MODUL PRAKTIKUM

PEMROGRAMAN JAVA

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN ILMU KOMPUTER

E L R A H M A

YOGYAKARTA

2008

BAB 1 VARIABEL DALAM JAVA

I. Tujuan Praktikum:

1. Mengetahui tipe-tipe variabel dalam Java.
2. mengetahui cara pendeklarasian variabel dalam Java.
3. Mengetahui dan memahami penamaan variabel dan scope variabel dalam Java..

II. Dasar Teori

Variabel merupakan *container* yang digunakan untuk menyimpan suatu nilai pada sebuah program dengan tipe tertentu. Pada dasarnya ada dua macam tipe variabel data dalam bahasa Java, yakni tipe primitif dan tipe *reference*. Adapun tipe primitif meliputi:

- Tipe boolean
- Tipe numerik, yang meliputi:
  - byte
  - short
  - int
  - long
  - char
  - float
  - double

Sedangkan tipe data variabel berupa *reference* terdiri atas tipe variabel data:

- Tipe class
- Tipe array
- Tipe interface

Adapula tipe variabel data yang khusus yang disebut *null types*, namun variabel dalam java tidak akan pernah memiliki tipe *null* ini.

Pendeklarasian sebuah variabel dalam Java menggunakan sintaks sebagai berikut:

<tipe_variabel> <nama_variabel>

Contohnya:

```
int dataInt;
char charData;
float x;
```

Java memungkinkan untuk menginisialisasi nilai sebuah variabel saat dideklarasikan dengan cara seperti ini:

```
int dataInt = 10;
char charData = '\u103';
float x =12.67;
```

Variabel dalam Java dapat dijadikan konstanta, sehingga nilainya tidak akan dapat diubah-ubah dengan mendeklarasikannya sebagai variabel *final* seperti ini:

```
final int dataInt =10;
final char dataChar = '\u103';
final float x = 12.67;
final byte y = 2;
```

Penamaan variabel dalam Java harus memenuhi aturan sebagai berikut:

- ☑ Harus terdiri atas sederetan karakter unicode yang diawali oleh karakter huruf atau garis bawah. Unicode merupakan sistem pengkodean karakter yang dapat dibaca oleh berbagai bahasa manusia. Ada maksimum 65.536 karakter yang dapat digunakan. Jika pada ASCII hanya terbatas pada karakter latin, maka dengan unicode kita mampu membaca karakter dengan jenis alfabet seperti Jepang, Yunani, Cyrillic, dan Hebrew. Namun, saat ini karakter unicode yang sudah terdefinisi baru sekitar 34.168 karakter.
- ☑ Tidak boleh berupa keyword (kata yang dicadangkan), null, atau literal true/false.
- ☑ Harus unik dalam satu scope.

Contoh pendeklarasian nama variabel yang salah:

```
int 8k;          /* salah, karena nama diawali dengan angka */
char null;       /* salah, karena nama variabel = null */
String public;   /* salah, karena nama = public yang merupakan keyword dalam Java */
```

Contoh pendeklarasian nama variabel yang benar:

```
int k8;          /* benar, karena nama tidak diawali dengan angka */
char Nnull;      /* benar, karena nama bukan true, false, null, atau keyword */
String Spublic;  /* benar, karena nama variabel bukan keywors */
```

Scope sebuah variabel harus diperhatikan dengan baik pada Java, karena program bisa saja menjadi error hanya karena urusan *scope* variabel yang tidak benar.

Lebih jauh lagi tentang Tipe Variabel dalam Java dapat dijelaskan sebagai berikut:

✍ **Tipe Boolean**

Dalam Java, setiap variabel yang bertipe *boolean* hanya akan memiliki nilai *true* atau *false*. Contoh:

```
class myBool {
        public static void main(String [] args) {
                boolean nilai1 = true;
                boolean nilai2 = false;

                System.out.println("Boolean 1 = " + nilai1);
                System.out.println("Boolean 1 = " + nilai1);
        }
  }
```

✍ **Tipe Numerik**

Adapun tipe numerik dalam Java ada tujuh macam seperti yang telah dijelaskan sebelumnya, yakni:

Nama Tipe	*Ukuran*	*Range nilai*
byte	8 bit	-128 . . . 127
short	16 bit	-32768 . . . 32767
int	32 bit	-2147483648 . . . 2147483647
long	64 bit	-9223372036854775808 . . . 9223372036854775807
char	16 bit	0 ... 65535 ('\u0000' ... '\uffff')
float	32 bit IEEE 754	
double	64 bit IEEE 754	

Selain adanya variabel-variabel di atas, juga ada kelas-kelas yang disediakan oleh Java API untuk tipe variabel di atas yakni **Boolean, Byte, Short, Integer, Long, Character, Float, Double**. Kelas-kelas ini secara default diimpor oleh Java saat kompilasi, namun secara eksplisit dapat diimport dari *package* **java.lang**.

BAB 2 OPERATOR DALAM JAVA

I. Tujuan Praktikum:

1. Mengetahui operator-operator yang dipakai dalam Java.
2. Mengetahui urutan presedence operator dalam Java.
3. Dapat menggunakan operator-operator dalam Java.

II. Dasar Teori

Macam-macam operator dalam Java sebagai berikut:

✍ Operator Aritmatika

Sama halnya dengan semua bahasa pemrograman, Java menyediakan operator-operator aritmatika untuk manipulasi variabel data numerik. Tabel penggunaan dan jenis operator aritmatika dalam Java bisa dilihat di bawah ini:

Operator	*Penggunaan*	*Deskripsi*
+	Op1 + Op2	Menambahkan Op1 dengan Op2
-	Op1 + Op2	Mengurangkan Op1 dengan Op2
*	Op1 * Op2	Mengalikan Op1 dengan Op2
/	Op1 / Op2	Membagi Op1 dengan Op2
%	Op1 % Op2	Menghasilkan sisa hasil bagi antara OP1 dengan Op2

Selain operator di atas, operator berikut juga termasuk dalam operator aritmatika:

Operator	*Penggunaan*	*Deskripsi*
++	Op++	Op dinaikkan nilainya 1 setelah dilakukan operasi pada Op
++	++Op	Op dinaikkan nilainya 1 sebelum dilakukan operasi pada Op
--	Op--	Op diturunkan nilainya 1 setelah dilakukan operasi pada Op
--	--Op	Op diturunkan nilainya 1 setelah dilakukan operasi pada Op
-	-Op	Mengubah nilai Op menjadi negasinya, jika Op positif maka menjadi negatif, jika Op negatif menjadi positif

✍ Operator Relasional

Operator relasional dalam Java dapat digunakan untuk membandingkan antara dua variabel data, lebih lengkapnya dapat dilihat dalam tabel berikut ini:

Operator	*Penggunaan*	*Deskripsi*
>	Op1 > Op2	Menghasilkan **true** jika Op1 lebih besar dari Op2
<	Op1 < Op2	Menghasilkan **true** jika Op1 lebih kecil dari Op2
>=	Op1 >= Op2	Menghasilkan **true** jika Op1 lebih besar atau sama dengan Op2
Operator	*Penggunaan*	*Deskripsi*
<=	Op1 <= Op2	Menghasilkan **true** jika Op1 lebih kecil atau sama dengan Op2

==	Op1 == Op2	Menghasilkan **true** jika Op1 sama dengan Op2
!=	Op1 != Op2	Menghasilkan **true** jika Op1 tidak sama dengan Op2

Operator Kondisional

Operator kondisional dalam Java ada enam macam sebagaimana digambarkan dalam tabel berikut ini:

Operator	*Penggunaan*	*Deskripsi*
&&	Op1 && Op2	Menghasilkan **true** jika Op1 dan Op2, keduanya bernilai *boolean* **true**
||	Op1 || Op2	Menghasilkan **true** jika Op1 atau Op2, salah satunya bernilai *boolean* **true**
!	Op1 ! Op2	Menghasilkan **true** jika Op1 bernilai *boolean* **false**
&	Op1 & Op2	Bitwise AND, menghasilkan **true** jika Op1 dan Op2 bernilai **true**
|	Op1 | Op2	Bitwise OR, Menghasilkan **true** jika Op1 atau Op2 salah satunya bernilai **true**
^	Op1 ^ Op2	Menghasilkan **true** jika salah satu diantara Op1 dan Op2 bernilai **true**, namun tidak keduanya

Operator Shift

Operator *shift* dalam Java digunakan untuk manipulasi bit. Operator *shift* digunakan untuk menggeser bit-bit sesuai yang diinginkan. Ada tiga operator *shift* dalam Java, yakni:

Operator	*Penggunaan*	*Deskripsi*
>>	Op1 >> Op2	Menggeser bit Op1 ke kanan sejauh Op2
<<	Op1 << Op2	Menggeser bit Op1 ke kiri sejauh Op2
>>>	Op1 >>> Op2	Menggeser bit Op1 ke kanan sejauh Op2

Operator Bitwise

Operator *bitwise* dalam Java juga digunakan untuk memanipulasi bit.

