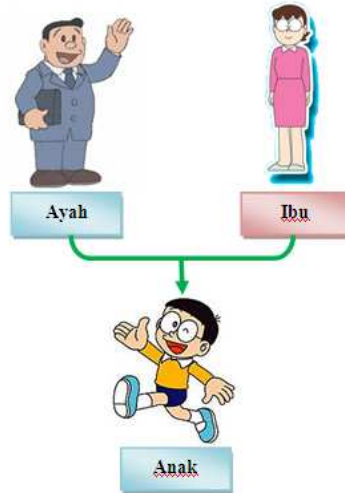


MODUL 3

Inheritance



Tujuan:

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

Materi:

- ✓ Pengantar
- ✓ Inheritance
- ✓ Manfaat Penggunaan Inheritance
- ✓ 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

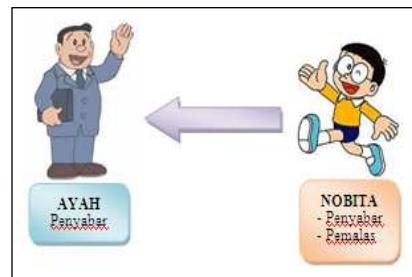
1.1. Pengantar

“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.

1.2. 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. Walaupun berbeda jenisnya, software tersebut pasti memiliki kesamaan dengan software sejenisnya. Dengan kata lain, apabila saya melihat dari segi pemrograman, maka **class Software** adalah **superclass**, sedangkan **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 }
```

1.3. Manfaat Penggunaan Inheritance

Berikut ini adalah beberapa manfaat apabila anda menggunakan konsep inheritance:

1. 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.

2. Kemudahan dalam manage 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.

1.4. Keyword “super”

Keyword **super** digunakan oleh subclass untuk memanggil constructor atau 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.

1.5. 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
 - Constructor pertama tidak memiliki parameter dan tidak ada isinya (kosongan)
 - 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
 - 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*)
 - 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)

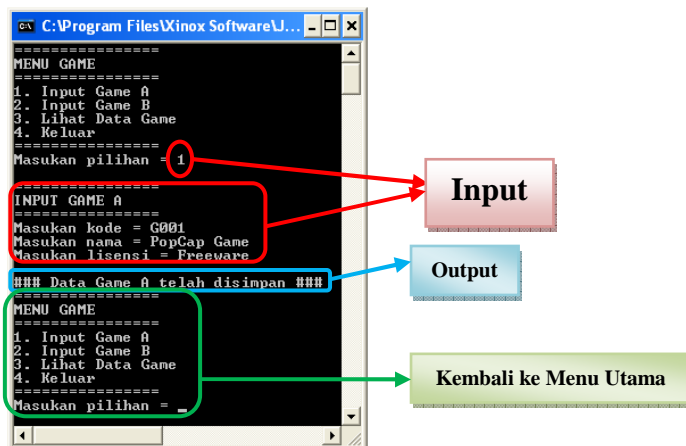
Sedangkan pada Class utama digunakan untuk memanggil class Game.

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

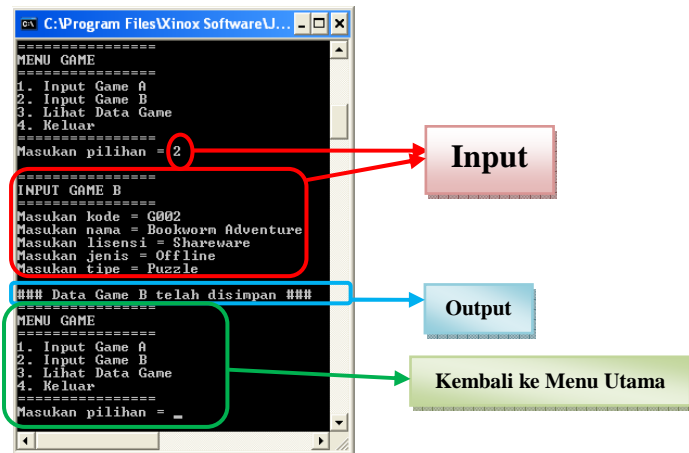
```

