BAB I

PENGANTAR

1.1 Bahasa Pemrograman Java

Bahasa pemrograman java yang telah dirilis adalah :

1. Produk Sun Microsystem : Java 2 Platform Standart Edition (J2SE), Java 2 Micro Edition (J2ME), Java 2 Enterprise Edition (J2EE).
2. Produk Borland : Borland JBuilder
3. Produk Microsoft : Microsoft Visual J/J++

Spesifikasi J2EE antara lain Java Database Connectivity, JDOM (untuk XML), Remote Method Invocation, Enterprise Java Beans, Java Messaging, Java Server Pages, Java Servlet, Socket Programming, dan SQLJ. Java/Standard Development Kit (JDK/SDK) merupakan alat-alat utama bagi programmer untuk membuat dan menjalankan java. Development Kit dapat didownload di :
http://java.sun.com/product/JDK/index.html atau
http://java.sun.com/j2se/1.5/index.html atau,
http://java.sun.com/cgi-bin/java-ports.cgi

Komponen JDK antara lain compiler(javac), interpreter(java) disebut juga java virtual machine atau java runtime environment, applet viewer(appletviewer), debugger(jdb), java class library(jcl), header dan stub generator(javah), dan yang penting yaitu java documentation(javadoc).

Penjelasan penggunaan komponen JDK :

1. Kompilator (javac)
   Berfungsi untuk kompilasi file source code : *.java menjadi *.class
   Syntax umum : javac nama_file.java
2. Interpreter (java)
   Bertugas untuk menjalankan bytecode (*.class)
   Syntax umum : java nama_file.class

3. Applet Viewer

   Digunakan untuk menjalanakan applet viewer, namun sekarang sudah digantikan browser.

   Syntax umum : appletviewer nama_file.html
4. Java Debugger

   Bertugas untuk melakukan debugging aplikasi java. Syntax umum : jdb option
5. Java Class File Diassembler (javap)

   Bertugas membuat daftar method dan attribute public dari suatu kelas.

   Syntax : javap namaKelas
6. Java Header and Stub Generator

   Bertugas menerjemahkan bahasa yang ditulis dalam bahasa Java menjadi bahasa pemrograman C.

   Syntax umum : javah namaKelas
7. Java Documentation Generator

   Menampilkan pustaka kelas, interface, constructor, dan method standard yang telah dibuat vendor.

   Dari hasil instalasi JDK, dokumentasi ini dapat dilihat pada C:\java\docs\api\index.html dan Dari hasil instalasi Netbeans, dapat dilihat pada C:\Program Files\NetBeans3.6\doc\junit\index.html
8. Source Code Java API

   Source code ini dapat diperoleh dari file src.zip.

Untuk pemrogram pemula, lingkungan pemrograman java dapat diringkas menjadi :

1. Editing source code menggunakan editor teks, seperti Notepad atau TextPad
2. Compiling menggunakan keyword javac melalui command prompt (dapat juga dari editor teks seperti TextPad).
3. Executing menggunakan :
   a. Command prompt untuk java application(atau dari editor teks TextPad)
   b. Browser atau appletviewer untuk java applet.

1.2. Pemrograman Java

Baris-baris program dalam java harus ditulis dalam lingkup *class*. Bagaimanakah membuat sebuah program paling sederhana dengan java? Berikut ini contoh program pendek untuk menampilkan tulisan "Belajar Java" di layar monitor anda.

```
class Latihan1 {
      public static void main(String args[]) {
            System.out.println("Belajar Java");
      }
}
```

Program pendek diatas disimpan dengan nama file Latihan1.java. Penamaan ini mengikuti aturan bahwa nama file harus sama dengan nama ***class***. Tampilan di layar monitor jika program tersebut berhasil dijalankan adalah :

```
Belajar Java
```

Dalam Java atau bahasa pemrograman yang lain dikenal istilah **Token**. Token adalah elemen terkecil di program yang masih memiliki arti. Ada 5 token dalam bahasa Java yaitu *identifier, keyword, literal* dan *tipe data*, *operator*, serta *separator.*

Identifier

Identifier adalah token yang merepresentasikan nama sesuatu. Sesuatu tersebut adalah variabel, atau konstanta, atau method, atau kelas, atau package, atau interface.

Keyword (kata kunci)

Kata kunci digunakan untuk suatu tujuan tertentu. Ada 51 keyword dalam java yaitu :

Tabel 1. Daftar Keyword dalam Java

abstract	continue	for	new	switch
boolean	default	goto	null	synchronized
break	do	if	package	this
byte	double	implements	private	threadsafe

byvalue	else	import	protected	throw
case	extends	instanceof	public	throws
catch	false	int	return	transient
car	final	interface	short	true
class	finally	long	static	try
const	float	native	super	void
				while

Literal dan Tipe Data Primitif

Literal adalah nilai variabel/attribute atau nilai konstanta atau nilai objek data. Ada tiga besaran literal dalam java yaitu angka, karakter, dan string. Angka terdiri dari byte, short, int, long, float, double, dan boolean (dianggap angka true = 1 atau false = 0). Semua variable dan konstanta yang akan digunakan harus dipesan terlebih dahulu dalam deklarasi.

Bentuk umum :

TipeData namaVar = ungkapan_atau_nilai;
TipeData namaVar1, namaVar2, …;
[modifier] static final TipeData NAMAKONSTANTA = nilai;

Contoh deklarasi :

double a=3, b=4;
*double c = Math.sqrt(a*a+b*b);*
static final PHI=3.14;
static final double CM_PER_INC = 2.54;

Berikut tabel jangkauan dan ukuran dari semua tipe data sederhana dalam java :

Tabel 2

Jangkauan dan Ukuran Tipe Data Sederhana dalam Java

Tipe Data Primitif	Jangkauan	Ukuran (bit)
byte	-128 s/d 127	8
short	-32767 s/d 32767	16
int	-2147483648 s/d 2147483647	32
long	-9223372036854775808 s/d 9223372036854775807	64
char	sebuah unicode	16

float	3.4e-038 s/d 3.4e+038	32
double	1.7e-308 s/d 1.7e+308	54
boolean	false = 0 atau true = 1	8

Operator

Operator melakukan komputasi terhadap satu/dua objek data. Operan yang dioperasikan dapat berupa literal, variabel, atau nilai yang dikirim method.

Berikut tabel dan hirarki operator :

Tabel 3. Tabel Hirarki Operator

Prioritas	Kelompok Operator	Keterngan
1	. [] ()	sekaligus
2	++var, --var, ~, instanceof	preincrement, predecrement, unary, instance dari kelas …
3	(type) (casting)	
4	!	not
5	*, /, %	perkalian, pembagian, modulus
6	+, -	penjumlahan, pengurangn
7	<<, >>, >>>	geser untuk bil biner
8	<, >, <=, >=	pembandingan
9	==, !=	kesamaan, ketidaksamaan
10	&	and
11	^	exclusive or
12	|	unconditional or
13	&&	conditional and
14	||	conditional or
15	? :	shorthand untuk if..then…else…
16	=, +=, -=, *=, /=, %=, ^=	operator penugasan
17	&=, |=, <<=, >>=, >>>=	operator penugasan
18	var++, var--	postincrement, postdecrement

Separator

Separator menginformasikan ke compiler java mengenai adanya kelompok kode program. Berikut adalah daftar separator di java :

Tabel 1.3. Daftar Separator di Java

Notasi	Nama	Deskripsi
(…)	kurung	mengelompokkan parameter method.
{…}	kurung kurawal	mengelompokkan nilai-nilai suatu array, mendefinisikan blok kode kelas ataupun kode

		method.
[…]	kurung siku	mendeklarasikan tipe array
:	titik koma	mengakhiri pernyataan
,	koma	memisahkan identifier-identifier di bagian deklarsi variable, merangkai pernyataan-pernyataan di dalam for.
.	titik	memisahkan nama-nama package, memisahkan kelas dari objek, dan objek dari method.

1.3. Keyword break, continue, dan return

Kelompok keyword ini berfungsi untuk melompat dari suatu baris program java ke baris yang lain (variasi lain dari perintah goto dalam bahasa BASIC), sehingga berakibat pengabaian baris program setelah keyword tersebut.

1.3.1. Penggunaan keyword break

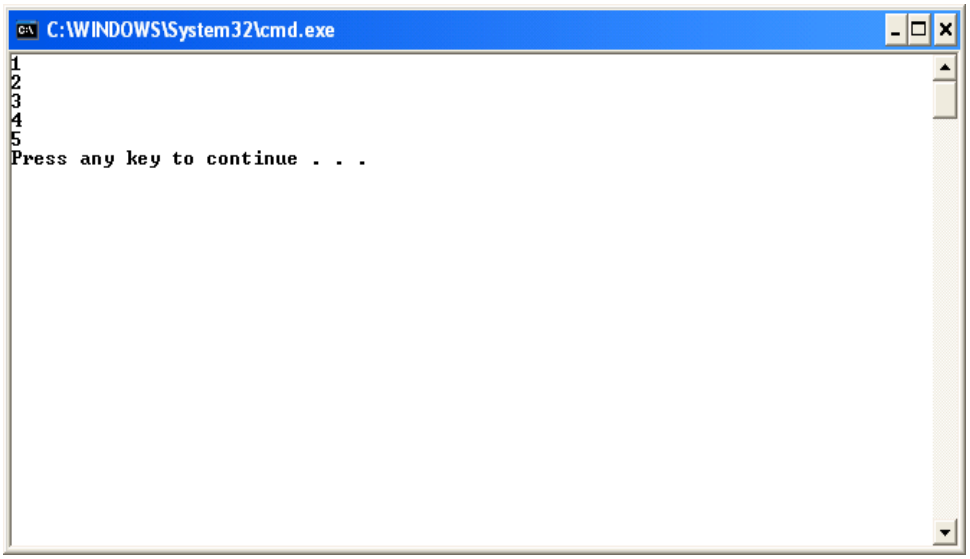
Ada dua penggunaan keyword break, yaitu untuk keluar dari kendali percabangan switch, dan untuk keluar dari kendali perulangan.

Dengan keyword ini berarti percabangan/perulangan akan diakhiri, kemudian eksekusi dilanjutkan ke pernyataan setelah blok percabangan/perulangan tersebut.

Contoh :

```
public class contohBreak {
   public static void main(String args[]) {
      int i = 0;
      do {
        i++;
        System.out.println(i);
        if (i==5) break;              // Jika i bernilai 5 maka perulangan
      } while (i <= 9)                // Do While dihentikan
   }
}
```

Simpanlah program diatas dengan nama contohBreak.java, kemudian compile dan jalankan. Hasil running program tersebut :



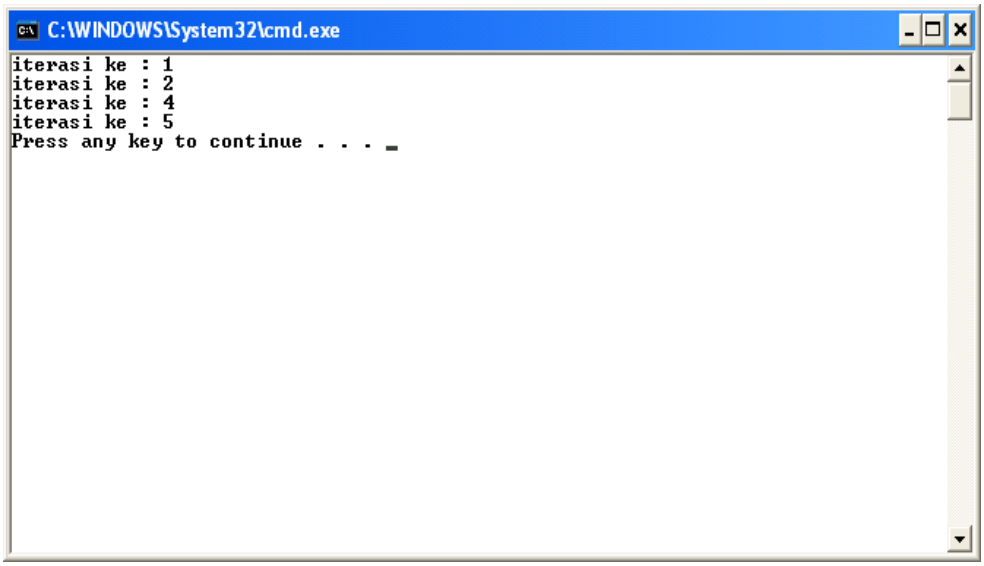
1.3.2. Penggunaan keyword continue

Penggunaan keyword ini untuk segera lompat ke perulangan berikutnya. Baris-baris program setelah keyword continue dalam blok perulangan saat itu berarti diabaikan.

Contoh :

```
public class contohContinue {
   public static void main(String args[]) {
      int i=0;
      do {
        i++;
        if (i==3) continue;
        System.out.println("iterasi ke : "+i);
        if (i==5) break;
      } while(i <= 9);
   }
}
```

Simpanlah program diatas dengan nama contohContinue.java, kemudian compile dan jalankan. Hasil running program tersebut :



1.3.3. Penggunaan keyword return

Keyword ini digunakan untuk keluar dari suatu method. Baris-baris program setelah keyword ini yang berada dalam blok method tersebut akan diabaikan. Kemudian eksekusi dilanjutkan ke pernyataan setelah blok method tersebut.

Contoh cuplikan program :

```
int abs(int x) {
   if (x >= 0)
      return x;
   else
      return(-x)
   . . .
   }
}
```

1.4. Soal Latihan

1. Sebutkan komponen-komponen yang terdapat dalam Java Development Kit!
2. Apa yang dimaksud dengan token dan identifier?
3. Sebutkan kegunaan dari operator , separator, keyword break dan keyword continue.
4. Buatlah program yang menampilkan tulisan :

```
Belajar java memang mudah
Jika dilakukan dengan tekun
```

5. Buatlah contoh program lain yang menggunakan keyword ***break*** dan ***continue.***

BAB II
KELAS, METHOD, DAN MODIFIER

Apapun kode program yang diimplementasikan di java harus *dikapsulkan* ke dalam bentuk kelas (class). Sebuah kelas akan mendefinisikan satu objek atau sekumpulan objek sama sifat dan perilakunya.

Ada dua kelompok kelas, yaitu **kelas standard** dan **kelas yang didefinisikan sendiri**. Kumpulan dari method/kelas standard dalam java dikenal dengan API (Application Programming Interface). Kelas standard disediakan oleh Java dan siap dipakai.

2.1. Pengertian Kelas

Kelas merupakan sarana pengkapsulan kumpulan data dan kumpulan method java. Kumpulan data dan method tersebut berwujud baris-baris program java. Kumpulan method berfungsi untuk mengoperasikan kumpulan data dalam kelas tersebut. Kelas digunakan untuk membuat objek, dan berperan sebagai tipe data dari objek.

2.2. Anatomi Kelas

Bentuk umum struktur anatomi kelas sebagai berikut :

```
(modifier1) class NamaKelas (modifier2) {
    classbody
}
```

Classbody terdiri dari satu atau beberapa *attribute*, *constructor*, dan *method.* Modifier 1 dan 2 pada anatomi kelas, sifatnya optional, digunakan berdasarkan kebutuhan. Modifier menunjukkan sifat-sifat tertentu dari kelasnya, sifat-sifat dari methodnya, atau sifat-sifat attributenya. Ada 10 *keyword* yang digunakan sebagai modifier 1 dan dikelompokkan menjadi :

1. Modifier akses (*public*, *protected*, *default*, *private*)
2. Modifier *final*
3. Modifier *static*
4. Modifier *abstract*

5. Modifier *synchronized*
6. Modifier *native*
7. Modifier *storage* (*transient*, *volatile*)

Modifier yang memiliki sifat saling kontradiktif yaitu *static* dan *abstract*. Sementara *static* tidak boleh memberi sifat pada *interface*, dan keyword *super* tidak boleh digunakan pada method *static*.

Ada 2 keyword yang sering digunakan sebagai modifier 2, yaitu *extends* dan *implements*.