Operator	*Penggunaan*	*Deskripsi*
&	Op1 >> Op2	Menggeser bit Op1 ke kanan sejauh Op2
<<	Op1 << Op2	Menggeser bit Op1 ke kiri sejauh Op2
>>>	Op1 >>> Op2	Menggeser bit Op1 ke kanan sejauh Op2

Bitwise AND

Bitwise AND akan menghasilkan bit "1" jika kedua operator bernilai bit "1"

Op1	*Op2*	*Op1 & Op2*
0	0	0
Op1	*Op2*	*Op1 & Op2*
0	1	0
1	0	0
1	1	1

- ***Bitwise OR***

  Bitwise OR akan menghasilkan bit "1" jika salah satu operator bernilai bit "1"

Op1	*Op2*	*Op1 | Op2*
0	0	0
0	1	1
1	0	1
1	1	1

- ***Bitwise XOR (Exclusive OR)***

  Bitwise XOR akan menghasilkan bit "1" jika kedua operator memiliki nilai bit yang berbeda.

Op1	*Op2*	*Op1 | Op2*
0	0	0
0	1	1
1	0	1
1	1	0

- ***Bitwise Complement***

  Bitwise Complement akan menghasilkan bit yang berlawanan dengan bit yang dioperasikan.

Op	*~Op*
0	0
1	0

- **Operator Assignment**

Operator *assignment* dalam Java digunakan untuk memberikan sebuah nilai ke sebuah variabel. Operator *assignment* hanya berupa '=', namun selain itu dalam Java dikenal beberapa shortcut assignment operator yang penting, yang digambarkan dalam tabel berikut:

Operator	*Penggunaan*	*Ekivalen dengan*
+=	Op1 += Op2	Op1 = Op1 + Op2
-=	Op1 -= Op2	Op1 = Op1 – Op2
*=	Op1 *= Op2	Op1 = Op1 * Op2
Operator	*Penggunaan*	*Ekivalen dengan*
/=	Op1 /= Op2	Op1 = Op1 / Op2
%=	Op1 %= Op2	Op1 = Op1 % Op2
&=	Op1 &= Op2	Op1 = Op1 & Op2
|=	Op1 |= Op2	Op1 = Op1 | Op2
^=	Op1 ^= Op2	Op1 = Op1 ^ Op2
<<=	Op1 <<= Op2	Op1 = Op1 << Op2
>>=	Op1 >>= Op2	Op1 = Op1 >> Op2
>>>=	Op1 >>>= Op2	Op1 = Op1 >>> Op2

Operasi-operasi yang menggunakan operator dapat melibatkan lebih dari satu operator dan satu operand. Adapun urutan precedence operator dalam Java secara lengkap sebagai berikut:

postfix operators	: [] . (params) expr++ expr--
unary operators	: ++expr --expr +expr –expr ~ !
creation or cast	: new (type) expr
multiplicative	: * / %
additive	: + -
shift	: << >> >>>
rerlational	: < > <= >= instanceof
equality	: == !=
bitwise AND	: &
bitwise exclusive OR	: ^
bitwise inclusive OR	: |
logical AND	: &&
logical OR	: ||
conditional	: ? :
assignment	: = += -= *= /= %= &= ^= |= <<= >>= >>>=

BAB 3 CONTROL FLOW DALAM JAVA

I. Tujuan Praktikum:

1. Mengetahui jenis-jenis Control Flow dalam Java.
2. Mengetahui dan memahami sintaks-sintaks dari operasi pengulangan dan operasi kondisional dalam Java.
3. Dapat mengimplementasikan Control Flow dalam program.

II. Dasar Teori

Operasi Pengulangan

☺ ***While***

Pernyataan perulangan *while* akan menguji sebuah persyaratan, dan kemudian menjalankan sekumpulan pernyataan jika persyaratan terpenuhi. Usai menjalankan kumpulan pernyataan, persyaratan akan diuji kembali, dan jika terpenuhi kumpulan peryataan akan dijalankan kembali. demikian seterusnya.

Tata cara penulisan *while* adalah:

```
while (persyaratan terpenuhi){
      pernyataan-pernyataan;
}
```

Contoh sederhana penggunaan *while* adalah sebagai berikut:

```
public class LatWhile {
      public static void main(String args[]) {
            int bilangan = 5;

            int x = 2;
            int hasil = 1;

            while(x<=bilangan) {
                  System.out.print(hasil + " * " + x + " = ");
                  hasil = hasil*x;
                  System.out.println(hasil);
                  x++;
            }

            System.out.println(bilangan + " != " + hasil);
        }
}
```

☺ ***Do-While***

Pernyataan perulangan *do-while* akan menjalankan sekumpulan pernyataan-pernyataan, dan kemudian mengulanginya lagi selama persyarataan terpenuhi.

Tata cara penulisan *do-while* adalah:

```
do {
      pernyataan-pernyataan;
}
while (persyaratan terpenuhi)
```

Contoh penggunaan *do-while*:

```
public class LatDoWhile {
      public static void main(String args[]) {
            int bilangan = 5;

            int x = 2;
            int hasil = 1;

            do {
                  System.out.print(hasil + " * " + x + " = ");
                  hasil = hasil*x;
                  System.out.println(hasil);
                  x++;
            }
            while(x<=bilangan);
            System.out.println( bilangan + " != " + hasil);
      }
}
```

☺ ***For***

Pernyataan perulangan *for* akan menjalankan sekumpulan pernyataan-pernyataan, dan kemudian mengulanginya lagi selama persyarataan terpenuhi.

Tata cara penulisan *for* adalah:

```
for (keadaan-awal, persyaratan, pernyataan-perulangan) {
      pernyataan-pernyatan;
}
```

Perulangan *for* akan menjalankan pernyataan-pernyataan mulai dari keadaan awal, selama persyaratan terpenuhi. Usai menjalankan pernyataan-pernyataan, pernyataan-perulangan akan dijalankan. Selanjutnya perulangan dimulai kembali dengan menguji persyaratan. Misalnya:

```
for (int i=0; i<=3; i++) {
      System.out.println(i);
}
```

Maka perulangan akan dimulai dengan variable i berharga 0. Selanjutnya karena persyaratan i<=3 terpenuhi, maka pernyataan System.out.println(i) akan dijalankan. Akhirnya pernyataan i++ dijalankan. Dan kemudian persyaratan i<=3 diuji lagi. Demikian seterusnya.

Contoh berikut ini akan menjelaskan bagaimana penggunaan for dalam Java:

```
public class LatFor {
     public static void main(String args[]) {
        int bilangan = 5;

        int hasil = 1;
        for(int iterator=2;iterator<=bilangan;iterator++) {
             System.out.print(hasil + " * " + iterator + " = ");
             hasil *= iterator;
             System.out.println(hasil);
        }
          System.out.println(bilangan + " != " + hasil);
     }
}
```

Operasi Kondisional

☺ ***If-Else***

Pernyataan kendali aliran *if* akan menguji sebuah keadaan, apakah keadaan tersebut *true* atau *false*. Jika keadaan tersebut *true* maka kumpulan pernyataan yang berkaitan akan dijalankan, dan jika *false* maka tidak dijalankan.

Tata cara penulisan *if* adalah :

```
if (keadaan) {
      pernyataan-pernyataan;
}
```

Misal:

```
if (x<0) {
      System.out.println ("x adalah kurang dari 0");
}
```

Maka jika x < 0, program akan menuliskan pesan "x adalah kurang dari 0", jika x >= 0 maka program tidak melakukan apa-apa.

