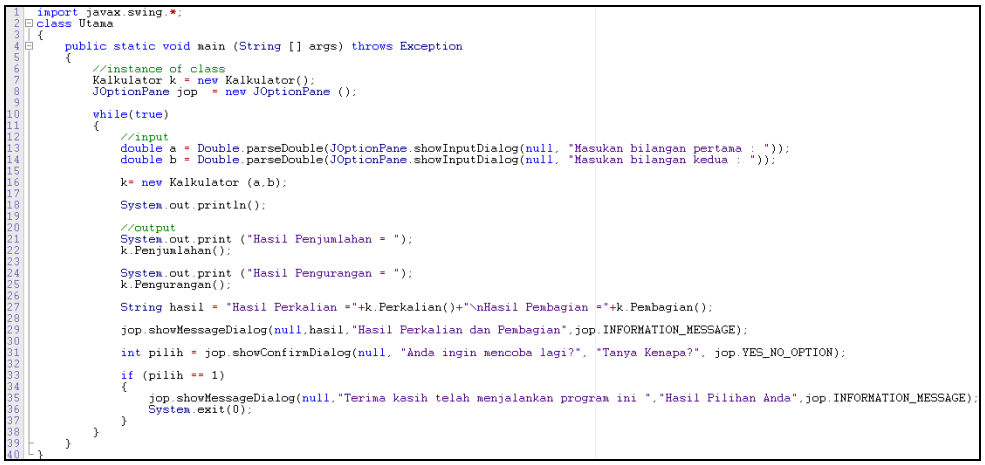


Apabila anda ingin mencoba lagi, maka aplikasi akan meminta inputan kembali untuk bilangan pertama dan bilangan kedua. Sedangan jika tidak, maka akan tampil dialog box, seperti di bawah ini:



Jawabannya…

Berikut adalah script class Utama:



```
1  import javax.swing.*;
2  class Utama
3  {
4      public static void main (String [] args) throws Exception
5      {
6          //instance of class
7          Kalkulator k = new Kalkulator();
8          JOptionPane jop  = new JOptionPane ();
9
10         while(true)
11         {
12             //input
13             double a = Double.parseDouble(JOptionPane.showInputDialog(null, "Masukan bilangan pertama : "));
14             double b = Double.parseDouble(JOptionPane.showInputDialog(null, "Masukan bilangan kedua : "));
15
16             k= new Kalkulator (a,b);
17
18             System.out.println();
19
20             //output
21             System.out.print ("Hasil Penjumlahan = ");
22             k.Penjumlahan();
23
24             System.out.print ("Hasil Pengurangan = ");
25             k.Pengurangan();
26
27             String hasil = "Hasil Perkalian ="+k.Perkalian()+"\nHasil Pembagian ="+k.Pembagian();
28
29             jop.showMessageDialog(null,hasil,"Hasil Perkalian dan Pembagian",jop.INFORMATION_MESSAGE);
30
31             int pilih = jop.showConfirmDialog(null, "Anda ingin mencoba lagi?", "Tanya Kenapa?", jop.YES_NO_OPTION);
32
33             if (pilih == 1)
34             {
35                 jop.showMessageDialog(null,"Terima kasih telah menjalankan program ini ","Hasil Pilihan Anda",jop.INFORMATION_MESSAGE);
36                 System.exit(0);
37             }
38         }
39     }
40 }
```

Keterangan:

Line 7 = instance of class dari class Kalkulator menggunakan variabel **k**

Line 7 = instance of class dari class JOptionPane menggunakan variabel **jop**

Line 13-14 = inputan user yang ditampung ke dalam variabel dimana dialogbox berbentuk *prompt* dengan menggunakan method **showInputDialog()**

Line 16 = mentransfer isi data pada variabel **a** dan **b** ke dalam constructor Kalkulator

Line 21-25 = mencetak hasil penjumlahan dan pengurangan ke dalam *command prompt* dengan memanggil method yang terdapat pada class Kalkulator

Line 29 = mencetak hasil perkalian dan pembagian dengan memanggil method yang terdapat pada class Kalkulator dan ditampilkan ke dalam dialogbox menggunakan method **showMessageDialog()**

Line 31-37 = menanyakan kepada user apakah akan mencoba menggunakan aplikasi ini lagi. Jika jawaban **Yes** (bernilai 0), maka program akan mengulang ke menu awal. Sedangkan apabila jawaban **No** (bernilai 1), maka script akan menjalankan line 35 dan program akan keluar (line 36).

Biografi Penulis

Edo Yonatan Koentjoro, S. Kom, merupakan salah satu staff pengajar di Laboratorium STIKOM Surabaya. Penulis yang memiliki nickname “EdTan” ini lahir pada tanggal 18 Desember 1989 dan telah menyelesaikan studi S1 di STIKOM Surabaya tahun 2011. Sebelum lulus S1, penulis aktif dalam organisasi UKM Paskibra. Penulis juga pernah terjun dalam dunia jurnalistik di majalah kampus STIKOM Surabaya News (SSNEWS). Selain itu, penulis juga pernah bekerja sebagai Co-Assisten di Laboratorium Komputer STIKOM Surabaya selama 2 tahun di bidang Office dan beberapa pemrograman, seperti Java, Visual Basic .Net, dan Web (HTML+Javascript).