2.3. Deklarasi Attribute (Variabel Anggota Data)

Deklarasi diletakkan di dalam classbody (di luar method). Bentuk umum deklarasi attribute :

```
[modifier] tipedata namavariabel;
[public] [static] final tipedata NAMA_KONSTANTA = nilai;
```

Contoh :

```
public class CircleClass {
   public static final double PI = 3.14159265358979323846;
   public double x, y, r;
    // dan seterusnya
}
```

2.4. Method

Method merupakan tingkah laku dari suatu objek, jika bersifat static berarti tingkah laku semua objek dalam kelas tersebut. Method diletakkan di dalam classbody (sebaiknya tidak diletakkan dalam method lain).

Bentuk umum method :

```
[modifier] tipe_return_value namaMethod(tipe parameter) {
   methodbody;
}
```

Modifier boleh lebih dari satu (dipisah oleh spasi). Pasangan tipe dan parameter dapat lebih dari satu (dipisah oleh koma).

Baris-baris program java saat dijalankan dimulai dari method main(). Bentuk umum method `main()` adalah sebagai berikut :

```
[modifier] tipe_return_value main(String args[]) {
   methodbody
}
```

Ada dua sintaks pemanggilan suatu method :

namaObjek.namaMethod([nilaiParamater]);

namaKelas.namaMethod([nilaiParamater]);

Tidak semua member (class, attribute, dan method) dapat diakses method, Berikut tabel aksesnya :

method	member (class, attribute, method)
static	static boleh lewat objek ataupun class, boleh langsung kalau dalam kelas sendiri
static	non static boleh lewat objek, langsung tidak boleh, lewat class tidak boleh
non static	static boleh lewat objek ataupun class, boleh langsung kalau dalam kelas sendiri
non static	non static boleh hanya lewat objek, langsung tidak boleh, lewat class tidak boleh

Method dasar ada dua jenis yaitu **`getter()`** dan **`setter().`** Method jenis getter() merupakan method-method yang berfungsi untuk mendapatkan informasi dari class, sedangkan jenis setter() berfungsi untuk menentukan isi atribut (variabel) dalam class.

Contoh : Berikut ini contoh method getter() dan setter() dalam kelas Dog yang kita definisikan sendiri.

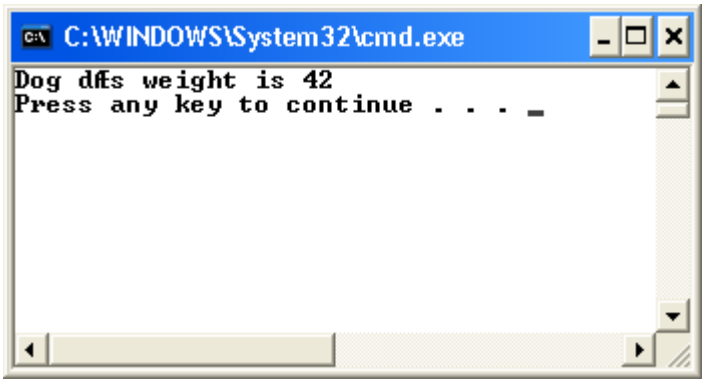
Sebuah kelas bernama Dog :

```
public class Dog {
   private int weight;              // deklarasi atribut

   public int getWeight() {         // method getter
      return weight;
   }

   Public void setWeight(int newWeight) { //method setter
      weight = newWeight;
   }
}
```

Dan sebuah kelas bernama TesDog.java yang akan memanfaatkan kelas Dog.java diatas.

```
public class TesDog {
public static void main(String[] args) {
Dog d = new Dog();
d.setWeight(42);          // method setter digunakan
System.out.println("Dog d's weight is "+d.getWeight())
                                                  //getter digunakan
   }
}
```

Setelah kedua kelas tersebut dicompile, maka hasil running dari program Tesdog tersebut :



2.4.1. Overloading pada Method

Overloading adalah mendefinisikan dua atau lebih method di dalam satu kelas yang sama, dengan nama yang sama, namun dengan **deklarasi parameter yang berbeda**. Overloading disini bukan overloading terhadap operator sebagaimana dalam bahasa C++.

Java interpreter mampu membedakan method mana yang dieksekusi dengan mengenali tipe parameter yang dilewatkan ke method, serta mengenali return valuenya.

Tugas dari method-method yang dioverloading tersebut berbeda. Contoh overloading terhadap **method pangkat** di bawah ini yang mmiliki parameter berbeda-beda :

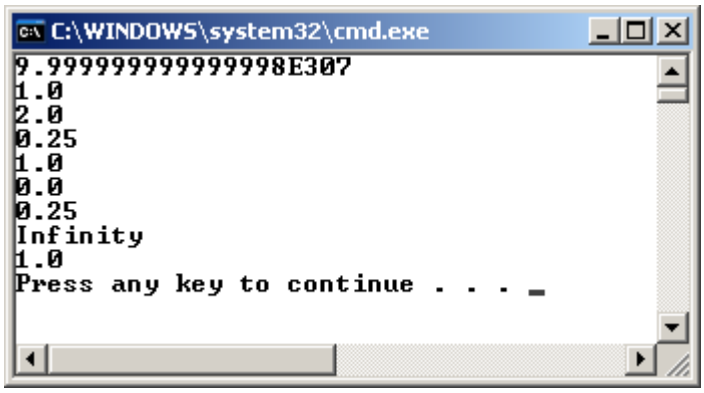
```
import java.lang.*;
public class Perkalian {

  private double pangkat(int a, int b) {
    double hasil = 1.0;
    if (b==0) {
      hasil = 1;
    }
    else {
      if ((a==0) && (b<0)) {
        System.out.println("Infinity");
      }
      else {
        if ((a==0) && (b>0)) {
          hasil=0;
        }
        else {
          if ((a != 0) && (b>0)) {
            for (int i=1; i <=b; i++) {
              hasil=hasil*a;
            }
          }
          else {
            if ((a != 0) && (b<0)) {
              for (int i=1; i <=b; i++) {
                hasil=hasil*a;
              }
```

```
                hasil = 1/hasil;
              }
            }
          }
        }
      }
      return hasil;
    }


    private double pangkat(double a, int b) {
      double hasil = 1.0;
      if (b==0) {
        hasil = 1;
      }
      else {
              if ((a==0) && (b<0)) {
                System.out.println("Infinity");
              }
              else {
                  if ((a==0.0) && (b>0)) {
                    hasil=0;
                  }
                  else {
                      if ((a != 0.0) && (b>0)) {
                        for (int i=1; i<=b; i++) {
                            hasil=hasil*a;
                        }
                      }
                      else {
                          if ((a != 0) && (b<0)) {
                            for (int i=1; i<=b; i++) {
                                hasil=hasil*a;
                            }
                            hasil=1/hasil;
                          }
                      }
                  }
              }
      }
      return hasil;
    }

    private double pangkat(int a, double b) {
            double hasil = 1.0;
```

```
  if (b==0) {
    hasil = 1;
  }
  else {
            if ((a==0) && (b<0)) {
              System.out.println("Infinity");
            }
            else {
                if ((a==0) && (b>0.0)) {
                  hasil=0;
                }
                else {
                    if ((a!=0) && (b>0)) {
                      hasil=Math.exp(b*Math.log(a));
                    }
                    else {
                        if ((a!=0) && (b<0.0)) {
                          hasil=1/Math.exp(b*Math.log1p(a));
                        }
                    }
                }
             }
          }
   return hasil;
}

private double pangkat(double a, double b) {
          double hasil = 1.0;
  if (b==0) {
    hasil = 1;
  }
  else {
                 if ((a==0) && (b<0)) {
                   System.out.println("Infinity");
                 }
                 else {
                    if ((a==0) && (b>0.0)) {
                      hasil=0;
                    }
                    else {
                        if ((a!=0) && (b>0)) {
                          hasil=Math.exp(b*Math.log(a));
                        }
                        else {
                            if ((a!=0) && (b<0.0)) {
```

```
                    hasil=1/Math.exp(b*Math.log1p(a));
                }
            }
        }
    }
}
    return hasil;
}

public static void main(String[] args) {
  Perkalian kali = new Perkalian();
  System.out.println(kali.pangkat(10,308));
  System.out.println(kali.pangkat(6,0));
  System.out.println(kali.pangkat(4,0.5));
  System.out.println(kali.pangkat(0.5,2));
  System.out.println(kali.pangkat(0.7,0));
  System.out.println(kali.pangkat(0,8));
  System.out.println(kali.pangkat(0.0625,0.5));
  System.out.println(kali.pangkat(0,-3));
}
}
```

Output program Pangkat tersebut :



Penjelasan program

Perhatikan pada sintaks berikut dari program :

```
System.out.println(kali.pangkat(10,308));
```

Pemanggilan method pangkat diatas diikuti oleh parameter berupa angka 10 dan 308. Karena angka 10 dan 308 bertipe integer maka method yang dipanggil adalah method : `pangkat(int a, int b).`

Jika pemanggilan method pangkat diubah menjadi :

System.out.println(kali.pangkat(10.2,308.5));

Maka method yang dipanggil adalah method : `pangkat(double a, double b)`

2.4.2. Keyword this

This adalah objek yang langsung digunakan tanpa didahului proses instansiasi. Penggunaan keyword ini yaitu bila ada attribute (non static) dari suatu kelas akan digunakan method yang berada dalam kelas yang sama, namun nama attribute tersebut dan nama parameter yang dilewatkan pada method tersebut SAMA. Keyword ini dapat digunakan secara implisit maupun eksplisit.

Contoh penggunaan yang eksplisit :

```
class RectangleToy {
   private double width, height;
   public void setRectangle(double width, double height) {
      this.width = width;
      this.height = height;
   }
}
```

Pemanggilan attribute yang eksplisit, biasa digunakan untuk mengatasi penamaan yang sama. Pada contoh ini parameter di method setRectangle() menggunakan nama yang sama dengan attribute di kelas RectangleToy.

Contoh pemanggilan implisit :

```
class RectangleToy {
   private double width, height;
    public void setRectangle(double newwidth, double newheight)
     {
       width = newwidth;
       height = newheight;
   }
}
```

2.5. Constructor

Pada prinsipnya constructor adalah method yang tidak memiliki return value (secara implisit adalah instant dari kelasnya). Nama constructor sama dengan nama kelas, dan dapat diberi modifier akses (public, protected, default, private).

Bentuk umum pendefinisian constructor :

```
[modifier] namaConstructor(tipe namaparameter) {
   constructorBody;
}
```

Tujuan constructor dibuat adalah untuk melakukan inisialisasi yang diperlukan ketika suatu objek baru dibentuk. Constructor akan langsung dijalankan ketika objek dipanggil/dibentuk.

Contoh constructor dan overloadingnya :

```
class PersonToy {
   String name;
   String addressLine1;
   String addressLine2;
   String city;
   int age;
   public PersonToy() {                    // constructor
      name = " ";
      addressLine1 = " ";
      addressLine2 = " ";
      city = " ";
      age = 0;
   }

   public PersonToy(String newName,
                    String newAddress1,
                    String newAddress2;
                    String newCity;
                    int newAge) {
      name = newName;
      addressLine1 = newAddressLine1;
      addressLine2 = newAddressLine2;
      city = newCity;
      age = newAge;
   }
}
```

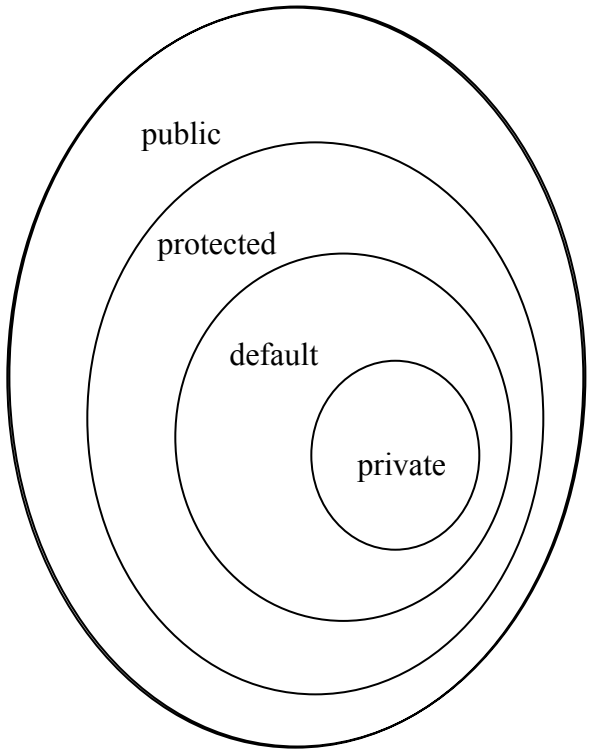
2.6. Modifier

Modifier untuk menentukan hubungan suatu unsur kelas dengan unsur kelas lainnya, contohnya hubungan kepemilikan antara kelas dan objek.

Wilayah modifier akses, dalam bentuk tabel :

Wilayah Akses	***public***	***protected***	***default***	***private***
Di kelas yg sama	√	√	√	√
Beda kelas, di package yg sama	√	√	√	x
Beda kelas, beda package, di kelas turunan	√	√	x	x
Beda kelas, beda package, tidak di kelas turunan	√	x	x	x

Wilayah modifier akses tersebut dapat diilustrasikan:



Gambar 2.1

Wilayah Modifier Akses

2.6.1. Public

Menyatakan bahwa kelas/method/attribute tersebut dapat diakses oleh kelas lain dimanapun.

2.6.2. Protected

Menyatakan bahwa kelas/method/attribute tersebut dapat diakses oleh kelas lain yang berada dalam satu package atau kelas lain tersebut merupakan turunannya.

2.6.3. Private

Menyatakan bahwa kelas tersebut tidak dapat diakses sama sekali oleh kelas lain bahkan juga tidak dapat diturunkan.

Berarti attribute-attribute yang private hanya dapat diakses oleh method-method dalam kelas yang sama, kelas lain masih dapat mengakses melalui method-method tersebut asal modifiernya public.

Pertimbangan suatu attribute dideklarasikan private :

1. Bila kelas lain tak memerlukan attribute tersebut.
2. Melindungi suatu attribute dari kemungkinan nilainya diubah oleh method lain dari kelas lain.

Final (no extended, no overrided)

- Menyatakan bahwa suatu kelas tidak dapat menurunkan (extend) kelas lain.
- Menyatakan bahwa suatu method tidak dapat dioverride oleh method lain.
- Membentuk suatu attribute menjadi konstanta.

Static (no need instanciation, no overrided)

Method dan attribute ada dua jenis, yaitu method dan attribute milik kelas serta method dan attribute milik suatu objek.

Method dan attribute milik kelas, diakses melalui tiga cara, pertama melalui nama kelasnya, kedua melalui nama objek yang diinstant dari kelasnya, dan ketiga bebas tanpa didahului nama kelas atau nama objeknya.

Method dan attribute milik objek, diakses hanya melalui nama objeknya.

Method static tidak bisa dioverride. Method main() harus memiliki modifier static.

Modifier static artinya method dan attribute milik kelas, menjadi sifat bersama dari semua objek dalam kelas tersebut (tidak memerlukan instansiasi objek untuk menjalankannya).

Contoh `System.out.println()` bersifat static artinya untuk memanggil method `println()` tidak harus dilakukan instansiasi dari kelas System.

2.6.6. Abstract (no instanciation, should be overrided)

Abstract class adalah kelas murni (tanpa objek) dan tidak boleh memiliki objek (tidak boleh ada instansiasi) serta method-method yang abstract harus disempurna-kan oleh kelas turunannya melalui override.

Kelas seperti ini biasanya merupakan root suatu struktur kelas.

Konsekuensi penggunaan sifat abstract :

1. Tidak dapat dibuat constructor yang abstract.
2. Tidak dapat dibuat method yang static dan abstract (kedua sifat sali ng kontradiktif).
3. Tidak diijinkan membuat method yang private dan abstract (kedua sifat ini juga saling kontradiktif).

2.6.7. Synchronized (khusus modifier method)

Pada lingkungan multithread, dimungkinkan lebih dari satu jalui eksekusi yang berjalan di kode yang sama, kondisi tersebut dapat diatur sehingga pada selang waktu tertentu hanya ada satu jalur eksekusi yang diijinkan di method yang synchronized (eksekusi dilakukan secara mutual exclusive).