Kendali aliran *if* dapat mempunyai bagian *else*, dengan penulisan:

```
if (keadaan) {
      pernyataan-pernyataan;
} else {
      pernyataan-pernyataan lain;
}
```

Dalam hal ini, pernyataan-pernyataan lain akan dijalankan jika keadaan adalah *false*. Misalnya:

```
if (x<0) {
      System.out.println ("x adalah kurang dari 0");
} else {
      System.out.println ("x tidak kurang dari 0");
}
```

Contoh program java dengan menggunakan *if-else* sebagai berikut:

```
public class LatIfElse {
    public static void main(String[] args) {
        double a = 4;
        double b = -13;
        double c = -12;

        double D = b*b - 4*a*c;
        double x1, x2;

        System.out.println("Persamaan kuadrat a*x^2+b*x+c=0, dimana");
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
        System.out.println("dengan menerapkan rumus ABC diketahui bahwa :
                           ");

        if(D<0) {
            System.out.println("Akar-akar persamaan nyata tetapi kembar.");
            x1 = -b/(2*a);
            System.out.println("x1 = x2 = " + x1);
        }
        Else {
            System.out.println("Akar-akar persamaan nyata dan berbeda.");
            x1 = (-b+Math.sqrt(D))/(2*a);
            x2 = (-b-Math.sqrt(D))/(2*a);
            System.out.println("x1 = " + x1 + "                                    x2 =
       }
    }
}
```

☺ ***Switch-Case***

Pernyataan kendali aliran *switch* digunakan untuk mengeksekusi sebuah pernyataan jika memenuhi suatu keadaan tertentu.

Tata cara penulisan *switch-case* adalah :

```
switch (variable) {
      case keadaan1     : pernyataan-1;
                          break;
      case keadaan2     : pernyataan-2;
                          break;
      case keadaan3     : pernyataan-3;
                          break;
```

```
            default           : pernyataan-default;
      }
```

Keadaan default bersifat opsional, artinya boleh ada atau tidak ada, yang berfungsi sebagai else pada kasus if-else dengan banyak percabangan, yaitu ketika ditemui keadaan yang tidak memenuhi dari semua keadaan sebelumnya.

Penulisan *switch-case* di atas dapat dipandang sebagai:

```
if (variabel==keadaan1) {
      pernyataan-1;
}
else if (variabel==keadaan2) {
      pernyataan-2;
}
else if (variabel==keadaan3) {
      pernyataan-3;
}
else {
      pernyataan-default;
}
```

Dalam kendali aliran *switch-case*, variabel penentu harus bertipe integer atau yang kompatibel dengan integer, yaitu boolean, character, byte, short dan long. Perhatikan contoh program Java berikut untuk lebih jelasnya:

```
public class LatSwitchCase {
      public static void main(String[] args) {
            char inisial = 'M';
            String nama = null;
            Switch (inisial) {
                  case 'A' : nama = "Agung"; break;
                  case 'E' : nama = "Eko"; break;
                  case 'F' : nama = "Fikri"; break;
                  case 'M' : nama = "Millati"; break;
                  case 'S' : nama = "Syarif"; break;
                  case 'T' : nama = "Toosa"; break;
                  case 'Y' : nama = "Yudo"; break;
                  default  : nama = "Tak dikenal";
            }
            System.out.println("Nama Anda adalah " + nama);
      }
}
```

Break, Label, Continue dan Return

Statement *break* ada dua macam, yakni:

- Break with label
- Break without label

Penggunaan *break* tanpa *label* umumnya dalam *switch*, seperti yang sudah dicontohkan sebelumnya, dan dalam pengulangan seperti *for*, *do-while*, dan *while*. Fungsi *break* adalah untuk menghentikan eksekusi sebuah pernyataan program, misalnya untuk memaksa keluar dari sebuah *looping*.

Label pada program Java hanya merupakan penanda yang berupa:

```
NamaLabel:
```

Penggunaan *break* dengan *label* adalah pada pengulangan yang berkalang, yakni pengulangan dalam pengulangan, yang mungkin berada dalam pengulangan yang lain, dan seterusnya, menyebabkan setelah eksekusi sebuah pernyataan terhenti, aliran eksekusi akan berlanjut ke posisi pengulangan tempat *label* berada. Artinya setelah *break*, maka eksekusi program akan menuju ke *label* yang didefinisikan.

Berbeda dengan *break*, *continue* digunakan untuk melewatkan eksekusi pada pengulangan yang ada, dan melanjutkan ke pengulangan berikutnya. Pada *continue*, juga dapat digunakan *label* dengan aturan yang sama dengan *break*.

Pernyataan *return* digunakan untuk keluar dari sebuah fungsi, dan dapat diberikan parameter nilai untuk menunjukkan *return value* dari sebuah fungsi.

BAB 4 PENGENALAN OBJEK

I. Tujuan Praktikum:

1. Mengetahui dan memahami konsep Pemrograman Berorientasi Objek.
2. Mampu membedakan konsep pemrograman tradisional dengan Pemrograman Berorientasi Objek.

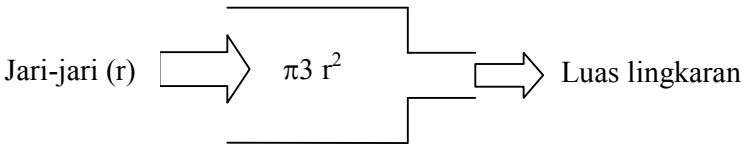
II. Dasar Teori

Pemrograman Berorientasi Objek merupakan pemrograman yang menitikberatkan pada keterkaitan yang erat erat antara proses dengan objek yang diproses. Konsep yang digunakan dalam OOP (Object Oriented Programming) berbeda dengan metode pemrograman procedural. Terdapat beberapa aspek yang harus diketahui sehubungan dengan metode OOP. Diantaranya adalah objek, class, inheritance, dan lain-lain. Dalam OOP dibutuhkan memori lebih banyak dibandingkan dengan jika kita membuat program dengan metode tradisional. Dua buah objek yang identik akan memerlukan dua area memori berbeda walaupun dari sisi data dan proses keduanya memiliki jumlah dan jenis yang sama. Hal ini disebabkan karena data dan proses pada kedua objek tersebut dipisahkan oleh computer.

Dalam praktikum Pemrograman Berorientasi Objek (Object Oriented Programming) ini digunakan bahasa pemrograman JAVA.

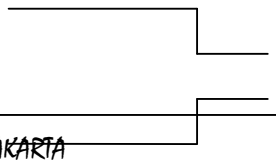
Objek

Terdapat fungsi untuk menghitung luas lingkaran dengan rumus yang sudah dikenal luas : $\pi . r^2$. Notasi 'π' atau phi adalah konstanta sedangkan 'r' adalah variable, sehingga hasil perhitungan akan sangat ditentukan oleh besar kecilnya nilai 'r', atau dengan kata lain hasil perhitungan berbanding lurus dengan perubahan nilai 'r'.

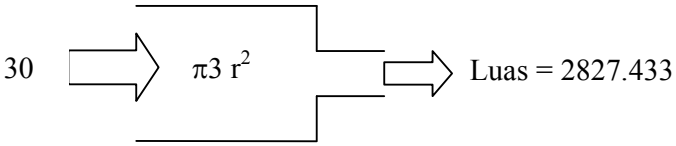


Fungsi menghitung luas lingkaran ini membutuhkan data jari-jari lingkaran. Sebagai contoh akan dihitung luas dari 2 lingkaran dengan nilai jari-jari masing-masing 7 dan 30. Perhitungan luas kedua lingkaran tersebut adalah sebagai berikut (pendekatan 3 angka dibelakang koma):

data jari-jari 7 ----



data jari-jari 30 ----



Melalui fungsi yang sama bisa diperoleh hasil perhitungan yang berbeda-beda tergantung data masukan yang diberikan. Maka, jika terdapat perubahan rumus menghitung luas lingkaran, perubahan tersebut akan berdampak pada dua lingkaran tersebut. Bila setiap lingkaran dianggap sebagai "objek" yang d dalamnya terdapat data jari-jari lingkaran dan fungsi untuk menghitung luas lingkaran, maka untuk menghitung luas beberapa lingkaran dibutuhkan "beberapa objek sejenis". Setiap objek bertanggung jawab terhadap data dan fungsi di dalamnya. Tidak ada hubungan antara objek yang satu dengan objek lainnya.