C:\Program File...
=====
MENU GAME
=====
1. Input Game A
2. Input Game B
3. Lihat Data Game
4. Keluar
=====
Masukan pilihan = 1
=====
INPUT GAME A
=====
Masukan kode = 
  
```

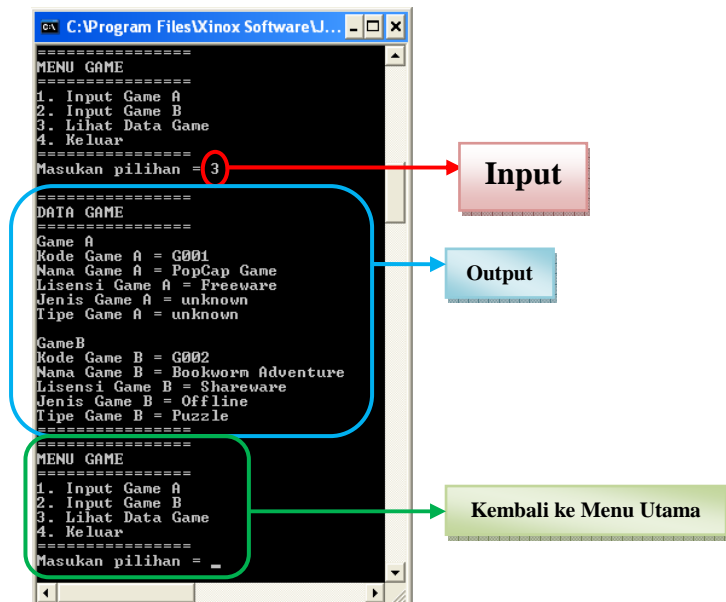
- ❖ Jika pilihan = 1, maka akan tampil sebagai berikut:



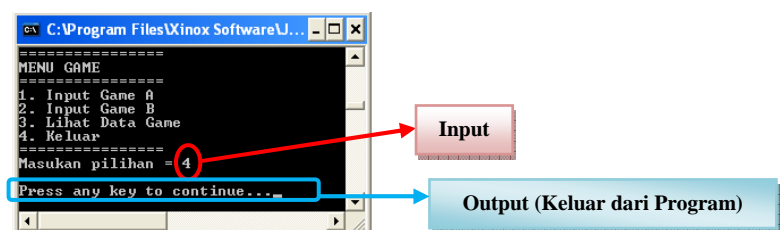
- ❖ Jika pilihan = 2, maka akan tampil sebagai berikut:



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



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



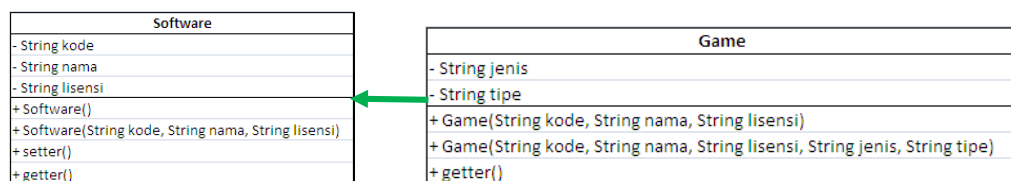
Jawabannya adalah...

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 inheritance, 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 }
```

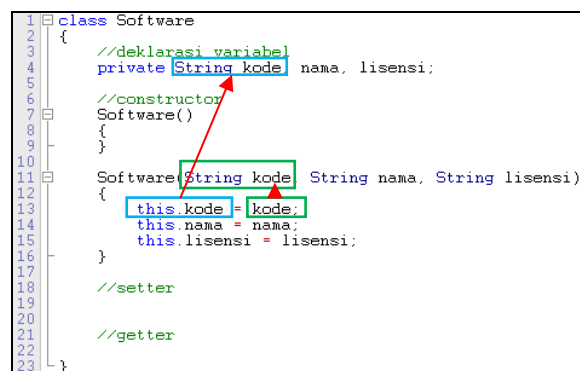
c. Mendeklarasi constructor

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).



```
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 }
```