2.6.8. Native (khusus modifier method)

Modifier ini digunakan untuk memanggil/mengakses method yang ditulis dalam bahasa C/C++.

Seperti method yang abstract, method yang native hanya berupa prototype, implementasi method ini berada di file external (dalam folder yang sama).

2.6.9. Transient (khusus modifier attribute)

Java memiliki konsep serialisasi, yaitu kemampuan untuk mentransformasikan objek menjadi suatu stream, sehingga objek dapat ditransfer dari suatu aplikasi ke aplikasi lainnya, atau dari suatu workstation ke workstation lainnya. Prinsip ini digunakan dalam aplikasi client-server.

Salah satu ketentuan serialisasi adalah tidak boleh ada perubahan nilai attribute suatu objek, saat objek tersebut ditransformasikan menjadi stream, dan sebaliknya, namun suatu objek dapat memiliki nilai attribute yang boleh berubah (bersifat transient).

2.6.10. Volatile (khusus modifier attribute)

Dalam manajemen thread, java dapat menyimpan nilai suatu attribute (yang sering diakses thread) menjadi cache value, sehingga tidak perlu selalu merujuk ke lokasi memori aslinya.

Attribute tersebut bersifat volatile, karena nilainya rentan berubah bila diakses oleh lebih dari satu thread.

2.6.11. Extends

Bila terjadi pewarisan, kelas yang mewariskan method dan attributenya disebut kelas super, sedangkan yang diwariskan disebut subkelas.

Kelas yang memiliki modifier ini berarti merupakan subkelas dari suatu kelas super.

Caranya :

```
[modifier1] class NamaSubKelas extends NamaKelasSuper {
   classBody
}
```

2.6.12. Implements

Kelas yang memilki modifier2 implements artinya kelas tersebut mengimplementasikan satu atau lebih interface. Bila terdapat lebih dari satu interface, gunakan tanda koma di antara interface-interface tersebut.

Caranya :

[modifer] class NamaKelas implements NamaInterface1, NamaInterface2
{
classBody
}

2.7. Soal Latihan

1. Apakah pengertian dari :
   a. kelas
   b. method
   c. modifier
2. Buatlah sebuah kelas yang terdiri dari attribute dan method. Kemudian buat pula kelas yang mengandung method `main()` untuk mengakses kelas tersebut.
3. Apa yang dimaksud dengan overloading method?
4. Buatlah contoh program yang mengandung overloading method !
5. Jelaskan yang dimaksud dengan modifier public, protected, private, static dan final !

BAB III.
OBJEK

Pada prinsipnya objek adalah sebuah pointer. Objek dibentuk/dicetak oleh sebuah **class** java. Proses pembentukan objek dari suatu class ini disebut dengan **instansiasi (instance).** Setelah dibentuk oleh suatu class maka objek akan terus berada di memory dan siap dipakai sampai kemudian dihancurkan.

Siklus Hidup Objek

Suatu objek di java, terlebih dahulu harus dideklarasikan, setelah itu mengalami pembuatan, penggunaan, kemudian penghancuran.

Deklarasi dan Pembuatan/Instansiasi Objek

Ilustrasi deklarasi objek :

```
MyClass mc;
```

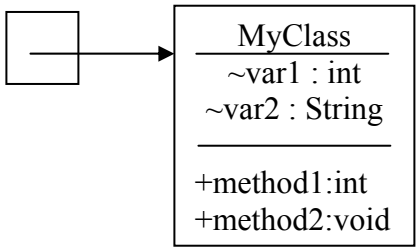
MyClass

Ditentukan variabel mc dengan tipe MyClass, dalam deklarasi mc masih berupa pointer dan belum menunjuk alokasi memori. Keyword **new** diperlukan untuk membentuk objek tsb sekaligus memanggil constructor.

Ilustrasi pembentukan objek :

```
mc = new MyClass()
```

MyClass



Ada 4 cara untuk melakukan deklarasi dan pembuatan objek (ingat NamaKelas sama dengan NamaConstructor), yaitu :

Cara1 :

NamaKelas namaObjek; *// deklarasi*
namaObjek = new NamaConstructor(); *// pembuatan*

Deklarasi dan pembuatan objek boleh jadi satu dalam suatu kelas. Deklarasi dan pembuatan objek boleh dipisah di kelas yang berbeda, misalnya deklarasi objek di method `main()` pada kelas1, sedangkan pembuatan/instansiasi objek berada pada kelas2.

Cara2 :

NamaKelas namaObjek = new NamaConstructor();

Artinya dua proses deklarasi dan pembuatan dijadikan satu. Cara inilah yang nanti akan banyak kita pakai dalam mempelajari materi buku ini.

Cara3 :

new NamaConstructor();

Cara ini objek diinstansiasi tanpa nama, tujuannya hanya untuk menjalankan constructor.

Cara4 :

New NamaKelas(”…..”).namaMethod();

Cara ini objek diinstansiasi dengan nama kelas sekaligus menjalankan method tertentu dalam objek tersebut.

Penggunaan Objek

Penggunaan objek sebenarnya untuk memanggil method yang merupakan sifat objek tersebut. Bentuk umum :

namaObjek.namaMethod([daftarNilaiParameter]);

Penghancuran Objek

Teknik yang digunakan java untuk menangani objek yang sudah tidak diperlukan lagi disebut *garbage collection*. Objek yang sudah tidak diperlukan lagi akan terdeteksi oleh JVM, sehingga secara otomatis dihancurkan oleh garbage collector (bukan oleh programmer).

3.5. Contoh Kasus Obyek

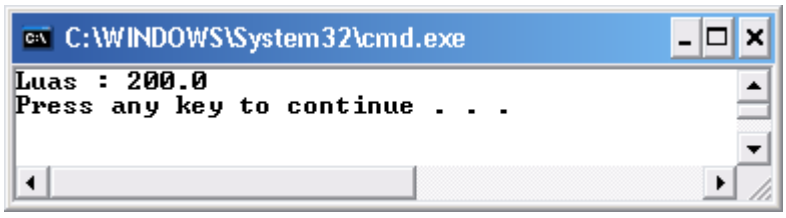
Untuk memberikan ilustrasi yang jelas tentang bagaimana membuat sebuah obyek, menggunakan obyek, dan menghancurkan obyek, berikut ini adalah contoh program pendek untuk menghitung luas persegi panjang dengan rumus **Luas=panjang* lebar**. Program di bawah ini menerapkan konsep berorientasi objek, dimana class pembentuk objek adalah **class PersegiPanjang**.

```
 1     class PersegiPanjang{
 2         private double panjang;
 3         private double lebar;
 4         private double tinggi;
 5
 6         public PersegiPanjang() {
 7             panjang = 0;
 8             lebar = 0;
 9         }
10
11         private double hitungluas(double p, double l) {
12             return p*l;
13         }
14
15         public void setPanjang(double panjang) {
16             this.panjang = panjang;
17         }
18
19         public void setLebar(double lebar) {
20             this.lebar = lebar;
21         }
22
23         public double getLuas() {
24             return hitungluas(panjang, lebar);
25         }
26
27         public static void main(String[] args) {
28            PersegiPanjang pp = new PersegiPanjang();
29            pp.setPanjang(10);
30            pp.setLebar(20);
31            System.out.println("Luas : "+ pp.getLuas());
32         }
33     }
```

Penjelasan program :

Proses pembentukan objek (instansiasi) ada pada baris 28, dimana pp adalah nama objek yang hendak dibentuk, sedangkan PersegiPanjang adalah nama class pembentuk objek. Setelah proses berhasil maka pp akan memiliki method dan atribut sesuai dengan yang ada dalam class PesegiPanjang.

Jika program dijalankan maka hasilnya adalah sebagai berikut :



C. Soal Latihan

a. Apa yang dimaksud dengan objek? Berikan contohnya (minimal 5) !

b. Sebutkan 2 dari 4 cara untuk melakukan deklarasi dan pembuatan objek ! Berikan penjelasan dari masing-masing cara tersebut !

c. Tuliskan bentuk umum dari objek !

BAB IV
PACKAGE

4.1. Pengertian Package

Package adalah sarana/cara pengelompokkan dan pengorganisasian kelas-kelas dan interface yang sekelompok menjadi suatu unit tunggal dalam library. Secara fisik package berupa folder yang berisi file-file class yang berfungsi khusus. Package juga mempengaruhi mekanisme hak akses ke kelas-kelas di dalamnya.

4.2. Pengaruh Package terhadap Method `main()`

Kelas yang mengandung method **`main()`** memiliki syarat tidak berada dalam suatu package, dan hirarki posisi foldernya di atas package yang diimport.

4.3. Membuat Package

Ada tiga langkah untuk membuat package :

1. Mendeklarasikan dan memberi nama package.
2. Membuat struktur dan nama direktori yang sesuai dengan struktur dan nama package.
3. Mengkompilasi kelas-kelas sesuai dengan packagenya masing-masing.

4.4. Mendeklarasikan dan Memberi Nama Package

Dekalrasi package harus diletakkan pada bagian paling awal (sebelum deklarasi import) dari source code setiap kelas yang dibungkus package tersebut.

Bentuk umum deklarasi package :

package namaPackage;

Deklarasi tersebut akan memberitahukan kompilator, ke library manakah suatu kelas dikompilasi dan dirujuk.

Syarat nama package :

1. Diawali huruf kecil
2. Menggambarkan kelas-kelas yang dibungkusnya
3. Harus unik (berbeda dengan nama package standard)

4. Merepresentasikan path dari package tersebut.
5. Harus sama dengan nama direktorinya.

Contoh package standard :

`java.lang` (berisi kelas-kelas fundamental yang sering digunakan).

`java.awt` dan `javax.swing` (berisi kelas-kelas untuk membangun aplikasi GUI)

`java.io` (berisi kelas-kelas untuk proses input output)

4.5. Membuat Struktur Direktori

Pada langkah ini, buatlah direktori menggunakan file manager (di windows menggunakan explorer) sesuai struktur package dari langkah sebelumnya. Kemudian tempatkan kelas-kelas tersebut ke direktori yang bersesuaian (mirip seperti menyimpan file-file ke dalam folder).

Package dapat bersarang di package lain, sehingga dapat dibuat hirarki package. Bentuk umum pernyataan package multilevel :

package namaPackage1[.namaPackage2[.namaPackage3]];

Contoh hirarki package di JDK :

package java.awt.image;

4.6. Compile dan Run Kelas dari suatu Package

Selanjutnya masing-masing kelas tersebut dalam package tersebut dikompilasi menjadi byte code (*.class). Jika semua class telah dikompilasi maka package tersebut siap digunakan.

4.7. Menggunakan Package

Ada dua cara menggunakan suatu package yaitu :

1. Kelas yang menggunakan berada dalam direktori (package) yang sama dengan kelas-kelas yang digunakan. Maka tidak diperlukan import.
2. Kelas yang menggunakan berada dalam direktori (package) yang berbeda dengan kelas-kelas yang digunakan. Maka pada awal source code di kelas pengguna harus mencantumkan :

import namaPackage.NamaKelas; atau

import namaPackage.;*

Contoh :

import java.text.DecimalFormat;

import javax.swing.;*

4.8. Setting Classpath

Path hirarki package, didaftarkan sebagai salah satu nilai variabel lingkungan yang bernama Classpath.

Classpath diset dengan aturan : berawal dari drive (C:\ atau D:\) sampai dengan satu tingkat sebelum kita mendeklarasikan package.

4.9. Soal Latihan

1. Apa pengertian dari package?
2. Apa fungsi package?
3. Sebutkan langkah-langkah yang dibutuhkan untuk membuat package!
4. Bagaimana bentuk umum package? Dan bagaimana pula bentuk umum pernyataan package multilevel?
5. Buat sebuah aplikasi package dengan beberapa file! Kemudian buatlah ilustrasi penempatan file-file program dari aplikasi package yang dibuat tersebut!

BAB V
INFORMATION HIDING, ENCAPSULATION, INHERITANCE, DAN POLYMORHISM

5. 1. Latar Belakang Encapsulation

Vendor perangkat lunak computer merahasiakan source code produknya, user hanya diberitahu melalui manual cara menggunakan produknya. Programmer java pun tidak perlu mengetahui bagaimana rinci source code dari modul interface referensi (API), programmer hanya perlu tahu return value dan parameter milik method-methodnya, atau hanya perlu tahu parameter milik constructor-constructornya.

5.2. Information Hiding dan Encapsulation

Information Hiding adalah menyembunyikan attribute dan method suatu objek dari objek lain. Encapsulation adalah menyembunyikan atribute suatu objek dari objek lain. Attribute maupun method disembunyikan dengan cara memberikan modifier private.

Method dalam kelas yang sama, yang memberikan nilai pada attribute private disebut method **setter**, sedangkan method masih dalam kelas yang sama, yang mengambil nilai dari attribute private disebut **getter**. Di bawah ini adalah contoh program Information Hiding dan Encapsulation

PersegiPanjang.java

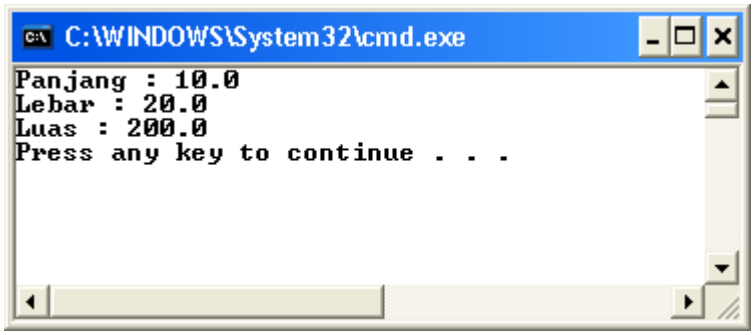
```
public class PersegiPanjang{
   private double panjang; // attribute yg di encap
   private double lebar;      // attribute yg di encap
   private double tinggi;     // attribute yg di encap
   public PersegiPanjang() {
      panjang = 0;
      lebar = 0;
   }
   private double luas(double p, double l) {   // di hide
      return p*l;
```

```
    }
    public void setPanjang(double panjang) {
        this.panjang = panjang;
    }
    public void setLebar(double lebar) {
        this.lebar = lebar;
    }
    public double getPanjang() {
        return panjang;
    }
    public double getLebar() {
        return lebar;
    }
    public double getLuas() {
        return luas(panjang, lebar);
        }
}
```

MainPersegiPanjang.java

```
public  class MainPersegiPanjang {
    public static void main(String[] srgs) {
        PersegiPanjang pp = new PersegiPanjang();
        pp.setPanjang(10);
        pp.setLebar(20);
        System.out.println("Panjang : "+ pp.getPanjang());
        System.out.println("Lebar : "+ pp.getLebar());
        System.out.println("Luas : "+ pp.getLuas());
    }
}
```

Outputnya adalah :



5.3. Inheritance

Semua attribute dan method dari suatu kelas super dapat diwariskan ke subkelas. Dalam hirarki kelas, jika kelas C merupakan turunan kelas B, dan kelas B

merupakan turunan kelas A, maka otomatis attribute dan method kelas A juga diwariskan kelas C.

Bentuk pewarisan :

[modifier] class namaSubKelas extend namaKelasSuper {
// classBody
}

5.4. Manfaat Pewarisan

Tanpa inheritance, maka semua attribute dan method yang pernah dibuat dan bituhkan kelas lain, harus ditulis ulang seluruhnya.

Dengan inheritance, seorang programmer ingin memodifikasi suatu attribute atau method yang dimanfaatkan subkelas, maka dilakukan modifikasi attribute dan method tersebut pada kelas supernya.