Objek merupakan kesatuan antara data dan fungsi yang memproses data tersebut.

BAB 5 CLASSIFICATION

I. Tujuan Praktikum :

1. Memahami Classification (class) dalam konsep Pemrograman Berorientasi Objek.
2. Dapat membuat Class.
3. Dapat membedakan antara Class dengan Objek.

II. Dasar Teori

Classification adalah suatu proses pembentukan class. Class merupakan cikal-bakal objek, karena merupakan mekanisme untuk mengelompokan dan mengidentifikasi objek. Selain itu class merupakan aspek penting yang harus diketahui dalam Pemrograman Berorientasi Objek. Proses pembuatan class harus dilakukan sedemikian rupa sehingga seluruh data yang diperlukan oleh objek bisa didaftarkan. Untuk class yang kompleks hal ini tentu merupakan pekerjaan yang rumit, namun konsep OOP menawarkan kemudahan dengan cara mengembangkan class yang telah kita buat. Bisa saja dibuat suatu class yang merupakan pengembangan dari class yang telah ada sebelumnya. Secara umum sebuah class Java didefinisikan dengan format sebagai berikut:

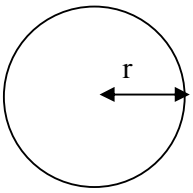
```
class  NamaClass {
       definisi_data_member;

       definisi_member_function;
}
```

Perbedaan Class dan Objek:

- ★ **Class merupakan desain dan objek merupakan perwujudan suatu class.**
- ★ **Class bersifat abstrak dan objek bersifat kongkrit.**

Abstraction

Abstraction adalah suatu proses dimana kita melakukan desain class dan menentukan data dan method yang akan dimiliki oleh sebuah class. Method digunakan untuk mengkomunikasikan data dalam class dengan lingkunga luar class, misalnya untuk mengetahui berapa nilai jari-jari lingkaran saat ini kita menyediakan method khusus. Pengaksesan data objek secara langsung tidak diperbolehkan. Sebuah bangun geometri Lingkaran dideskripsikan dengan bentuk seperti berikut ini:



Lingkaran memiliki jari-jari (r) untuk menyatakan ukurannya. Nilai jari-jari sebuah lingkaran bersifat bebas, tidak bergantung pada nilai jari-jari lingkaran lain. Jadi jari-jari adalah data yang dimiliki oleh sembarang lingkaran, dan inilah yang menjadi karakteristik bangun geometri lingkaran.

Format class Lingkaran berdasar karakteristik di atas adalah sebagai berikut:

```
class Lingkaran {
      data: jari-jari

      method: update jari-jari
              informasi jari-jari
              hitung luas
              hitung keliling
}
```

Encapsulation

Data yang ada pada suatu objek tidak boleh diubah secara langsung adri luar objek tersebut. Harus ada mekanisme untuk mengubah data objek dan menginformasikan data tersebut ke lingkungan luar. Proteksi terhadap data objek harus diiringi dengan mekanisme untuk mengakses data tersebut.

Merujuk pada contoh lingkaran, nilai jari-jari haruslah dinamis, tidak boleh statis. Untuk menjaga validitas nilai jari-jari (tidak bernilai nol atau negative), harus dibuat method sebagai mekanisme untuk memeriksa validitas data jari-jari yang baru masuk.

Dalam kode program terdapat kata kunci “private” yang digunakan untuk mengunci data atau method agar tidak terlihat dari luar objek, serta kata kunci “public” yang digunakan untuk mem-publish data atau method agar dikenal dari luar objek. Berikut contoh programnya:

```
class Lingkaran {
      private double jarijari;
      public void setjarijari (double r) {
            if (r ∃ 0) jarijari = r;
      }
      public double getjarijari () {
            return (jarijari);
      }
      public double hitungLuas() {
            return (Math.PI * jarijari * jarijari);
      }
      public double hitungKeliling() {
            return (Math.PI * 2.0 * jarijari);
      }
}
```

Kode program di atas tidak dapat dijalankan melainkan hanya dapat di compile tanpa kesalahan, karena kode-kode tersebut hanya merupakan implementasi class. Hal seperti ini disebut dengan *modul*.

BAB 6 CONSTRUCTOR

I. Tujuan Praktikum:

1. Memahami dan mengetahui makna Constructor serta karakteristiknya.
2. Dapat membuat constructor dalam membuat program menggunakan bahasa Java.

II. Dasar Teori

Constructor adalah sebuah method khusus dalam metode OOP yang akan dipanggil secara otomatis pada saat sebuah objek diciptakan. Kita tidak perlu memanggil constructor ini, karena Java-lah yang akan melakukannya. Ini berarti proses pemberian nilai awal seluruh data objek dilakukan melalui constructor. Isi constructor dapat berupa pemanggilan method internal atau instruksi lain.

Single Constructor

Single constructor menyatakan bahwa hanya ada satu constructor dalam class. Hal ini bisa ditemukan pada class yang sederhana. Berikut diberikan contoh constructor untuk proses inisialisasi nilai jari-jari dengan analog contoh lingkaran sebelumnya, dengan data awal 1 (satu).

```
class Lingkaran {
      private double jarijari;

      public Lingkaran() {
            setjarijari (1);
      }

      public void setjarijari (double r) {
            if (r ∃ 0) jarijari = r;
      }

      public double getjarijari () {
            return (jarijari);
      }

      public double hitungLuas() {
            return (Math.PI * jarijari * jarijari);
      }

      public double hitungKeliling() {
            return (Math.PI * 2.0 * jarijari);
      }
}
```

Kode program di atas juga termasuk modul, karena kode-kode tersebut hanya merupakan implementasi class. Modul tersebut bernama Lingkaran. Untuk bisa dijalankan menjadi sebuah

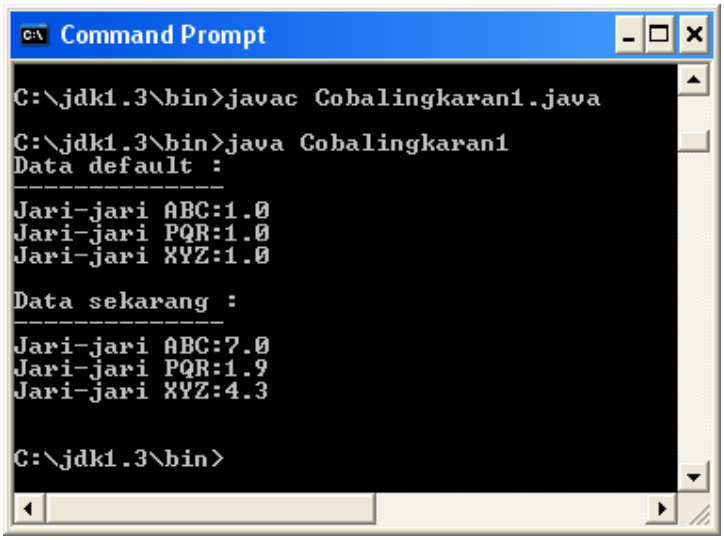
program lengkap, harus dibuat sebuah class lain yang berisi rutin program yaitu class dengan fungsi main(), dengan karakteristik:

- Dideklarasikan sebagai class public
- Nama class menjadi nama file
- Berisi seluruh kode class Lingkaran

```
class Lingkaran {
   private double jarijari;

   public Lingkaran() {
         setjarijari (1);
   }

   public void setjarijari (double r) {
         if (r ∃ 0) jarijari = r;
   }

   public double getjarijari () {
         return (jarijari);
   }

   public double hitungLuas() {
         return (Math.PI * jarijari * jarijari);
   }

   public double hitungKeliling() {
         return (Math.PI * 2.0 * jarijari);
   }
}
public class Cobalingkaran1 {
   public static void main (String[] args) {
         Lingkaran abc = new Lingkaran ();
         Lingkaran pqr = new Lingkaran ();
         Lingkaran xyz = new Lingkaran ();

         System.out.println ("Data default :");
         System.out.println ("--------------");
         System.out.println ("Jari-jari ABC:"+
                              abc.getjarijari());
         System.out.println ("Jari-jari PQR:"+
                              pqr.getjarijari());
         System.out.println ("Jari-jari XYZ:"+
                              xyz.getjarijari());
         System.out.println;
         abc.setjarijari (7);
         pqr.setjarijari (1.9);
         xyz.setjarijari (4.3);

         System.out.println ("Data sekarang :");
         System.out.println ("--------------");
         System.out.println ("Jari-jari ABC:"+
                              abc.getjarijari());
         System.out.println ("Jari-jari PQR:"+
                              pqr.getjarijari());
         System.out.println ("Jari-jari XYZ:"+
```

```
                                        xyz.getjarijari());
            System.out.println;
    }
}
```

Ouput Program :



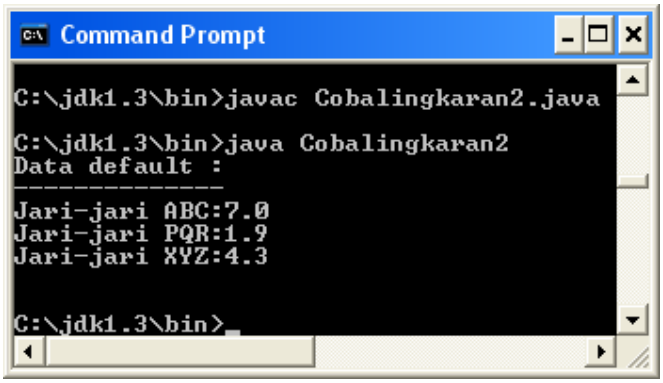
Constructor dengan Parameter

Yaitu constructor dengan data yang bisa kita kirim pada saat penciptaan objek. Dalam model constructor dengan parameter, kita dapat langsung menentukan data jari-jari pada saat penciptaan objek, dengan menyisipkan parameter pada constructor. Metode ini lebih ringkas dari metode sebelumnya (*Single Constructor*). Berikut diberikan program selengkapnya:

```
class Lingkaran {
      private double jarijari;
      public Lingkaran(double r) {
            if (r>0)
                  jarijari = r;
            else
                  jarijari = 1.0;
      }

      public void setjarijari (double r) {
            if (r ∃ 0) jarijari = r;
      }

      public double getjarijari () {
            return (jarijari);
      }
      public double hitungLuas() {
            return (Math.PI * jarijari * jarijari);
      }

      public double hitungKeliling() {
```

```
            return (Math.PI * 2.0 * jarijari);
      }
}
public class Cobalingkaran2 {
      public static void main (String[] args) {
            Lingkaran abc = new Lingkaran (7);
            Lingkaran pqr = new Lingkaran (1.9);
            Lingkaran xyz = new Lingkaran (4.3);

            System.out.println ("Data default :");
            System.out.println ("--------------");
            System.out.println ("Jari-jari ABC:"+
                                abc.getjarijari());
            System.out.println ("Jari-jari PQR:"+
                                pqr.getjarijari());
            System.out.println ("Jari-jari XYZ:"+
                                xyz.getjarijari());
            System.out.println;
      }
}
```

Output Program :



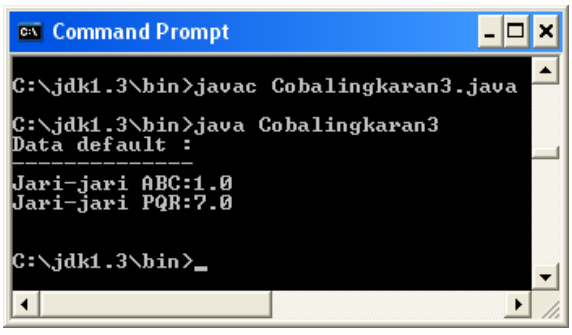
Multiple Constructor

Multiple constructor yaitu terdapat beberapa constructor yang bisa kita pilih pada saat penciptaan objek. Pemakaian beberapa model constructor dalam sebuah program ini dimungkinkan terjadi karena semakin kompleks class yang kita buat akan semakin banyak variasi constructor yang akan terjadi.

```
class Lingkaran {
      private double jarijari;
      public Lingkaran () {
            setjarijari (1);
      }
      public Lingkaran(double r){
            if (r>0)
                  jarijari = r;
            else
                  jarijari = 1.0;
```

```
        }

        public void setjarijari (double r) {
                if (r ∃ 0) jarijari = r;
        }

        public double getjarijari () {
                return (jarijari);
        }

        public double hitungLuas() {
                return (Math.PI * jarijari * jarijari);
        }

        public double hitungKeliling() {
                return (Math.PI * 2.0 * jarijari);
        }
}

public class Cobalingkaran3 {
        public static void main (String[] args) {
                Lingkaran abc = new Lingkaran ();
                Lingkaran pqr = new Lingkaran (7);

                System.out.println ("Data default :");
                System.out.println ("--------------");
                System.out.println ("Jari-jari ABC:"+
                                    abc.getjarijari());
                System.out.println ("Jari-jari PQR:"+
                                    pqr.getjarijari());
                System.out.println;
        }
}
```

Output Program :



Karakterisrik constructor:

★ **Didefinisikan public**
★ **Memiliki nama sama dengan nama class**

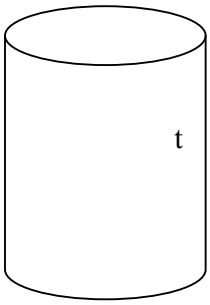
I. Tujuan Praktikum:

1. Mengetahui dan memahami Inheritance dalam Pemrograman Berorientasi Objek.
2. Dapat membedakan dengan class biasa.

II. Dasar Teori

Inheritance atau pewarisan adalah memperluas suatu kelas atau mendefinisikan class baru dengan memanfaatkan class yang telah ada. Metode OOP memungkinkan kita memperoleh seluruh data dari class induk (base-class) untuk diberikan kepada class anak(derived-class) tanpa harus melakukan *copy-paste* seluruh kode base-class.

Sebagai contoh akan dibuat class Silinder. Penggambaran bangun geometri "silinder" adalah sebagai berikut:



Kedua ujung silinder memiliki bentuk lingkaran yang dihubungkan dengan suatu permukaan pada jarak tertentu. Jika jarak kedua silinder disebut tinggi silinder (t) maka dapat disimpulkan bahwa selain 't' silinder juga memiliki 'r', yaitu jari-jari lingkaran pada kedua ujungnya.

Bagan untuk class Silinder dengan seluruh data dan methodnya menurut gambar di atas adalah sebagai berikut:

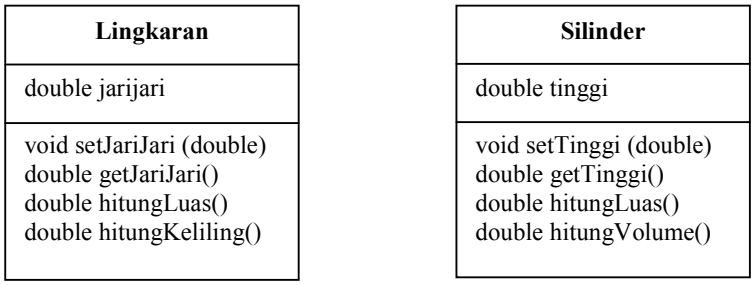
Silinder
double jarijari double tinggi
void setJariJari(double) void setTinggi (double) double getJariJari() double getTinggi() double hitungLuas() double hitungVolume()

Untuk melihat perbandingannya, diberikan bagan kedua bagan, yaitu class Lingkaran dan class Silinder.