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.

```
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     public void setKode(String k)
20     {
21         kode = k;
22     }
23     public void setNama(String n)
24     {
25         nama = n;
26     }
27     public void setLisensi(String l)
28     {
29         lisensi = l;
30     }
31
32     //getter
33
34 }
```

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.

```
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     }
14
15     //setter
16     public void setKode(String k)
17     {
18         kode = k;
19     }
20     public void setNama(String n)
21     {
22         nama = n;
23     }
24     public void setLisensi(String l)
25     {
26         lisensi = l;
27     }
28
29     //getter
30     public String getKode()
31     {
32         return kode;
33     }
34     public String getNama()
35     {
36         return nama;
37     }
38     public String getLisensi()
39     {
40         return lisensi;
41     }
42
43
44
45
46
47 }
```

➤ **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 **Game.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.

```

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)
9     {
10         super(kode,nama,lisensi);
11     }
12
13     Game (String kode, String nama, String lisensi, String jenis, String tipe)
14     {
15         super.setKode(kode);
16         super.setNama(nama);
17         super.setLisensi(lisensi);
18
19         this.jenis = jenis;
20         this.tipe = tipe;
21     }
22
23     //getter
24
25 }
    
```

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).

```

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     public void setKode(String k)
20     {
21         kode = k;
22     }
23     public void setNama(String n)
24     {
25         nama = n;
26     }
27     public void setLisensi(String l)
28     {
29         lisensi = l;
30     }
31
32     //getter
33
34 }
    
```

```

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)
9     {
10         super(kode,nama,lisensi);
11     }
12
13     Game (String kode, String nama, String lisensi, String jenis, String tipe)
14     {
15         super.setKode(kode);
16         super.setNama(nama);
17         super.setLisensi(lisensi);
18
19         this.jenis = jenis;
20         this.tipe = tipe;
21     }
22
23     //getter
24
25 }
    
```

The diagram illustrates the method resolution process. Red arrows originate from the `super(kode,nama,lisensi);` call in the `Game` class constructor (line 10) and point to the `Software(String kode, String nama, String lisensi)` constructor (line 11) in the `Software` class. Another set of red arrows points from the `super.setKode(kode);`, `super.setNama(nama);`, and `super.setLisensi(lisensi);` calls in the `Game` class constructor (lines 15-17) to their respective setter methods in the `Software` class (lines 19-21).

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)
9
10    Game (String kode, String nama, String lisensi, String jenis, String tipe)
11
12
13
14
15
16
17
18
19
20
21
22
23
24 //getter
25 public String getKode()
26 {
27     return super.getKode();
28 }
29
30 public String getNama()
31 {
32     return super.getNama();
33 }
34
35 public String getLisensi()
36 {
37     return super.getLisensi();
38 }
39
40 public String getJenis()
41 {
42     return jenis;
43 }
44 public String getTipe()
45 {
46     return tipe;
47 }
```

Seperti halnya penggunaan Constructor kedua pada class Game yang menggunakan `super.[nama_method]`, maka untuk method `getKode()`, `getNama()`, dan `getLisensi()` menggunakan keyword “*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         //instance of class
8         //menu
9         //input
10        //proses + output
11    }
12 }
13
14
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            System.out.println();
29
30            //proses + output
31        }
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         gameA = new Game(kode, nama, lisensi);
47         System.out.println();
48         System.out.println("### Data Game A telah disimpan ###");
49         break;
50
51
52

```

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, 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     gameB = new Game(kode, nama, lisensi, jenis, tipe);
71     System.out.println();
72     System.out.println("### Data Game B telah disimpan ###");
73     break;
74
75
76

```

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;
```