5.5. Overrriding Attribute dan Method

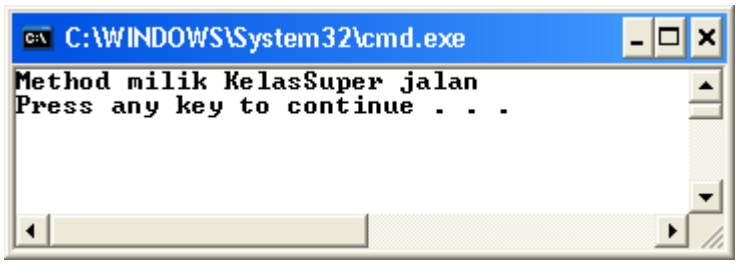
Overriding adalah kemampuan suatu subkelas untuk memodifikasi attribute dan method milik kelas supernya (tentu yang memiliki sifat private atau final tidak bisa dilakukan overriding).

Modifikasi yang dilakukan, misalnya jumlah parameter, tipe parameter, tipe return value, ataupun lingkungan pemrosesan datanya. Di bawah ini adalah contoh program overriding :

KelasSuper.java

```
class KelasSuper {
    public void methodAsli() {
        System.out.println("Method milik KelasSuper jalan");
    }
    public static void main(String[] args) {
        KelasSuper oks = new KelasSuper();
        oks.methodAsli();
    }
}
```

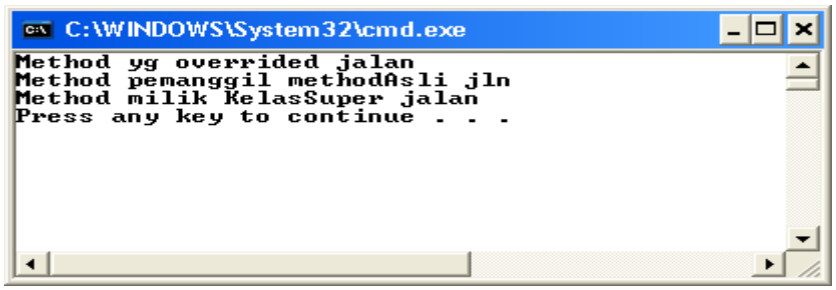
Outputnya adalah :



SubKelas.java

```
class SubKelas extends KelasSuper {
     public void methodAsli() {
        System.out.println("Method yg overrided jalan");
    }
    public void methodPemanggil (){
        System.out.println("Method pemanggil methodAsli jln");
        super.methodAsli();  // yg dipanggil milik kelas super
    }
    public static void main(String [] args) {
        SubKelas osk = new SubKelas();
        osk.methodAsli();
        osk.methodPemanggil();
    }
}
```

Outputnya adalah :



5.6. Menggunakan Method dan Constructor Kelas Super

Java tidak memperbolehkan subkelas memanggil contructor milik kelas supernya dengan cara hanya memanggil namanya.

Cara yang benar :
super();

super(tipe parameter);

Cara yang pertama, akan memanggil constructor default milik kelas supernya, sedangkan cara kedua, akan memanggil constructor kelas supernya yang sesuai dengan parameter tersebut.

Jika ada overriding, misal nama methodnya *namaMethod(),* maka cara pemanggilan method (**non static**) milik kelas super :

`super.namaMethod();` // perhatikan contoh pada 5.5

Sehingga dapat dibedakan `namaMethod()` milik siapa yang dipanggil.

5.7. Polymorphism

Polymorphism artinya bersifat poly morphy (memiliki banyak bentuk).

Method-method overloading masih dalam kelas yang sama, namun contoh berikut memvisualisaikan method `respon()` nama sama, namun pada kelas yang berbeda dapat memilki isi methoid yang berbeda pula tergantung kelasnya.

Contoh pertama :

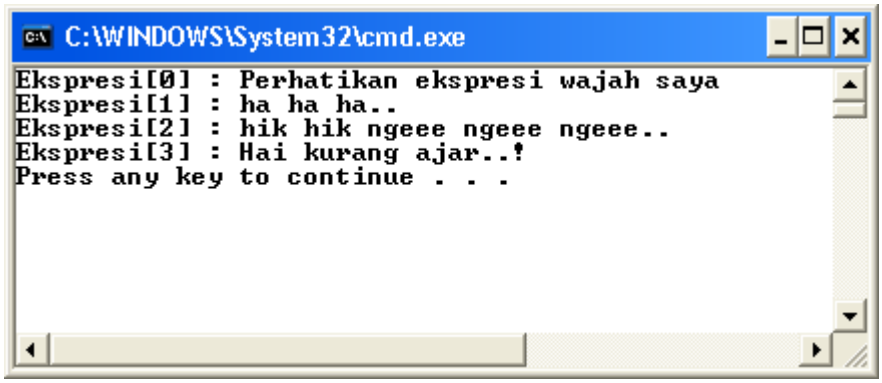
EkspresiWajah.java

```
class EkspresiWajah{
   public String respons() {
      return("Perhatikan ekspresi wajah saya");
   }
}
class Gembira extends EkspresiWajah{
   public String respons() {
      return("ha ha ha..");
   }
}
class Sedih extends EkspresiWajah{
   public String respons() {
      return("hik hik ngeee ngeee ngeee..");
   }
}
class Marah extends EkspresiWajah{
   public String respons() {
      return("Hai kurang ajar..!");
   }
}
```

MainEkspresiWajah.java

```
class MainEkspresiWajah{
  public static void main(String args[]) {
    EkspresiWajah objEkspresi = new EkspresiWajah();
      Gembira objGembira = new Gembira();
      Sedih objSedih = new Sedih();
      Marah objMarah = new Marah();

      EkspresiWajah[] arrEkspresi = new EkspresiWajah[4];
      arrEkspresi[0] = objEkspresi;
      arrEkspresi[1] = objGembira;
      arrEkspresi[2] = objSedih;
      arrEkspresi[3] = objMarah;

      System.out.println("Ekspresi[0]: "+arrEkspresi[0].respons());
      System.out.println("Ekspresi[1]: "+arrEkspresi[1].respons());
      System.out.println("Ekspresi[2]: "+arrEkspresi[2].respons());
      System.out.println("Ekspresi[3]: "+arrEkspresi[3].respons());
    }
}
```

Output contoh pertama :



Contoh kedua :

Employee.java

```
public class Employee {
   private String name;
   private double salary;
   protected Employee(String n, double s) {
      name = n;
      salary = s;
   }
```

```
    protected String getDetails() {
        return "Name : "+name+ "\nSalary : "+salary;
    }
    public void cetak() {
        System.out.println("Coba di Employee");
    }
}
```

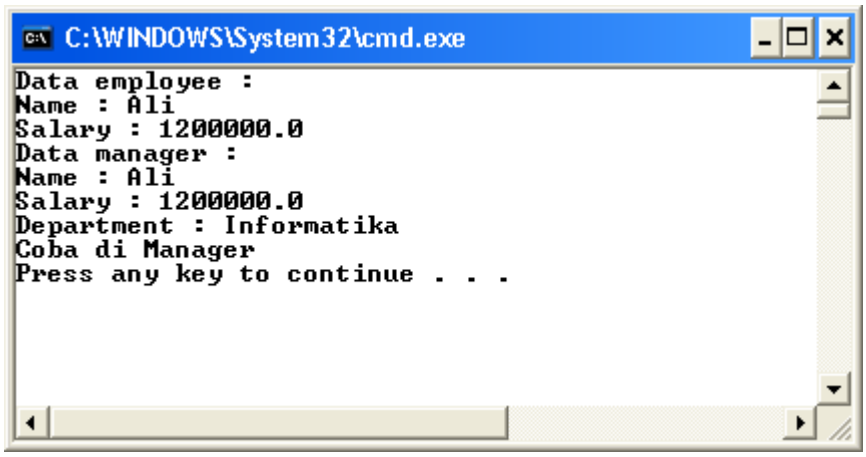
Manager.java

```
public class Manager extends Employee {
    private String department;
    public Manager(String nama, double salary, String dep) {
        super(nama, salary);
        department = dep;
    }
    public String getDepartment() {
        return department;
    }
    public String getDetails() {
        return super.getDetails()+"\nDepartment : "+getDepartment();
    }
    public void cetak() {
        System.out.println("Coba di Manager");
    }
}
```

View.java

```
public class View {
    public static void main(String[] args) {
       Employee e = new Employee("Ali",1200000);
       Employee em = new Manager("Ali",1200000,"Informatika");
       System.out.println("Data employee :\n"+e.getDetails());
       System.out.println("Data manager :\n"+em.getDetails());
       em.cetak();
        }
}
```

Outputnya adalah :



Catatan :

Kalau method cetak() di kelas Employee dan kelas Manager ada, maka yang dijalankan adalah method milik kelas Manager. Prioritasnya adalah kelas Manager kemudian kelas Employee.

5.8. Soal Latihan

1. Buatlah sebuah superclass yang bernama Kendaraan, dimana kendaraan mempunyai : Roda, kemudi(stang), sadel, dan mempunyai action : jalankan, rem

   Buatlah subclass Motor yang inherit superclass Kendaraan, dengan atribut jumlahroda=2, dan mempunyai method tambahan jumping.

   Buatlah subclass Mobil yang inherit superclass Kendaraan, dengan attribute jumlahroda=4 dan mempunyai method tambahan mudur.
2. Buatlah class sederhana yang di dalamnya terkandung information hiding dan encapsulation!

BAB VI
KELAS INNER, KELAS ABSTRACT, DAN INTERFACE

1. Kelas Inner

Java membolehkan programmer mnyisipkan suatu kelas ke dalam kelas lainnya. Kelas sisipan ini disebut kelas Inner.

Kelas Inner berguna untuk mendukung suatu proses yang akan dijalankan oleh kelas luarnya.

Beberpa ketentuan kelas Inner :

a) Kelas Luar yang mengandung kelas Inner, bila dikompilasi akan menghasilkan dua file *.class, yaitu *Luar.class* dan *Luar$Inner.class*

b) Kelas Inner boleh tidak diberi nama, yang disebut Anonymous Inner.

c) Kelas Inner dapat diberi modifier akses public, atau protected, atau default, ataupun private.

d) Untuk mengakses referensi this dari kelas luar digunakan bentuk NamaKelasLuar.this.

e) Kelas Luar ikut bertanggung-jawab dalam instansiasi kelas Inner (yang non static) . Kalau objek kelas Luar adalah a, dan objek kelas Inner adalah b, maka sintaks yang benar adalah :

```
Luar a = new Luar();
Luar.Inner b = a.new Inner();
```

f) Jika kelas Inner bersifat static, maka objek milik kelas Inner dapat dibuat sendiri tanpa melalui kelas Luarnya, (Artinya kelas Inner tidak dapat mengakses attribute ataupun method non static milik kelas Luarnya).

2. Menggunakan Kelas Inner

Kelas Inner lazim digunakan untuk membuat *handler* di method *main()* pada suatu aplikasi GUI.

Handler merupakan bagian program yang akan memproses event-event yang dipicu ketika user berinteraksi dengan komponen-komponen GUI.

Contoh program berikut adalah aplikasi Button sederhana dengan handlernya dari kelas Inner :

DemoJButtonInner.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class DemoJButtonInner extends JFrame {
    private JButton btn;
    public DemoJButtonInner (){
        super("Demo JButton Inner Class");
        Container c = getContentPane();
        c.setLayout(new FlowLayout());
        btn = new JButton("Button");
        c.add(btn);
        // membuat event handler
        ButtonHandler handler = new ButtonHandler();
        btn.addActionListener(handler);
        setSize(275, 100);
        show();
    }
    public static void main(String args[]) {
              DemoJButtonInner app = new DemoJButtonInner();
              app.addWindowListener(new WindowAdapter() {
                  public void windowClosing(WindowEvent we) {
                      System.exit(0);
                  }
              });
          }




// kelas Inner untuk Event Handling pada button
      private class ButtonHandler implements ActionListener {
                public void actionPerformed(ActionEvent ae) {
                   JOptionPane.showMessageDialog(null,
                           "anda telah menekan"
                           +ae.getActionCommand()+"\n"
                           +"Handler button ini pakai kelas Inner");
                    }
                }
            }
```

Output :

3.1. Analogi Kelas yang Abstract

Suatu kelas dapat diinstansiasi menjadi objek, misal kelas Dosen dapat diinstansiasi menjadi budi, heri, heru, namun tidak mungkin dapat menginstansiasi kelas MahlukHidup, kelas Hewan, dan kelas Manusia, sebab kelas tersebut terlalu umum (abstract), kelas seperti inilah yang disebut kelas abstract. Dibutuhkan kelas turunan yang lebih khusus.

3.2. Analogi Method yang Abstract

Bila kelas MahlukHidup mempunyai method bernafas, maka tidak dapat ditentukan cara suatu mahluk hidup tersebut bernafas (dengan paru-paru, insang, atau stomata), method seperti inilah yang disebut method abstract. Dibutuhkan kelas turunan yang khusus dan method override dari method yang abstract.

4. Interface

Interface adalah kelas yang paling abstract, yang berisi daftar deklarasi method (seluruh method belum memiliki implementasi).

4.1. Analogi Interface

Interface dapat dianalogikan sebagai kontrak/profesi/peran yang dipakai oleh setiap kelas.

Dalam kehidupan nyata dapat diketahui ada manusia yang bekerja sebagai da'i, dosen, tentara, penyanyi, pengacara, dan sebagainya, tentunya manusia-manusia tersebut selain harus memiliki method standard sebagai seorang manusia, juga harus memiliki method yang sesuai dengan pekerjaannya.

Dengan demikian untuk membuat objek seorang budi bekerja sebagai dosen, harus dibuat kelas yang merupakan turunan kelas manusia yang meng-implementasikan interface dosen.

5. Deklarasi Interface

Bentuk umum deklarasi:

```
[modifier] interface NamaInterface {
    // deklarasi konstanta
    // deklarasi method
}
```

Catatan : modifier static tidak boleh digunakan dalam interface

6. Implementasi Interface

Bentuk umum implementasi :

```
[modifier] class NamaKelas implements NamaInterface {
    // penggunaan konstanta
    // implementasi method
}
```

7. Contoh Abstract Class dan Interface

Hewan.java

```
abstract class Hewan {
    protected String nama;
    protected int jumKaki;
    protected boolean bisaTerbang = false;

    public Hewan(String nama, int kaki, boolean terbang) {
        this.nama = nama;
```

```
        jumKaki = kaki;
        bisaTerbang = terbang;
    }
    public abstract void bersuara();

    public  static void  makan() {
        System.out.println("nyam, nyam, nyam");
    }
    public void isHewan() {
        System.out.println("nama : "+nama);
        System.out.println("jumlah kaki : "+jumKaki);
        System.out.println("bisa terbang : "+bisaTerbang);
    }
}
```

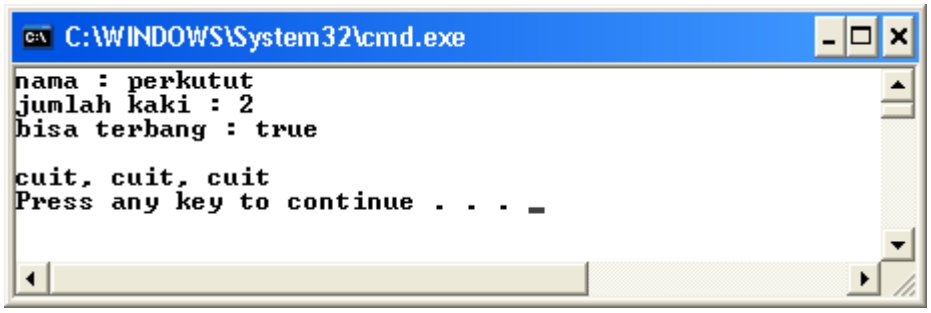
Manusia.java

```
Interface Manusia {
    public void menyanyi();
    public void ketawa();
}
```

Perkutut.java