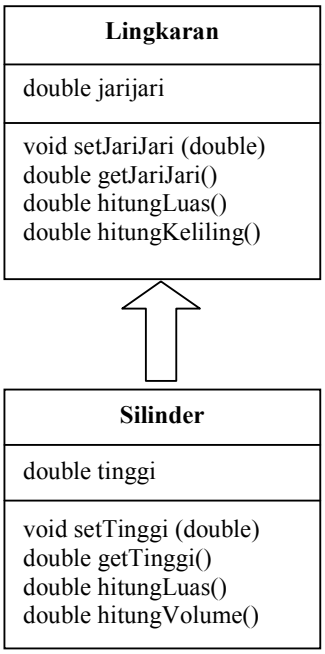
Lingkaran
double jarijari
void setJariJari (double) double getJariJari() double hitungLuas() double hitungKeliling()

Silinder
double jarijari double tinggi
void setJariJari (double) void setTinggi (double) double getJariJari() double getTinggi() double hitungLuas() double hitungVolume()

Terlihat ada data dan method yang sama pada kedua class. Terlihat pula bahwa class Silinder merupakan *superset* dari class Lingkaran. Tidak semua method class Lingkaran dihilangkan pada class Silinder, seperti nampak berikut ini:



Secara hirarki, struktur kedua class tersebut digambarkan sebagai berikut:



Berikut ini merupakan implementasi Java untuk class Silinder

```
class Silinder extends Lingkaran {
      private double tinggi;
      public Silinder () {
            setjarijari (1);
            settinggi (1);
      }

      public Silinder (double r, double t) {
            setjarijari (r);
            settinggi (t);
      }
```

```
      public void settinggi (double t) {
            if (t > 0) tinggi = t;
      }

      public double gettinggi () {
            return (tinggi);
      }

      public double hitungLuas () {
            return (super.hitungLuas () * 2.0 +
                        super.hitungKeliling () * tinggi );
      }

      public double hitungVolume () {
            return (super.hitungLuas () * tinggi);
      }

      public double hitungKeliling () {
            return (Math.PI * 2.0 * jarijari);
      }
}
```

Kata kunci extends menyatakan bahwa suatu class merupakan turunan dari kelas lain. Class yang diturunkan disebut *superclass, baseclass,* atau *parentclass*. Class turunan disebut *subclass, derivedclass,* atau *childclass*. *Pada* baris pertama didefinisikan class Silinder sebagai turunan dari class Lingkaran.

Terdapat dua constructor pada class Silinder, yaitu constructor untuk menginisialisasi data dengan nilai default dan constructor untuk menginisialisasi data jari-jari alas Silinder serta data tinggi Silinder.

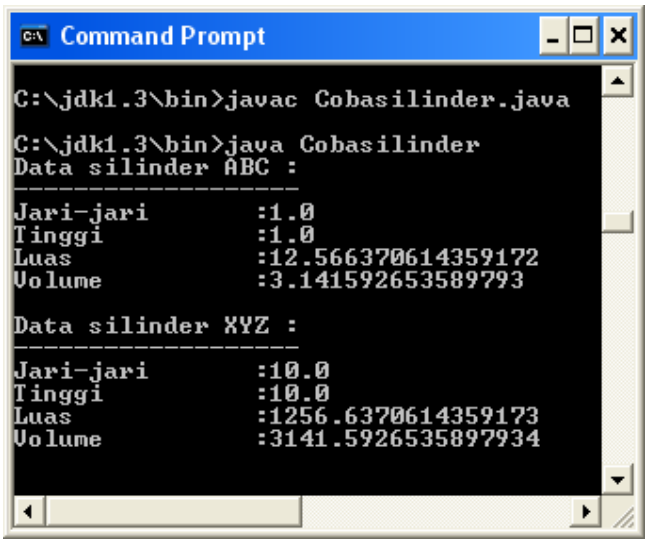
Agar program di atas bisa di-compile, maka seluruh kode class Lingkaran harus disisipkan di atas class Silinder. Untuk dapat menjalankannya harus ditambahkan fungsi main ().

Secara lengkap program ditulis sebagai berikut :

```
class Lingkaran {
      private double jarijari;
      public Lingkaran() {
            setjarijari (1);
      }
      public void setjarijari (double r) {
            if (r ∃ 0) jarijari = r;
      }
      public double getjarijari () {
            return (jarijari);
      }
      public double hitungLuas() {
            return (Math.PI * jarijari * jarijari);
      }
      public double hitungKeliling() {
            return (Math.PI * 2.0 * jarijari);
      }
}
```

```
class Silinder extends Lingkaran {
      private double tinggi;

      public Silinder () {
            setjarijari (1);
            settinggi (1);
      }
      public Silinder (double r, double t) {
            setjarijari (r);
            settinggi (t);
      }
      public void settinggi (double t) {
            if (t > 0) tinggi = t;
      }
      public double gettinggi () {
            return (tinggi);
      }
      public double hitungLuas () {
            return (super.hitungLuas () * 2.0 +
                        super.hitungKeliling () * tinggi );
      }
      public double hitungVolume () {
            return (super.hitungLuas () * tinggi);
      }

}
public class Cobasilinder {
      public static void main (String[] args) {
            Silinder abc = new Silinder ();
            Silinder xyz = new Silinder (10, 10);

            System.out.println ("Data Silinder ABC:");
            System.out.println ("------------------");
            System.out.println ("Jari-jari: "+
                                 abc.getjarijari());
            System.out.println ("Tinggi   : "+
                                 abc.gettinggi());
            System.out.println ("Luas     : "+
                                 abc.hitungLuas());
            System.out.println ("Volume   : "+
                                 abc.hitungVolume());
            System.out.println ();

            System.out.println ("Data Silinder XYZ:");
            System.out.println ("------------------");
            System.out.println ("Jari-jari: "+
                                 xyz.getjarijari());
            System.out.println ("Tinggi   : "+
                                 xyz.gettinggi());
            System.out.println ("Luas     : "+
                                 xyz.hitungLuas());
            System.out.println ("Volume   : "+
                                 xyz.hitungVolume());
            System.out.println ();
      }
}
```

Output Program :



Penulisan super.hitungLuas() menyatakan bahwa nilai yang diambil adalah nilai method hitungLuas() yang ada di class induk. Seandainya kata super dihilangkan maka Java akan menganggap method yang harus dipanggil ada di class Silinder. Kata super digunakan karena method dengan nama hitungLuas() ada di class Silinder dan di class Lingkaran, sehingga harus dipastikan method mana yang akan digunakan. Kecuali untuk tujuan rekursif, penggunaan kata kunci super akan mencecah terjadinya pemanggilan method secara rekursif yang tidak dikehendaki.

Pada program di atas method hitung keliling masih bisa diakses untuk menghitung keliling Silinder. Padahal diketahui bahwa benda tiga dimensi tidak memiliki keliling karena bentuk ruangnya. Dalam kasus seperti ini harua dibatasi pewarisan data atau method agar tidak bisa diakses dari luar namun tetap bisa diwariskan ke kelas dibawahnya. Kata kunci protected digunakan untuk membuat pertahanan baru dengan membatasi akses ke data atau method pada suatu class hanya untuk turunannya saja, tidak termasuk ke bagian lain yang hanya menggunakan class tersebut.

Kata kunci super merujuk kepada class induk satu tingkat di atas class anak

BAB 8 POLYMORPHISM

I. Tujuan Praktikum:

1. Mengetahui apa yang dimaksud dengan *polimorfisma*
2. Memahami cara kerja objek

II. Dasar Teori

Polimorfisma (polymorphism) merupakan kemampuan objek untuk menentukan method yang akan diaplikasikan terhadap objek itu sendiri. Ide dibalik *polimorfisma* adalah pesan yang sama oleh objek mungkin ditanggapi secara berbeda. Secara mudah *polimorfisma* bisa disamakan dengan *method-overloading,* dimana di dalam sebuah class terdapat beberapa method dengan nama yang sama. *Polimorfisma* dapat diaplikasikan pada method yang diturunkan dari superclass.

Kunci agar *polimorfisma* bekerja disebut *late binding*. Artinya, compiler tidak menyebabkan program memanggil method pada saat melakukan kompilasi. Sebaliknya, setiap kali sebuah method didefinisikan dengan objek, compiler menyebabkan program memperhitungkan method mana yang memanggil, dengan menggunakan tipe informasi objek. Proses ini biasanya disebut *binding, dynamic binding,* atau *dynamic dispatch*.