```
class Perkutut extends Hewan {
    public Perkutut() {
        super("perkutut",2,true);
    }
    public void bersuara() {
        System.out.println("\ncuit, cuit, cuit");
    }
    public static void main(String[] args) {
        Perkutut p = new Perkutut();
        p.isHewan();
        p.bersuara();
    }
}
```

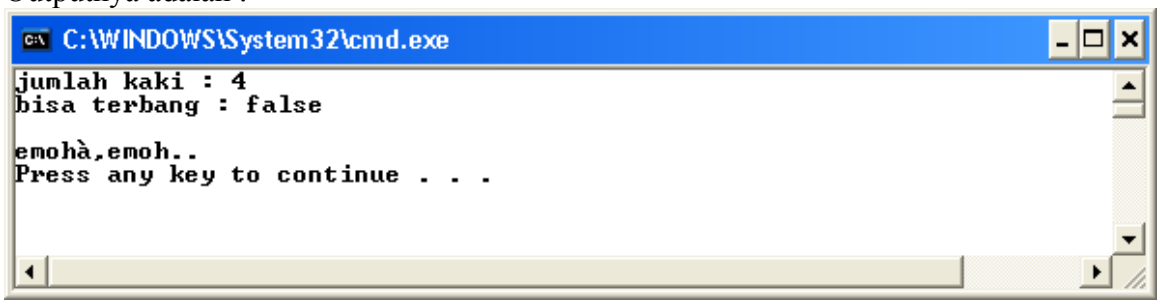
Outputnya adalah :



Sapi.java

```
class Sapi extends Hewan {
    public Sapi() {
        super("sapi", 4, false);
    }
    public void bersuara() {
        System.out.println("\nemoh…,emoh..");
    }
    public static void main(String[] args) {
        Sapi s = new Sapi();
        s.isHewan();
        s.bersuara();
    }
}
```

Outputnya adalah :

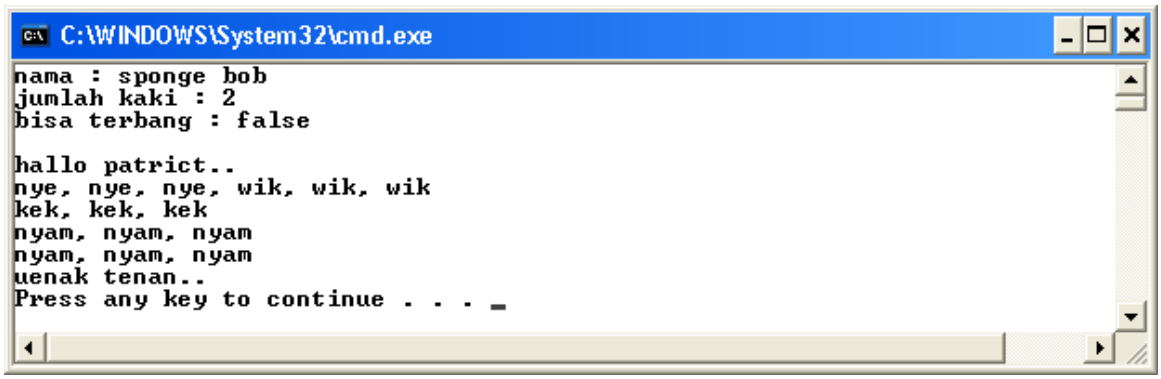


SpongeBob.java

```
class SpongeBob extends Hewan implements Manusia {
    public SpongeBob() {
        super("sponge bob", 2, false);
    }
    public void bersuara() {
        System.out.println("\nhallo patrict..");
    }
    public void menyanyi() {
        System.out.println("nye, nye, nye, wik, wik, wik");
    }

    public void ketawa() {
        System.out.println("kek, kek, kek");
    }
    public static void makan() {
        System.out.println("uenak tenan..");
    }
    public void makan2() {
        super.makan();
    }
    public static void main(String[] args) {
        SpongeBob s=  new SpongeBob();
        s.isHewan();
```

```
        s.bersuara();
        s.menyanyi();
        s.ketawa();
        s.makan2();
        Hewan.makan();
        makan();
    }
}
```

Outputnya adalah :



8. Interface vs Class

Perbandingan interface dengan class :

komponen	interface	class
definisi	daftar deklarasi method	model objek
kandungan informasi	isi/implementasi semua methodnya berada di luar interface ini	mendefinisikan attribute dan method secara rinci dan konkret
instansiasi	tidak boleh	boleh

9. Interface vs Inheritance

Inheritance adalah proses pewarisan attribute dan method dari satu kelas super kepada satu/lebih subkelas.

Bagaimana kalau dibutuhkan suatu kelas yang attribute dan methodnya berasal dari lebih dari satu kelas super ? disinilah keterbatasan inheritance, namun interface

berperan, karena dalam interface bisa dimasukkan method-method dari beberapa library referensi tanpa harus menurunkannya.

Syntax kelas yang menggunakan lebih dari satu interface :

```
[modifier]    class    NamaKelas    implements    NamaInterface1,
NamaInterface2, … {
    //interfaceBody
}
```

Nama-nama interface tersebut dapat dijadikan tipe data attribute ataupun tipe data parameter dalam kelas yang menggunakan.

10. Interface vs Abstract Class

Interface dan kelas abstract memilki kesamaan sama-sama tidak boleh instansiasi objek.

Perbedaan interface dangan kelas abstract adalah sebagai berikut :

Komponen	interface	abstract class
Attribute	hanya berupa konstanta	bebas memilki tipe data apa saja
Method	berupa deklarasi	boleh deklarasi, boleh berupa method lengkap
Syntax	seluruhnya abstract (berupa deklarasi) Boleh sebagian yg diimplementasikan	sebagian abstract Method yg abstrak harus semua diimplementasikan subclassnya

11. Aplikasi Interface

Interface sering digunakan untuk menambah *event handling* pada program aplikasi GUI, perhatikan contoh berikut :

AppJButton.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class AppJButton extends JFrame implements ActionListener {
    private JButton btn;
    public AppJButton() {  // constructor
        super("Demo JButton Interface");
        Container c = getContentPane();
        c.setLayout(new FlowLayout());
        btn = new JButton("Button");
        c.add(btn);

        btn.addActionListener(this);
        setSize(275, 100);
        show();
    }
    public static void main(String args[]) {
        AppJButton app = new AppJButton();
        app.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });
    }
    public void actionPerformed(ActionEvent ae) {
       JOptionPane.showMessageDialog(null,"Anda    telah    menekan:
"+ae.getActionCommand()+"\n"+
           "Handler Button ini mengimplementasikan interface");
    }
}
```

Outputnya :

bila komponen berupa tombol Button di click akan muncul window pesan :

D. Rangkuman

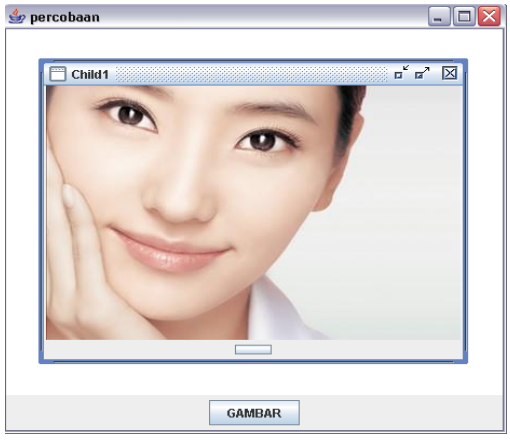
Inner kelas adalah satu atau beberapa kelas yang disisipkan pada kelas lainnya yang berguna untuk mendukung suatu proses yang akan dijalankan oleh kelas luarnya. Ada beberapa bentuk inner kelas seperti yang telah disebutkan sebelumnya.

Abstract class adalah suatu kelas yang dinyatakan abstract yang umumnya memiliki satu atau lebih abastract method. Abstract method adalah suatu method yang belum memiliki implementasi dan menggunakan modifier abstract. Abstract class biasanya dijadikan parent atau super class dari kelas-kelas yang dapat membuat object.

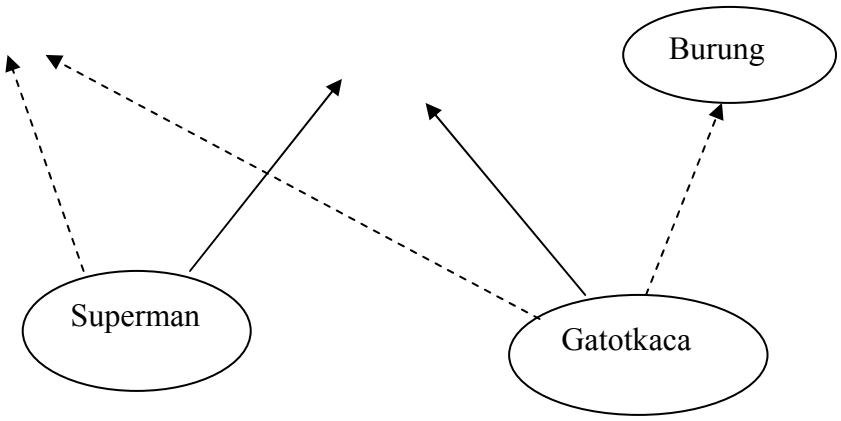
Sedangkan interface adalah kelas yang berisi method-method tanpa implementasi, namun tanpa modifier abstract, apabila suatu interface memiliki atribut, maka atributnya akan berlaku sebagai konstanta (static final).

E. Soal Latihan

1. Dengan menggunakan konsep inner class, buatlah program yang memunculkan sebuah frame induk yang memiliki sebuah button sebagai elemen, yang jika tombol tersebut ditekan akan muncul frame anak yang memuat gambar di dalam frame induk seperti contoh berikut :



2. Bentuklah kode program yang merepresentasikan bagan dibawah ini !


Burung
Superman
Gatotkaca

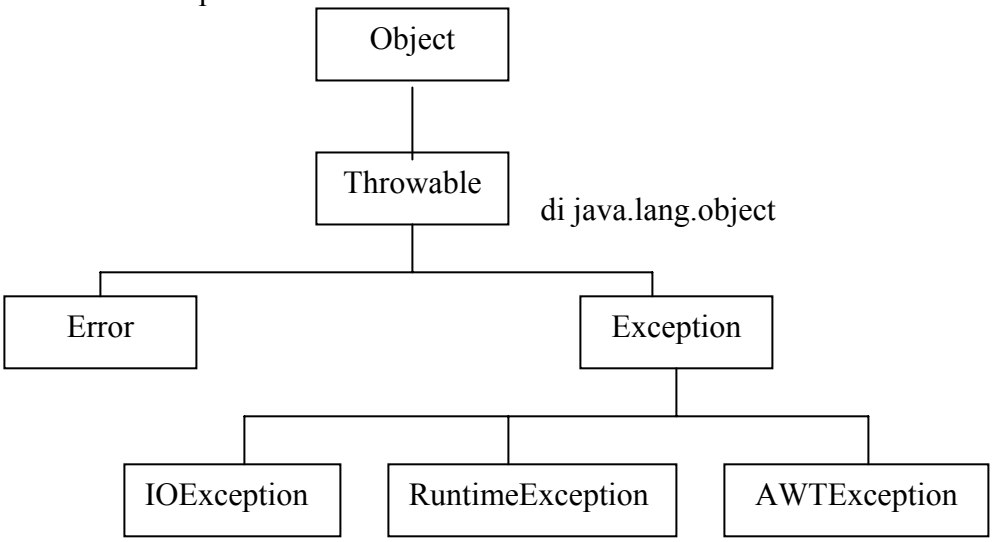
BAB VII
EXCEPTION HANDLING

7.1. Pendahuluan

Error (kesalahan) dalam pemrograman dibagi dalam tiga katagori yaitu *syntax error* (muncul saat kompilasi), *run time error*, dan *logic error* (ketika output belum sesuai dengan yang diharapkan).

Exception digunakan sebagai sarana untuk melaporkan jenis kondisi kesalahan saat program dijalankan, dan mengendalikannya agar *run time error* tersebut tidak mengakibatkan eksekusi dihentikan (statement setelahnya tetap dieksekusi).

7.2. Jenis-Jenis Exception



Dalam java, exception merupakan objek dari subkelas yang diturunkan dari kelas *Throwable*. Kelas *Throwable* ini terdapat dalam package *java.lang.object*. Kelas ini mengandung dua sub kelas berikut :

a. Kelompok Kelas Error

Error ini bersifat fatal sehingga sistem tidak dapat dimanipulasi, contoh kelas: *LinkageError*, *VirtualMachineError*, dan *AWTError*.

b. Kelompok Kelas Exception

Jenis error ini masih dapat diantisipasi dengan menyisipkan statement tambahan untuk mendeteksi statement penyebab dan jenisnya.

Ada kelompok *RuntimeException* yang diperiksa oleh interpreter, apakah akan ditangani atau dilempar, namun ada pula exception yang tidak diperiksa interpreter.

Disamping itu programmer dibolehkan membuat exception sendiri dengan cara *extends* atau *implements* kelas *Exception*.

Tabel Checked Exception

No	Exception	Deskripsi
1	*ClassNotFoundException*	Kelas tidak ditemukan
2	*CloneNotSupportedException*	melakukan clone objek yang tidak mengimplementasikan interface *Cloneable*
3	*IllegalAccessException*	Pengaksesan ke kelas ditolak
4	*InstantiationException*	Menciptakan objek dari kelas abtract ataupun dari interface
5	*InterruptedException*	Thread telah diinterupsi oleh thread lain
6	*NoSuchFieldException*	Field yang diminta tidak ada
7	*NoSuchMethodException*	Method yang diminta tidak ada

Tabel Uncheked Exception

No	Exception	Deskripsi
1	*AritmaticException*	Kesalahan Aritmatik seperti pembagian dengan nol
2	*ArrayIndexOutBoundsException*	Index array di luar batas
3	*ArrayStoreException*	Pemberian nilai ke elemen array tidak sesuai dengan tipenya
4	*ClassCastException*	Cast yang tidak sah
5	*IllegalArgumentException*	Argument illegal
6	*IllegalMonitorStateException*	Operasi monitor illegal seperti menunggu di thread yang tidak terkunci

7	*IllegalStateException*	Lingkungan atau aplikasi state yang tidak benar
8	*IllegalThreadStateException*	Operasi yang diminta tidak kompatibel dengan state thread saat itu
9	*IndexOutOfBoundException*	Indeks di luar batas
10	*NegativeArraySizeException*	Array diciptakan dengan ukuran negatif
11	*NullPointerException*	Penggunan null yang tidak sah
12	*NumberFormatException*	Konversi yang tidak sah dari string ke format numerik
13	*SecurityException*	Melanggar aturan security
14	*StringIndexOutOfBounds*	Index di luar batas string
15	*UnsupportedOperationException*	Ditemukan operasi yang tidak didukung

7.3. Mengantisipasi Exception

Diperlukan tiga langkah berikut ini untuk mengantisipasi exception :

a. Mendeklarasikan Exception
Bentuk umum :

[modifier] returntype namaMethod() throws tipeException{
}

Contoh :
```
public void operasiMatematika() throws IOException,
   ClassNotFoundException {
}
public void beriPinjaman() throws TolakException{
}
```

b. Melempar Exception

Bentuk umum :

```
TipeException namaObjek = new TipeException();
throw namaObjek;
```

Diringkas menjadi :

throw namaObjek TipeException();

atau

throw new TipeException();

Contoh :

```
TolakException t = new TolakException(”lagi pelit”);
throw t;
```

Diringkas menjadi :

throw new TolakException(”lagi pelit!”);

c. Menangkap Exception

Penangkapan runtime error, dapat mempunyai beberapa blok yang menangkap setiap jenis exception.

Bentuk umum :

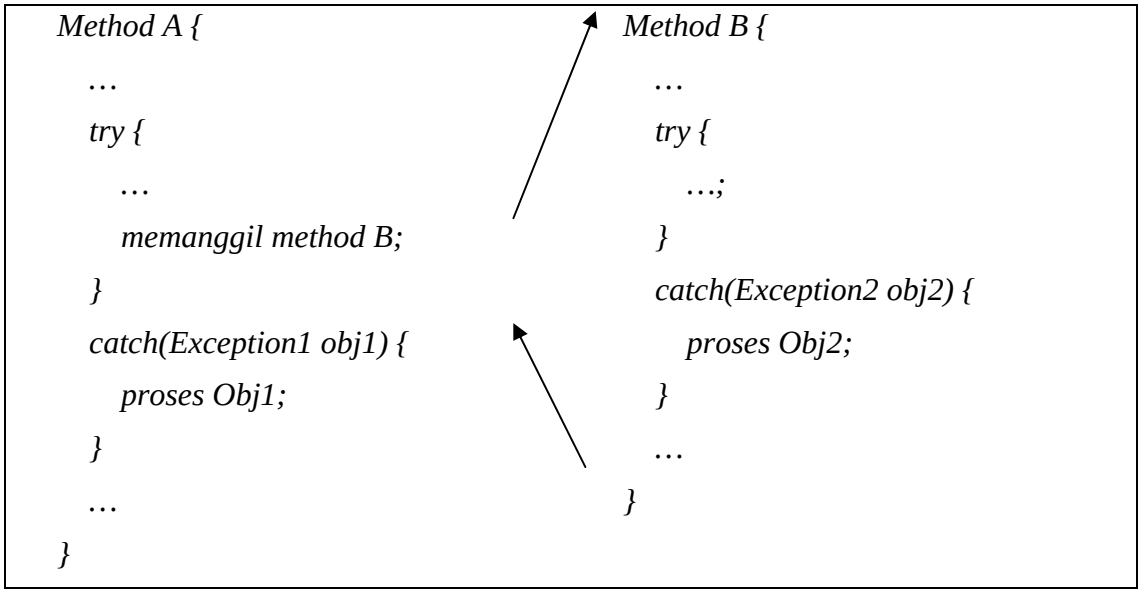
```
try {
  // pemanggilan method yg mungkin menghasilkan exception
   // blok statement yg mungkin menghasilkan exception
}
catch(TipeException1 namaObjek) {
   // penanganan salah-satu jenis exception
}
catch(TipeException2 namaObjek) {
   // penanganan salah-satu jenis exception
}
catch(TipeExceptionN namaObjek) {
   // penanganan salah-satu jenis exception
}
finally {
   // blok yang harus dieksekusi
}
```

Jika pada blok *try* tidak terjadi *exception*, maka blok *catch* tidak ada yang dieksekusi dan segera blok *finally* yang dieksekusi.

Jika terjadi *exception* pada blok *try*, maka salah satu blok *catch* dieksekusi, kemudian blok *finally* dieksekusi.

7.4. Mekanisme Mengantisipasi Exception

Ada tiga kemungkinan skenario exception, pertama jika tidak terjadi exception (tidak ada blok catch yang dieksekusi), kedua jika exception terjadi pada blok method tunggal (salah-satu blok catch dieksekusi), ketiga jika terjadi exception pada blok tersarang.



7.5. Menampilkan Pesan Exception

Beberapa method standard yang dapat digunakan untuk menampilkan pesan exception merupakan anggota dari kelas *java.lang.Throwable*.

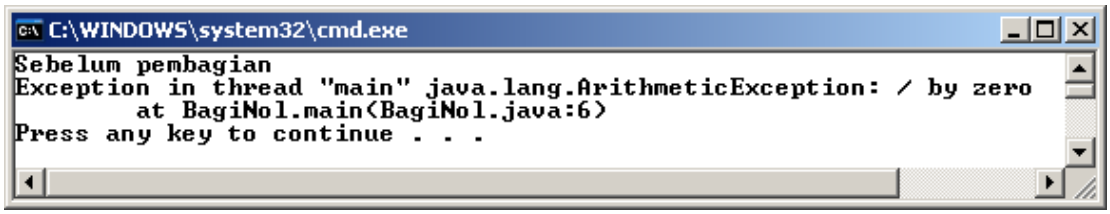
No	Method Pesan Exception	Deskripsi
1	*getMessage()*	Mengembalikan nilai string yang berisi pesan rinci tentang objek Throwable yang mengalami exception
2	*toString()*	Mengembalikan nilai string yang berisi pesan singkat tentang objek yang mengalami exception
3	*getLocalizedMessage()*	Menampilkan pesan exception lokal (yang terjadi pada subkelas saja)
4	*printStackTrace()*	Method ini bersifat void, dan hanya mencetak informasi tentang objek Throwable

7.6. Contoh Aplikasi Exception Handling

Contoh1 :

```
public class BagiNol {
   public static void main(String[] args) {
      System.out.println("Sebelum pembagian");

    // try {
         System.out.println(5/0);
    // }
    // catch (Throwable e) {
    //    System.err.println("Terjadi pembagian dengan nol");
    //    System.err.println("Pesan dari Sun : "+e.getMessage());
    // }
      System.out.println("Sesudah pembagian");
   }
}
```

Output Contoh1 :



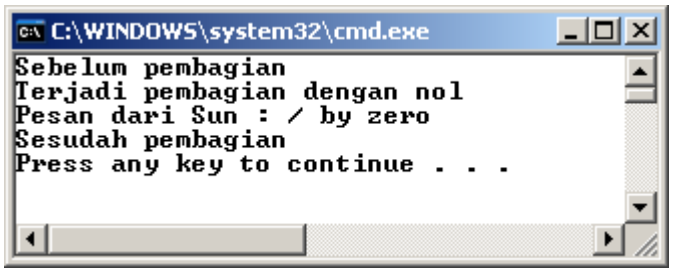
Terlihat hasil output contoh1 ternyata statement sesudah pembagian tidak pernah dieksekusi, bahkan program ini memberikan perluang terjadinya *hang* saat eksekusi, program tersebut diperbaiki dengan contoh2 berikut ini :

Contoh2 :

```
public class BagiNol {
   public static void main(String[] args) {
      System.out.println("Sebelum pembagian");

      try {
         System.out.println(5/0);
      }
```

```
        catch (Throwable e) {
            System.err.println("Terjadi pembagian dengan nol");
            System.err.println("Pesan dari Sun : "+e.getMessage());
        }
        System.out.println("Sesudah pembagian");
    }
}
```

Output Contoh2 :



Terlihat output contoh2 bahwa error tidak mengakibatkan terhentinya eksekusi program, bahkan statement berikutnya setelah statement penyebab error tetap dieksekusi.

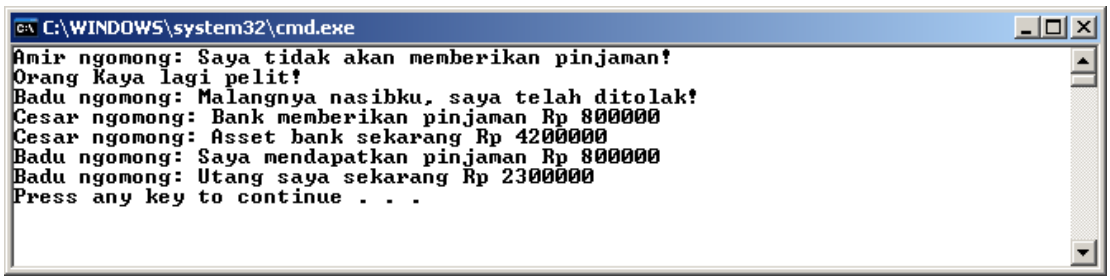
Contoh3 :

```
public class TolakException extends Exception{
    public TolakException(String pesan) {
        super(pesan);
    }
}
public interface Kreditor {
    public void beriPinjaman(int vduit) throws TolakException;
}

public class Orang {
    protected String nama;
    public Orang(String vnama) {
        nama = vnama;
    }
    public String getNama() {
        return nama;
    }
```

```
    public void ngomong(String message) {
    System.out.println(nama + " ngomong: " + message);
    }
}

public class Bank implements Kreditor {
    private Orang manager;
    private int asset;
    public Bank(String vmanager, int vasset) {
        manager = new Orang(vmanager);
        asset = vasset;
    }
    public void beriPinjaman(int vduit) throws TolakException {
        if (vduit < 1000000) {
            asset = asset - vduit;
            manager.ngomong("Bank memberikan pinjaman Rp " + vduit);
            manager.ngomong("Asset bank sekarang Rp " + asset);
        }
        else {
            manager.ngomong("Bank tidak akan memberikan pinjaman!");
            throw new TolakException("Bank lagi pelit!");
        }
    }
}
public class OrangKaya extends Orang implements Kreditor {
    private int duit;
    public OrangKaya(String vnama, int vduit) {
        super(vnama);
        duit = vduit;
    }
    public void beriPinjaman(int vduit) throws TolakException{
        if (vduit < 500000) {
            duit = duit - vduit;
            ngomong("Saya memberikan pinjaman Rp " + vduit);
            ngomong("Duit saya sekarang Rp " + duit);
        }
        else {
            ngomong("Saya tidak akan memberikan pinjaman!");
            TolakException t = new TolakException("Orang Kaya lagi pelit!");
            throw t;
        }
    }
}
```

```
public class OrangMiskin extends Orang {
   private int utang;
   public OrangMiskin(String vnama, int vutang) {
      super(vnama);
      utang = vutang;
   }
   public void pinjamDuit(Kreditor vkreditor, int vduit) {
      try {
         vkreditor.beriPinjaman(vduit);
         utang = utang + vduit;
         ngomong("Saya mendapatkan pinjaman Rp " + vduit);
         ngomong("Utang saya sekarang Rp " + utang);
      }
      catch (TolakException e) {
         System.out.println(e.getMessage());
         ngomong("Malangnya nasibku, saya telah ditolak!");
      }
   }
}

public class PinjamApp {
   public static void main(String args[]) {
      Bank bankRut = new Bank("Cesar", 5000000);
      OrangKaya amir = new OrangKaya("Amir", 2000000);
      OrangMiskin badu = new OrangMiskin("Badu", 1500000);
      try {
         badu.pinjamDuit(amir, 600000);
         badu.pinjamDuit(bankRut, 800000);
      }
      catch (Exception e) {
         System.out.println("Nggak pernah jalan!");
      }
   }
}
```

Outputnya :



7.7. Soal Latihan

1. Buatlah suatu program manajemen memori yang meghitung memori yang kita gunakan pada sebuah file eksekusi ! Sisipkan penanganan exception untuk error-error yang mungkin terjadi.

BAB VIII
KELAS-KELAS DASAR

Dalam pemrograman java, suatu string adalah **objek**. Ada dua kelas string yang akan dibahas yaitu kelas **String** dan kelas **StringBuffer**.

1. Kelas String

Kelas String memodelkan deretan karakter. Kelas ini terdapat dalam java.lang. Sesuai dengan kuantitas constructornya, ada 7 cara untuk membuat objek String:

```
String();               // cara1
String(String value);   // cara2
String(char value[]);   // cara3
String(byte ascii[], int hibyte);   // cara4
String(char value[], int offset, int count); // cara5
String(byte ascii[], int hibyte, int offset, int count);    //cara6
String(StringBuffer buffer);  // cara7
```

Contoh penggunaan Constructor tersebut :

```
String aString1 = new String();  // cara1
String aString2 = new String("haii…");  // cara2
char aArray[] = {'H','E','L','L','O'};
String aString3 = new String(aArray);  // cara3
String aString4 = new String(aArray,0,4);  // cara5
System.out.println(aString4); // HELL
```

Ada 33 method yang digunakan untuk melakukan 8 macam operasi pada kelas String, yaitu :

```
1. int length;
2. char charAt(int index);
3. boolean startsWith(String prefix);
4. boolean startsWith(String prefix, int tooffset);
5. boolean endsWith(String suffix);
6. int indexOf(int i);
7. int indexOf(int i, int mulaiIndex);
8. int indexOf(String str);
```

```
9. int indexOf(String str, int mulaiIndex);
10. int lastIndexOf(int i);
11. int lastIndexOf(int i, int mulaiIndex);
12. int lastIndexOf(String str);
13. int lastIndexOf(String str, int mulaiIndex);
14. String substring(int mulaiIndex);
15. String substring(int mulaiIndex, int sampaiIndex);
16. boolean equals(Object anObject);
17. boolean equalsIgnoreCase(String aString);
18. int compareTo(String str);
19. int compareTo(Object anObject);
20. String concat(String s);
21. String replace(char oldChar, char newChar);
22. String trim();
23. String toLowerCase();
24. String toUpperCase();
25. static String valueOf(Object anObject);
26. static String valueOf(char ch[]);
27. static String valueOf(char ch[], int offset, int count);
28. static String valueOf(boolean b);
29. static String valueOf(char ch);
30. static String valueOf(int i);
31. static String valueOf(long l);
32. static String valueOf(float f);
33. static String valueOf(double d);
```

Ada 8 operasi pada kelas String, yaitu :

1. Membuat dan menginisialisasi String

   String saran = ”Raihlah scjp”;
   String saran = new String(”Raihlah scjp”);

2. Membaca character dalam String

   int len = saran.length(); // 12
   char ch = saran.charAt(3); // h

3. Membandingkan dua String

```
String s1 = new String("abcd");
String s2 = new String("abcdz");
System.out.println(s1.equals(s2)); // true
System.out.println(s1.equalsIgnoreCase("ABCFJ");//true
System.out.println(s1.compareTo(s2);
```

hasilnya 0, jika s1 sama dengan s2

hasilnya < 0, jika s1 < s2

hasilnya > 0, jika s1 > s2

Catatan :

Membandingkan nilai-nilai di attribute int, long, float, dan double, harus mengunakan operator ==, namun operator ini tidak berlaku untuk objek-objek milik kelas String. Jika contoh di atas dilakukan **s1==s2**, maka hasilnya **selalu false**.

4. Mengubah character kecil menjadi capital

```
String s1 = new String("Raihlah scjp \n");
String s2 = s2.trim(); // Raihlah scjp (tanpa spasi)
System.out.println(s2.toLowerCase());
System.out.println(s2.toUpperCase());
```

5. Concatenation dua String

```
String s1 = new String("Saya belajar j2se");
String s2 = new String(s1+ " sendiri");
String s3 = new String(s2+ " di rumah");
System.out.println(s3);
```

6. Mencari character dan substring

```
String aString = new String("Nilai objek milik kelas String");
    int index1 = aString2.indexOf('a');
    // index1 berisi posisi 'a'
    int index2 = aString.indexOf('a',index1+1);
    // index2 berisi posisi kemunculan kedua huruf 'a'
    int index3 = aString.indexOf("String");
    // index3 berisi lokasi karakter 'S' pada aString
    int index4 = aString.indexOf("i");
    // index4 berisi lokasi karakter i pada aString
    boolean hasil1 = aString.startWith("Nilai"); // true
    boolean hasil = aString.endsWith("String"); // true
```

7. Mengambil suatu substring

```
String strAsli = new String("Saya belajar j2se");
String strBaru1 = strAsli.substring(12);
// strBaru1 berisi string mulai index  ke-12, "j2se""
String strBaru2 = strAsli.substring(5,11);
// strBaru2 berisi string mulai index ke-5 sampai index ke-11
// isinya "belajar"
```

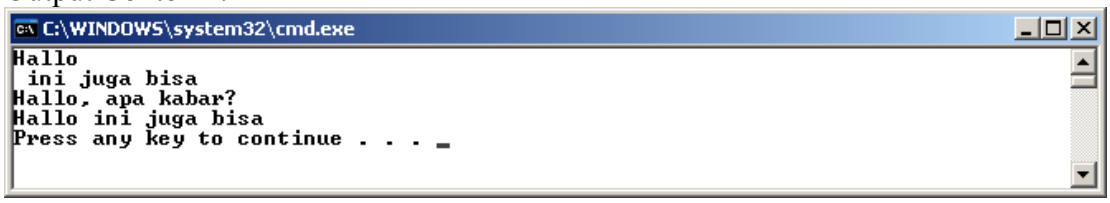
8. Menggantikan suatu character dalam String

```
String strBaru3 = strAsli.replace('S','m');
System.out.println(strBaru3);
// "maya belajar j2se"
```

1.1. Contoh Pertama

```
public class TestString {
  public static void main (String argv[]) {
    String s1;
    String s2;
    String s3;
    String s4;
    s1 = "Hallo";
    s2 = new String(" ini juga bisa");
    s3 = s1+", apa kabar?";
    s4 = s1+s2;
    System.out.println(s1);
    System.out.println(s2);
    System.out.println(s3);
    System.out.println(s4);
  }
}
```

Output Contoh1 :