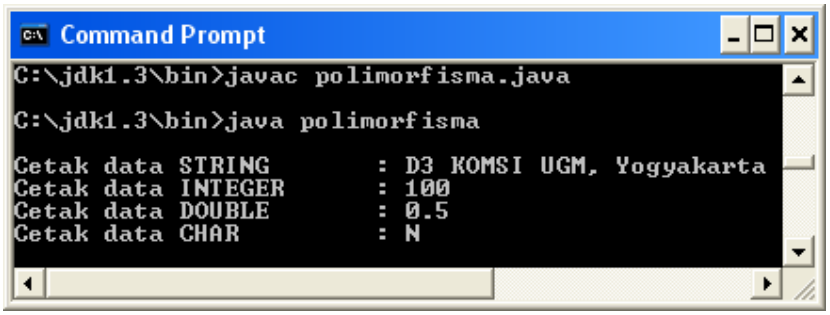
Sebuah method bernama tulisData() yang berfungsi menampilkan data string misalnya, tidak bisa menangani masukan berupa data numeric, Boolean maupun karakter, demikian juga sebaliknya. Solusi yang bisa dilakukan adalah dengan menyediakan beberapa method tulisData() yang akan menangani setiap data, sehingga data apapun yang diberikan sebagai parameter tetap bisa diproses.

Contoh Program :

```
class Tampildata {
      public void tulisData (String data) {
            System.out.println (data);
      }

      public void tulisData (int data) {
            System.out.println (data);
      }

      public void tulisData (double data) {
            System.out.println (data);
      }
      public void tulisData (char data) {
            System.out.println (data);
      }

public class Polymorphism {
      public static void main (String[] args) {
```

```
            TampilData xyz = new TampilData ();
            System.out.println();
            System.out.print ("Cetak data STRING      :");
            xyz.tulisData (" D3 KOMSI UGM, Yogyakarta");
            System.out.print ("Cetak data INTEGER     :");
            xyz.tulisData (100);
            System.out.print ("Cetak data DOUBLE      :");
            xyz.tulisData (1.0 / 2.0);
            System.out.print ("Cetak data CHAR  :");
            xyz.tulisData (' N');
      }
}
```

Output Program :



Method tulisData() akan bereaksi dengan satu dari empat macam cara tergantung dari data yang di*passing* ke dalamnya. Keragaman model aksi ini disebut polymorph atau banyak bentuk.

BAB 9 PACKAGE

I. Tujuan Praktikum:

1. Mengetahui apa yang dimaksud dengan package
2. Dapat membuat package

II. Dasar Teori

Sejauh ini seluruh class yang dibuat harus menyati dengan program utama (class dengan fungsi main()). Sebuah program yang baik haruslah mampu menggunakan modul atau class dari file lain tanpa harus menyalin file tersebut ke dalam file yang sedang di-edit.

Java memungkinkan class-class dikelompokkan dalam suatu kumpulan yang disebut package. Package memberikan kemudahan untuk mengatur pekerjaan dan memisahkan pekerjaan dari kode librari yang disediakan pihak lain. Package dapat disebut sebagai sebuah direktori yang berisi sekumpulan file .class dengan kegunaan spesifik. Berbeda dengan bahasa lain, Java mengorganisasikan package dalam bentuk direktori, dimana dalam direktori ini bisa terdapat satu atau lebih file java yang telah di-*compile* atau dalam bentuk .class. Di dalam direktori tersebut bisa juga terdapat satu atau lebih sub-directory yang memiliki file .class atau sub-directory lain.

Proses pembentukan direktori package tidak sama dengan pembuatan direktori atau folder yang selama ini dikenal. Class yang akan dimasukkan ke dalam package harus mencantumkan nama package di baris pertama. Secara otomatis file hasil compilenya akan dimasukkan kedalam direktori yang tidak lain nama package tersebut.

Instruksi umum meng-compile class Java menjadi package adalah:

```
javac -d <directory_target> <nama_file_java>
```

directory_target tidak sama dengan nama package. Opsi ini digunakan untuk mengarahkan di direktori mana akan dibuat direktori baru sebagai nama package.

Pada pembahasan sebelumnya telah dibuat dua class yaitu Lingkaran dan Silinder. Kedua class ini akan dijadikan package agar program lain yang menggunakannya tidak menyertakan kode kedua class ini di dalamnya.

Berikut modifikasi dari class Lingkaran agar menjadi package:

```
package moduljava;

class Lingkaran {
      private double jarijari;
```

```
	public Lingkaran(double r) {
		if (r>0)
			jarijari = r;
		else
			jarijari = 1.0;
	}

	public void setjarijari (double r) {
		if (r ∃ 0) jarijari = r;
	}

	public double getjarijari () {
		return (jarijari);
	}

	public double hitungLuas() {
		return (Math.PI * jarijari * jarijari);
	}

	public double hitungVolume () {
		return (super.hitungLuas () * tinggi);
	}

	public double hitungKeliling() {
		return (Math.PI * 2.0 * jarijari);
	}
}
```

Kata kunci package digunakan untuk menyatakan nama package yang akan dibuat.

Karakteristik package:
1. **Bersifat public**
2. **Diawali dengan huruf kecil**

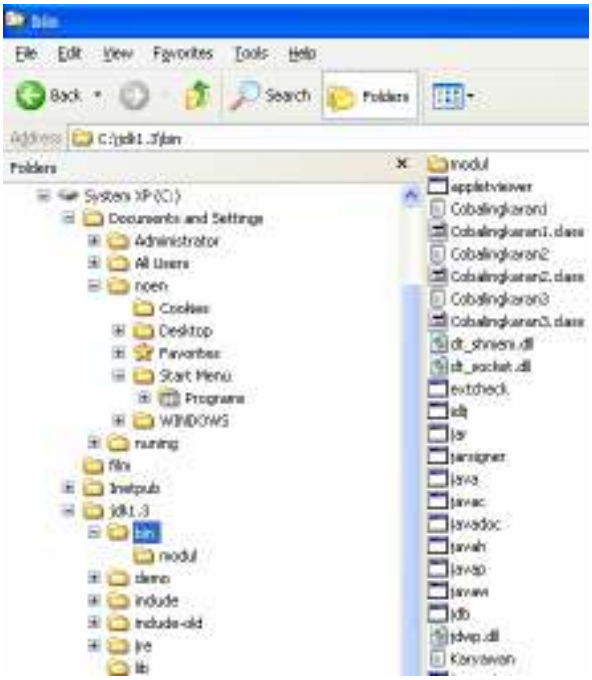
Berikut adalah modifikasi class Silinder agar menjadi package. Beberapa hal yang perlu diperhatikan :

- Class Silinder merupakan turunan dari class Lingkaran
- Class Lingkaran harus sudah di-*compile* terlebih dulu
- Di dalam class Silinder harus ada instruksi yang mengarahkan Java agar mengambil modul Lingkaran dari directory modul.
- Class Silinder harus menyertakan package moduljava agar hasil compilenya disimpan di direktori moduljava. Karena direktori moduljava sudah ada maka penyertaan ini hanya akan menambahkan daftar file .class ke dalam directory moduljava, bukan membuat direktori baru dengan nama moduljava.

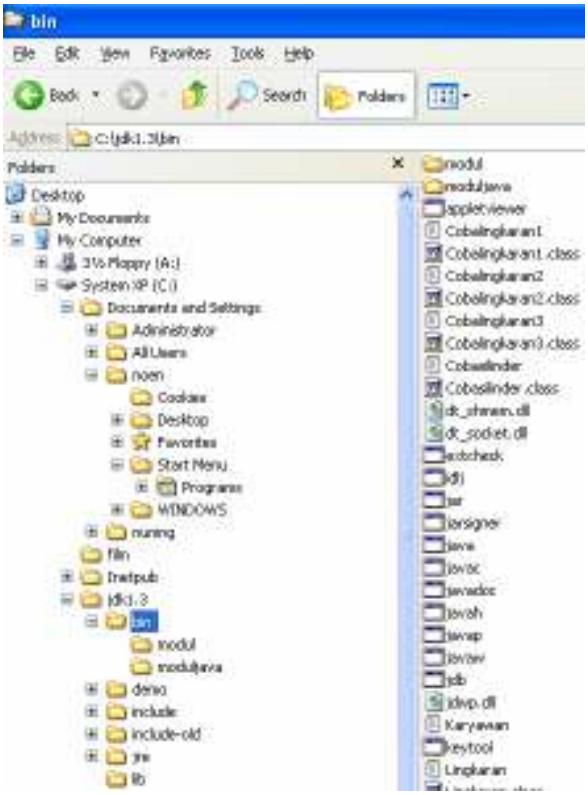
```
package moduljava;

import modul.Lingkaran;
```

```
class Silinder extends Lingkaran {
   private double tinggi;
   public Silinder () {
         setjarijari (1);
         settinggi (1);
   }

   public Silinder (double r, double t) {
         setjarijari (r);
         settinggi (t);
   }

   public void settinggi (double t) {
         if (t > 0) tinggi = t;
   }

   public double gettinggi () {
         return (tinggi);
   }

   public double hitungLuas () {
         return (super.hitungLuas () * 2.0 +
                     super.hitungKeliling () * tinggi );
   }

   public double hitungVolume () {
         return (super.hitungluas () * tinggi)
   }

   public double hitungkeliling () {
         return (Math.PI * 2.0 * jarijari);
   }
}
```