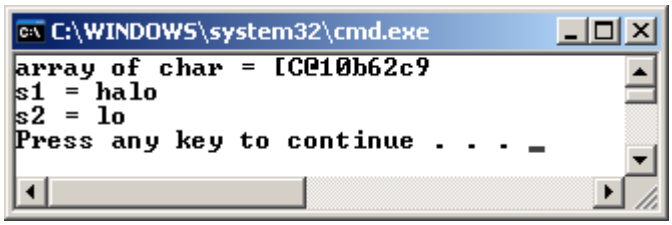


1.2. Contoh Kedua

```
public class Strkonstruktor {
  public static void main(String argv[]) {
    char[] arrayChar = {'h', 'a', 'l', 'o'};
    String s1 = new String(arrayChar);
```

```
        String s2 = new String(arrayChar,2,2);
        // mulai elemen kedua, sebanyak dua karakter, berisi ”lo”

        System.out.println("array of char = " + arrayChar);
        // mestinya didisplay per karakter

        System.out.println("s1 = " + s1);
        System.out.println("s2 = " + s2);
      }
    }
```

Output Contoh Kedua



1.3. Contoh Ketiga

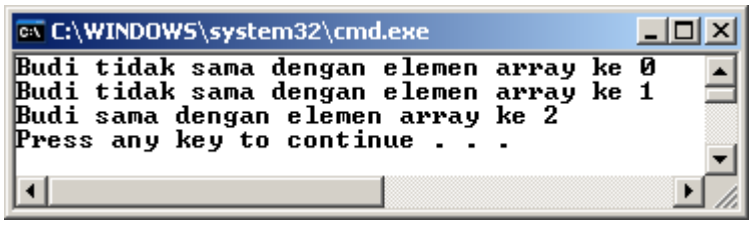
```
public class Caristr {
  public static void main(String argv[]) {
    String str = "Ini adalah # String yang # diproses";
    boolean found = false;
    int i;
    for (i = 0; i < str.length(); i++)
      if (str.charAt(i) == '#') {
        found = true;
        break;
      }
      if (found)
        System.out.println("Posisi '#' pertama ada pada index ke " + i);

      // cari char '#' mulai dari index ke 12
      System.out.println("Posisi '#' kedua ada pada index ke " + str.indexOf('#', 12));
      System.out.println("Posisi 'adalah' mulai index ke " + str.indexOf("adalah"));
  } // akhir dari for
} // akhir dari kelas
```

Output Contoh Ketiga :

1.4. Contoh Keempat

```
public class Compare {
  public static void main(String argv[]) {
    String tabel[] = {"Willis", "Richard", "Budi", "Novrido"};
    String nama1 = "Budi";
    String nama2 = "Heru";
    for (int i = 0; i < tabel.length; i++) {
      if (tabel[i].equals(nama1)==false) {
        System.out.println(nama1 + " tidak sama dengan elemen array ke " + i);
      }
      else {
        System.out.println(nama1 + " sama dengan elemen array ke " + i);
        break;
      }
    }
  } // akhir method
} // akhir kelas
```

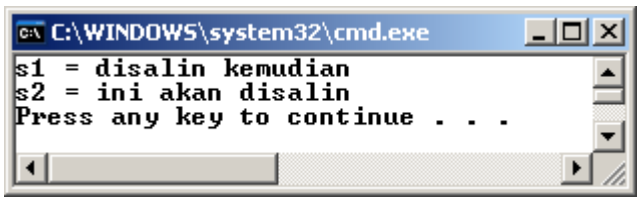
Output Contoh Keempat :



1.5. Contoh Kelima

```
public class SubStr {
  public static void main(String argv[]) {
    String s = "Kalimat ini akan disalin kemudian";
    String s1 = s.substring(17);
    String s2 = s.substring(8,24); // dari index ke-8 sampai index ke-24
    System.out.println("s1 = " + s1);
    System.out.println("s2 = " + s2);
  }
}
```

Output Contoh Kelima :



2. Kelas StringBuffer

Isi objek dari kelas String, bersifat konstanta, sehingga diperlukan kelas StringBuffer yang objeknya berupa variabel string yang dapat diedit dengan methodnya yang khusus seperti append(String), insert(posisi, String), setCharAt(posisi, String), walaupun kelas String juga memiliki method replace(String, String) untuk mengedit isi objeknya.

Ada tiga constructor milik kelas StringBuffer :

```
1. public StringBuffer();
2. public StringBuffer(int n);
3. public StringBuffer(String str);
```

Contoh pembuatan objek melalui constructor milik kelas StringBuffer :

StringBuffer sb1 = new StringBuffer();
// membuat objek sb1 yang kosong

String sb2 = new StringBuffer(32);
// membuat objek sb2 yang panjangnya 32 karakter

String sb3 = new String(”Belajar StringBuffer”);
// membuat objek sb3 yang berisi string tersebut

String sb4 = new StringBuffer(sb2);
// membuat objek sb4 yang isinya sama dengan sb2

Beberapa method untuk memodifikasi isi buffer yang berisi string :

StringBuffer s1 = new StringBuffer(14);
System.out.println(”Kapasitas = ”+ s1.capacity()); //14
System.out.println(”Panjang = ”+ s1.length()); // 0
s1.setLength(3);
System.out.println(s1); // Bel
System.out.println(”Kapasitas = ”+ s1.capacity()); //14
System.out.println(”Panjang = ”+ s1.length()); //3

Contoh penggunaan method charAt() dan setCharAt() :

StringBuffer s1 = new StringBuffer(”Belajar StringBuffer”);
char c1 = s1.charAt(9);
System.out.println(c1); // S
s1.setCharAt(4,’r’);
*System.out.println(s1); // Bel**r**jar StringBuffer*
Beberapa overloading method append() dan insert() :

```
synchronized StringBuffer append(Object obj);
synchronized StringBuffer append(String str);
synchronized StringBuffer append(char ch);
synchronized StringBuffer append(char ch[]);
synchronized StringBuffer append
(char str[],int offset,int len);
StringBuffer append(boolean b);
StringBuffer append(int i);
StringBuffer append(long l);
StringBuffer append(float f);
StringBuffer append(double d);
synchronized StringBuffer insert(int offset, Object obj);
synchronized StringBuffer insert(int offset, String str);
synchronized StringBuffer insert(int offset, char ch);
synchronized StringBuffer insert(int offset, char ch[]);
StringBuffer insert(int offset, boolean b);
StringBuffer insert(int offset, int i);
StringBuffer insert(int offset, float f);
StringBuffer insert(int offset, double d);
```

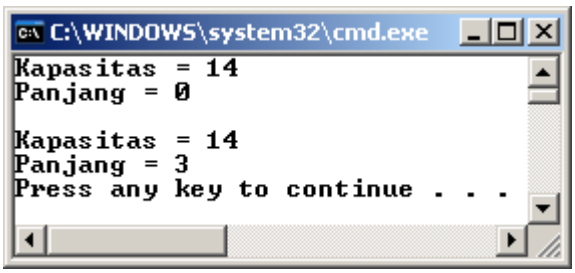
Contoh pengunaan method *append()* dan *insert()* :

```
StringBuffer sb1 = new StringBuffer(”2 + 2 = ”);
StringBuffer sb2 = new StringBuffer(”Raihlah scjp”);
sb1.append(2 + 2);
sb2.append(” depan”);
sb2.insert(14, ”tahun”);
System.out.println(sb1);
System.out.println(sb2);
```

2.1. Contoh Pertama

```
public class PanjangKapasitas {
  public static void main(String argv[]) {
    StringBuffer s1 = new StringBuffer(14);
    //kapasistas = panjang maksimum
        System.out.println("Kapasitas = "+ s1.capacity()); //14
        System.out.println("Panjang = "+ s1.length()); // 0
        s1.setLength(3);
        System.out.println(s1);
        System.out.println("Kapasitas = "+ s1.capacity()); //14
        System.out.println("Panjang = "+ s1.length()); // 3
  }
}
```

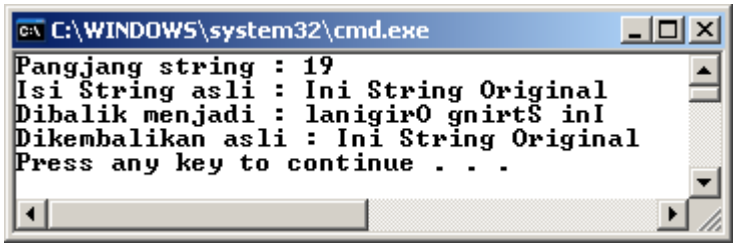
Output Contoh Pertama :



2.2. Contoh Kedua

```
public class MembalikString {
  public static void main(String argv[]) {
    String str = "Ini String Original";
    //           1234567890123456789

    int panjang = str.length();
    System.out.println("Panjang string : "+panjang);
    StringBuffer asli = new StringBuffer(1);
    StringBuffer asli2 = new StringBuffer(1);
    StringBuffer balik = new StringBuffer(1);

    char ch;
    for (int i = (panjang-1); i >= 0; i--) {
      ch = str.charAt(i);
      balik.append(ch);  // balik.append(charAt(i));
    }
    System.out.println("Isi String asli : "+str);
    System.out.println("Dibalik menjadi : "+balik);
    asli2 = balik.reverse();
    System.out.println("Dikembalikan asli : "+asli2);
  }
}
```

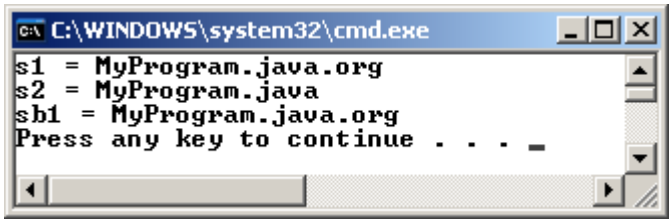
Output Contoh Kedua :



2.3. Contoh Ketiga

```
public class LastIndex {
  public static void main(String argv[]) {
    String s1 = "MyProgram.java.org";
    String s2 = s1.substring(0, s1.lastIndexOf('.'));
    StringBuffer sb = new StringBuffer(s2);
    sb.append(".org");
    System.out.println("s1 = " + s1);
    System.out.println("s2 = " + s2);
    System.out.println("sb1 = " + sb);
  }
}
```

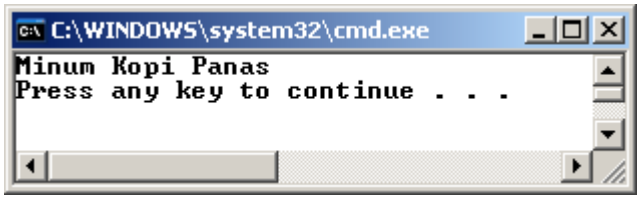
Output Contoh Ketiga :



2.4. Contoh Keempat

```
public class Insert {
  public static void main(String argv[]) {
    StringBuffer str = new StringBuffer("Minum Panas");
    str.insert(6,"Kopi ");
    System.out.println(str);
  }
}
```

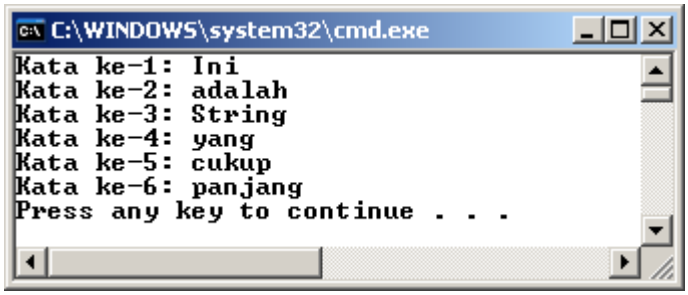
Output Contoh Keempat :



2.5. Contoh Kelima

```
import java.util.*;
public class StrToken {
  public static void main(String argv[]) {
    String str = "Ini adalah String yang cukup panjang";
    StringTokenizer st = new StringTokenizer(str);
    int i = 1;
    while (st.hasMoreTokens()) {
      System.out.println("Kata ke-" + i + ": " + st.nextToken());
      i++;
    }
  }
}
```

Output Contoh Kelima :



3. Kelas Math dan Kelas StrictMath

Kelas Math berisi sekumpulan method dan konstanta matematika.

Kelas ini bersifat *final* (tidak dapat diturunkan, dan tidak dapat melakukan instansiasi), dan semua attribute dan methodnya bersifat *static*.

Java2 mulai versi1.3 menambah kelas StrictMath yang berisi sekumpulan method matematika yang lengkap, serupa dengan kelas Math. Perbedaannya adalah StricMath dijamin menghasilkan hasil identik untuk diimplementasikan di Java manapun.

Method-method di kelas Math :

```
static double toRadians(double sudut);
static double toDegrees(double sudut);
static double sin(double d);
static double cos(double d);
```

static double tan(double d);
static double asin(double d);
static double acos(double d);
static double atan(double d);
static double exp(double d); // e pangkat d
static double log(double d); // ln(d);
static double sqrt(double d); // $\sqrt{d}$
static double pow(double a, double b); // a pangkat b
static double ceil(double d); // pembulatan ke atas
static double floor(double d); //pembulatan ke bawah
static int round(float f); // pembulatan biasa
static long round(double d); // pembulatan biasa
static double rint(double d); // pembulatan ke int terdekat
static double atan2(double a, double b); //cartesius ke polar
static synchronized double random();

Lanjutan method-method di kelas Math :

static int abs(int i);
static long abs(long l);
static float abs(float f);
static double abs(double d);
static int min(int a, int b);
static long min(long a, long b);
static float min(float a, float b);
static double min(double a, double b);
static int max(int a, int b);
static long max(long a, long b);
static float max(float a, float b);
static double max(double a, double b);
Math.E=2.7…; Math.PI=3.14…;

Penggunaan method-method pada kelas Math :

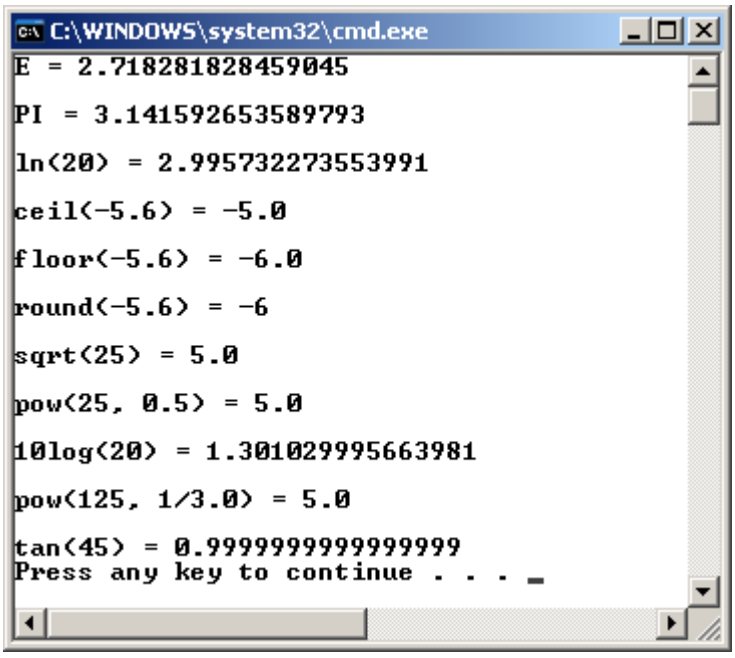
double d = Math.sin(Math.PI/2);
double d1 = 12.3;
double d2 = Math.exp(d1);
double d3 = Math.log(d1);
double d4 = Math.sqrt(d1);
double d5 = Math.pow(d1, 3.0);
double d6 = Math.ceil(7.3); // hasilnya 8
double d7 = Math.ceil(-7.3); // hasilnya -7
double d8 = Math.floor(7.3); // hasilnya 7
double d9 = Math.floor(-7.3); // hasilnya -8

```
double d10 = 37.125;
int i = Math.round((float) d10);
long l = Math.round(d10);
double d11 = 14.2, d12 = 18.5;
double d13 = Math.min(d11, d12);
double d14 = Math.max(d11, d12);
static double random(); // 0.0 <=hasil<=1.0
```