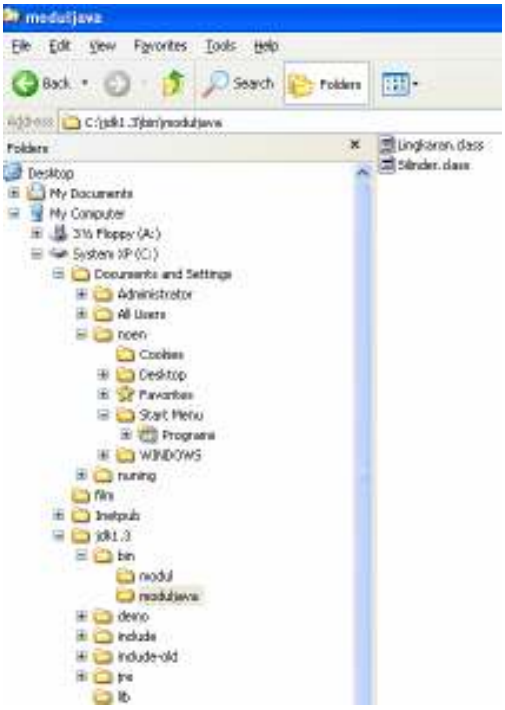
Tampilan berikut merupakan kondisi direktori tempat program-progam java diletakkan sebelum file package di-*compile*.



Setelah file package moduljava di-*compile* didalam direktori terdapat folder baru dengan nama moduljava



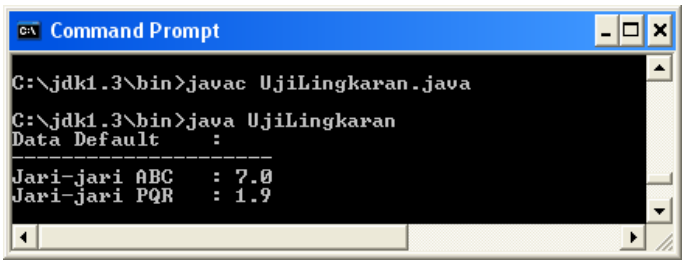
Sekarang dalam direktori moduljava terdapat dua file .class. setiap file java yang didesain sebagai package moduljava akan menambah daftar file ini.



Implementasi penggunaan sebuah package tidak sulit. Cukup dengan meng-import package yang dikehendaki. Berikut adalah contoh penggunaan class Lingkaran.

```
import moduljava.Lingkaran;

public class UjiLingkaran {
      public static void main (String[] args) {
            Lingkaran abc = new Lingkaran (7);
            Lingkaran pqr = new Lingkaran (1.9);

            System.out.println ("Data Default   : ");
            System.out.println ("---------------------");
            System.out.println ("Jari-jari ABC  : "+
                                 abc.getjarijari());
            System.out.println ("Jari-jari PQR  : "+
                                 pqr.getjarijari());
            System.out.println ();
      }
}
```

Ouput Program :



Penggunaan package memberikan beberapa keuntungan :

1. Program utama terlihat ramping sesuai kebutuhan.
2. Pembacaan alur program menjadi lebih terfocus.
3. Penelusuran lokasi kesalahan program lebih mudah dilakukan.
4. Waktu yang diperlukan oleh computer untuk menyimpan kode program lebih cepat
5. Efisiensi waktu untuk *maintenance* program menjadi lebih terjamin.

Meng-compile package berbeda dengan meng-compile file program biasa

BAB 10 APPLET

I. Tujuan Praktikum:

1. Mengetahui dan memahami dasar-dasar applet.
2. Mampu membuat dan menjalankan applet

II. Dasar Teori

Salah satu kemampuan yang dimiliki oleh Java adalah menjalankan applet di dalam Web Browser. Aplikasi Java dapat dijalankan dari baris perintah dengan memerintahkan interpreter Java untuk meninterpretasikan kode byte yang terdapat di dalam file class. Sebaliknya, applet biasanya berjalan pada halaman Web via browser Java-enabled. JDK diluncurkan dengan program applet viewer yang berdiri sndiri yang memungkinkan menguji applet dengan lebih mudah.

Untuk memnaggil (load) applet ke dalam Web browser, harus dibuat file terpisah yang berisi tag HTML untuk memberitahu browser applet yang akan dipanggil dan tempat untuk setiap applet pada halaman Web. Sebelum pengembangan Java, fungsi HTML hanyalah sebagai sarana untuk menunjukkan elemen-elemen halaman hypertext. Misalnya, <TITLE> menunjukkan judul dari halaman / page, dan text yang mengikuti tag ini menjadi judul dari page. Tag </TITLE> menyatakan akhir dari judul .

Ekstensi HTML dari Java memberitahu browser Java-enabled hal-hal berikut:

1. Nama file class,
2. Lokasi file class,
3. Bagaimana penempatan applet pada halaman Web.

Browser kemudian me-retrieve file class dari Net (atau dari direktori pada komputer yang digunakan pemakai) dan secara otomatis menjalankan applet.

Di samping applet, halaman Web dapat berisi semua elemen HTML lain yang dapat dilihat di halaman Web: berbagai jenis huruf, senarai bersimbol, grafik, link, dan lain sebagainya. Applet hanya salah satu bagian dari halaman hypertext. Yang selalu perlu diingat adalah Java bukan untuk merancang halaman HTML melainkan sarana untuk menghidupkannya.

Applet Sederhana

Dalam pandangan seorang programmer, applet hanyalah class Java yang memperluas class Applet. Applet merupakan bagian paket java.applet. Class Applet berelasi dengan class-class AWT.

Berikut adalah contoh applet Java:

```
import java.awt.*;
import java.applet.*;
public class NotHelloAgain extends Applet {
      Font f = new Font ("System", Font.BOLD, 18);
      Public void paint (Graphocs g) {
            g.setFont(f);
            g.drawString ("we won't use 'hello world'",25,50);
            }
      }
```

Perhatikan bahwa method paint didefinisikan kembali untuk menampilkan sesuatu. Untuk mendapatkan file .class potongan program Java di atas harus di-compile terlebih dahulu.

Di samping mengkompilasi file Java ini, Java memerlukan minimal sebuah file HTML yang sederhana untuk dikompilasi ke dalam file class. Meskipun tidak harus, biasanya nama yang diberikan adalah sama dengan nama applet yang besar yang terdapat di dalamnya.

```
<APPLET CODE = "NotHelloAgain.class" width=100 height=100>
</APPLET>
```

Menampilkan Applet

Ada tiga cara untuk melihat suatu Applet bekerja:

1. Langsung gunakan AppletViewer Sun.
   Masukkan perintah:
   ```
   Appletviewer NotHelloAgain.html
   ```
2. Dalam Netscape, dipanggil file lokal dengan memilih menu File, dan pilih submenu Open File. Kemudian masukkan nama file atau dengan memilih nama file dari dialog box. Perintah ini memberitahu Netscape untuk memanggil file HTML.
3. Menyertakan URL untuk halaman HTML yang berisi Applet dan memberitahu Netscape untuk membuka URL. Lakukan ini dengan memilih perintah File | Open Location (atau klik tombol Open) dan ketik dalam file URL.

Di samping Netscape dapat digunakan browser Java-enabled lain seperti Internet Explorer milik Microsoft.