3.1. Contoh Pertama

```
import java.io.*;
public class OperasiMtk {
  public static void main(String[] args) {
    System.out.println("E = " + Math.E);
    System.out.println();
    System.out.println("PI = " + Math.PI);
    System.out.println();
    System.out.println("ln(20) = " + Math.log(20));
    System.out.println();
    System.out.println("ceil(-5.6) = " + Math.ceil(-5.6));
    System.out.println();
    System.out.println("floor(-5.6) = " + Math.floor(-5.6));
    System.out.println();
    System.out.println("round(-5.6) = " + Math.round(-5.6));
    System.out.println();
    System.out.println("sqrt(25) = " + Math.sqrt(25));
    System.out.println();
    System.out.println("pow(25, 0.5) = " + Math.pow(25, 0.5));
    System.out.println();
    System.out.println("10log(20) ="+Math.log(20)/Math.log(10)); // basis 10
    System.out.println();
    System.out.println("pow(125, 1/3.0) = " + Math.pow(125, 1/3.0));
    System.out.println();
    double radians = Math.toRadians(45);
    System.out.println("tan(45) = " + Math.tan(radians));
  }
}
```

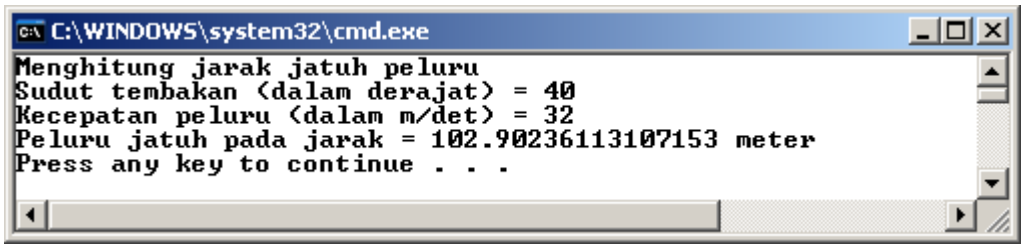
Output Contoh Pertama :



3.2. Contoh Kedua

```
import java.io.*;
public class Lintasan {
  public static void main(String[] args) throws IOException {

    BufferedReader dataBaris = new BufferedReader(new
                                        InputStreamReader(System.in));

    System.out.println("Menghitung jarak jatuh peluru ");
    System.out.print("Sudut tembakan (dalam derajat) = ");
    String str1 = dataBaris.readLine();
    double derajat = Double.valueOf(str1).intValue();
    double radian = Math.toRadians(derajat);
    //double radian = derajat*Math.PI/180;
    System.out.print("Kecepatan peluru (dalam m/det) = ");
    String str2 = dataBaris.readLine();
    double kec = Double.valueOf(str2).doubleValue();
    double jarak = 2*Math.pow(kec, 2)
                          *Math.sin(radian)*Math.cos(radian)/9.8;

    System.out.println("Peluru jatuh pada jarak = " + jarak + " meter");
  }
}
```

Output Contoh Kedua:



4. Kelas-Kelas Tipe Data Wrapper

Sembilan tipe data dasar (*boolean, byte, char, double, float, int, long, short, void*) di java tidak diimplementasikan sebagai kelas. Kelas wrapper bertindak sebagai versi kelas dari tipe data dasar, yang namanya serupa namun diawali huruf kapital.

Jadi kelas tipe data wrapper di java5 ada sebelas yaitu *Boolean, Byte, Character, Double, Float, Integer, Long, Number, Short, Void, dan Dimension.*

Penting untuk diperhatikan bahwa tipe data wrapper dan tipe data dasar **tidak saling menggantikan**.

Tipe data dasar diperoleh dari tipe data wrapper dengan cara memanggil method di tipe data wrapper.

Tipe data dasar dilewatkan ke method dengan *pass by value*, jadi jika ada method membutuhkan proses pengiriman argument dengan *pass by reference* harus memanfaatkan kelas tipe data wrapper, juga tentunya kelas String atau StringBuffer.

Kelas-kelas tipe data wrapper menyediakan versi objek dari tipe data dasar, maka dimungkinkan memanfaatkan method-method yang tersedia untuk masing-masing tipe data.

Ada enam subkelas kongkret yang menyimpan nilai-nilai secara eksplisit masing-masing tipe data dasar yaitu double, float, byte, short, integer, dan long.

4.1. Kelas Number

Kelas ini bersifat abstract serta mendefinisikan super kelas yang diimplementasikan oleh kelas-kelas yang membungkus tipe data dasar.

Method-method dari kelas Number :

byte byteValue();

double doubleValue();

float floatValue();

int intValue();

long longValue();

short shortValue();

4.2. Kelas Boolean

Kelas ini membungkus tipe data dasar Boolean, dan memiliki dua constructor :

Boolean(Boolean boolValue);

Boolean(String boolString);

Method-method di kelas ini adalah :

boolean booleanValue();

boolean equels(Object boolObj);

static boolean getBoolean(String propertyName);

int hashCode();

String toString();

Static boolean valueOf(String boolString);

Dua konstanta pada kelas ini yaitu Boolean.*TRUE* dan Boolean.*FALSE*.

4.3. Kelas Character

Kelas ini membungkus tipe data dasar dan memiliki beberapa method :

static int digit(char ch, int radix);

Contoh :

```
char ch1 = '4';
char ch2 = 'c';
int four = Character.digit(ch1, 10); // hasilnya bil 4
int twelve = Character.digit(ch2, 12); // hasilnya hexa c
static char forDigit(int digit, int radix);
```

Contoh :
int i = 9;
char c = Character.forDigit(i, 10); //hasilnya karakter '9'
static boolean isDefined(char ch);
static boolean isDigit(char ch);

Contoh :

boolean isDigit = Character.isDigit('7'); // hasilnya true
static boolean isIdentifierIgnorable(char ch);
static boolean isJavaIdentifierPart(char ch);
static boolean isLetter(char ch);
static boolean isLetterOrDigit(char ch);
static boolean isLowerCase(char ch);

Contoh :

Character c = new Character('g');
boolean isLower = Character.isLowerCase(c); // true
static boolean isSpace(char ch);

Contoh :

boolean isSpace = Character.isSpace('\t');//hasilnya true
static boolean isUpperCase(char ch);
static boolean isTitleCase(char ch);
static boolean isUnicodeIdentifierPart(char ch);
static boolean isUnicodeIdentifierStart(char ch);
static boolean isWhitespace(char ch);
static char toLowerCase(char ch);

Contoh :

char ch = Character.toLowerCase('M');
System.out.println(ch); // hasilnya m
static char toTitleCase(char ch);
static char toUpperCase(char ch);

Contoh :

char ch = Character.toUpperCase('n');
System.out.println(ch); // hasilnya N

Saat konversi dari numerik ke karakter atau sebaiknya, kelas Character memiliki attribute yang bersifat static final (konstanta), yaitu MIN_RADIX (basis 2) dan MAX_RADIX (basis 36).

4.4. Kelas Byte, Short, Integer, dan Long

Method-method di kelas Byte adalah :

byte byteValue(); // objek kelas Byte menjadi nilai byte
// membandingkan dua objek Byte secara numerik
int compareTo(Byte b);
int compareTo(Object obj);
static Byte decode(String str) throws NumberFormatException
doubleValue(); // objek kelas Byte menjadi nilai double
// membandingkan dua objek milik kelas Byte
boolean equels(Object obj);
float floatValue();// objek kelas Byte menjadi nilai float
int hashCode();
int intValue();// objek kelas Byte menjadi nilai int
long longValue();// objek kelas Byte menjadi nilai long
static byte parseByte(String str) throws NumberFormatException
static byte parseByte(String str, int radix) throws NumberFormatException
short shortValue();//objek kelas Byte menjadi nilai short
String toString();
// konversi nilai numerik menjadi objek milik kelas String
static String toString(byte num);
static Byte valueOf(String str) throws NumberFormatException
static Byte valueOf(String str, int radix) throws NumberFormatException

Method-method di kelas Short :

byte byteValue(); // objek kelas Short menjadi nilai byte
// membandingkan dua objek milik kelas Short secara numerik
int compareTo(Short sh);
int compareTo(Object obj);
static Short decode(String str) throws NumberFormatException
//konversi objek kelas Byte menjadi nilai double
daouble doubleValue();
// membandingkan dua objek milik kelas Short
boolean equels(Object shortObj);
float floatValue(); // objek kelas Short menjadi nilai float
int hashCode();
int intValue(); // objek kelas Short menjadi nilai int
long longValue(); // objek kelas Short menjadi nilai long
static byte parseShort(String str) throws NumberFormatException

static byte parseShort(String str, int radix) throws NumberFormatException
short shortValue(); //objek kelas Short menjadi nilai short
// konversi nilai numerik short menjadi objek milik kelas String
String toString(short num);
static Short valueOf(String str) throws NumberFormatException
static Short valueOf(String str, int radix) throws NumberFormatException

Method-method di kelas Integer :

byte byteValue(); //objek kelas Integer menjadi nilai byte
// membandingkan dua objek milik kelas Integer secara numerik
int compareTo(Integer i);
int compareTo(Object obj);
static Integer decode(String str) throws NumberFormatException
//objek milik kelas Integer menjadi nilai double
double doubleValue();
// membandingkan dua objek milik kelas Integer
boolean equels(Object IntegerObj);
//konversi objek milik kelas Integer menjadi nilai float
float floatValue();
static Integer getInteger(String propertyName);
static Integer getInteger(String propertyName, int default);
static Integer getInteger(String propertyName, Integer default);
static String toBinaryString(int num);
static String toHexString(int num);
static String toOctalString(int num);
int hashCode();
int intValue();//objek milik kelas Integer menjadi nilai int
// konversi objek milik kelas Integer menjadi nilai long
long longValue();
static byte parseInt(String str) throws NumberFormatException
static byte parseInt(String str, int radix) throws NumberFormatException
// konversi objek milik kelas Integer menjadi nilai short
short short()Value;
// konversi nilai numerik short menjadi objek milik kelas String
String toString();
// konversi nilai numerik int menjadi objek milik kelas String
static String toString(int num);
static String toString(int num, int radix);
static Integer valueOf(String str) throws NumberFormatException
static Integer valueOf(String str, int radix) throws NumberFormatException

Dua attribute yang bersifat final static (konstanta) pada kelas Integer :

MINVALUE (nilai terkecil yg dapat disajikan objek milik kelas Integer).

MAXVALUE (nilai terbesar yg dapat disajikan objek milik kelas Integer).

Method-method di kelas Long adalah :

byte byteValue(); // objek kelas Long menjadi nilai byte
// membandingkan dua objek milik kelas Long secara numerik
int compareTo(Long obj);
int compareTo(Object obj);
static Long decode(String str) throws NumberFormatException
//objek milik kelas Long menjadi nilai double
double doubleValue();
// membandingkan dua objek milik kelas Long
boolean equels(Object obj);
//konversi objek milik kelas Long menjadi nilai float
float floatValue();

contoh :

Integer i = new Integer(17);
float f = i.floatValue(); // hasilnya 17.0
static Long getLong(String propertyName);
static Long getLong(String propertyName, int default);
static Long getLong(String propertyName, Integer default);
int hashCode();
int intValue(); // objek milik kelas Long menjadi nilai int
// konversi objek milik kelas Long menjadi nilai long
long longValue();
static long parseLong(String str) throws NumberFormat Exception
static long parseLong(String str, int radix) throws NumberFormat Exception
// konversi objek milik kelas Long menjadi nilai short
short shortValue();
static String toBinaryString(long num);
static String toHexString(long num);
static String toOctalString(long num);
// konversi nilai numerik long menjadi objek milik kelas String
String toString();
Static String toString(long num);
static Long valueOf(String str) throws NumberFormatException
static Long valueOf(String str, int radix) throws NumberFormatException

4.5. Kelas Float dan Double

Method-method di kelas Float :

byte byteValue(); // objek kelas Float menjadi nilai byte
// membandingkan dua objek milik kelas Float secara numerik
int compareTo(Float f);

int compareTo(Object obj);
//objek milik kelas Float menjadi nilai double
double doubleValue();
// membandingkan dua objek milik kelas Float
boolean equals(Object obj);
// konversi objek milik kelas Float menjadi nilai float
float floatValue();
int hashCode();
static float intBitsToFloat(int bits); // bit biner ke float
static int floatToIntBits(float num); //bil float ke bit biner
int intValue(); // objek milik kelas Float menjadi nilai int
boolean isInfinite();
static boolean isInfinite(float f);
boolean isNaN();
static boolean isNaN(float f);
// konversi objek milik kelas Float menjadi nilai long
long longValue();

contoh :

Float f = new Float(34.237);
long l = f.longValue(); // hasilnya long 34
static float parseFloat(String str) throws NumberFormatException
// konversi objek milik kelas Float menjadi nilai short
short shortValue();
// konversi nilai numerik float menjadi objek milik kelas String
String toString();
static String toString(float num);
static Float valueOf(String str) throws NumberFormatException
Method-method di kelas Double :
// objek kelas Double menjadi nilai byte
byte byteValue();
// membandingkan dua objek milik kelas Double secara numerik
int compareTo(Double d);
int compareTo(Object obj);
static int doubleToLongBits(double num);
//objek milik kelas Double menjadi nilai double
double doubleValue();
// membandingkan dua objek milik kelas Double
boolean equals(Object obj);
// konversi objek milik kelas Double menjadi nilai float
float floatValue();
int hashCode();
int intValue(); // objek milik kelas Double menjadi nilai int
boolean isInfinite();

static boolean isInfinite(double num); //num tak hingga
boolean isNaN();
static boolean isNaN(double num); //NaN = not a number
staic double longBitsToDouble(long num);
// konversi objek milik kelas Double menjadi nilai long
long longValue();
static float parseDouble(String str) throws NumberFormatException
// konversi objek milik kelas Double menjadi nilai short
short shortValue();
// konversi nilai numerik double menjadi objek milik kelas String
String toString()
static String toString(double num);

Kelas Float memiliki beberapa attribut yang bersifat final dan static (konstanta) yaitu sebagai berikut :

Nama Konstanta	Deskripsi
MINVALUE	Nilai terkecil objek Float
MAXVALUE	Nilai terbesar objek Float
NEGATIVE_INFINITY	Angka tak berhingga negative
POSITIVE_INFINITY	Angka tak berhingga positif
NaN	Kondisi bukan angka

4.6. Kelas System

Kelas ini mendukung penggunaan method untuk memperoleh properti sistem. Berikut tiga method yang sering digunakan dalam kelas ini :
getProperties() //menghasilkan objek Properties yang berisi properti sistem.
getProperty(String kunci) // menghasilkan string yang menyatakan nilai property
getenv(String nama) // menghasilkan string yang menyatakan nilai var lingkungan

4.6.1. Contoh Program

```
import java.util.Properties;
import java.util.Enumeration;
```

```
public class PropertiSistem {
  public static void main(String[] args) {
    Properties p = System.getProperties();

    Enumeration elemen = p.propertyNames();
    while (elemen.hasMoreElements()) {
      String namaProperti = (String) elemen.nextElement();
      System.out.println(namaProperti + " = " + p.getProperty(namaProperti));
    }
  }
}
```

Outputnya :

```
java.runtime.name = Java(TM) 2 Runtime Environment, Standard Edition
sun.boot.library.path = C:\Program Files\Java\jdk1.5.0_01\jre\bin
java.vm.version = 1.5.0_01-b08
java.vm.vendor = Sun Microsystems Inc.
java.vendor.url = http://java.sun.com/
path.separator = ;
java.vm.name = Java HotSpot(TM) Client VM
file.encoding.pkg = sun.io
user.country = US
sun.os.patch.level = Service Pack 2
java.vm.specification.name = Java Virtual Machine Specification
user.dir = D:\BahanKuliah\bahan ajar PBO
java.runtime.version = 1.5.0_01-b08
java.awt.graphicsenv = sun.awt.Win32GraphicsEnvironment
java.endorsed.dirs = C:\Program Files\Java\jdk1.5.0_01\jre\lib\endorsed
os.arch = x86
java.io.tmpdir = C:\DOCUME~1\COMPAQ\LOCALS~1\Temp\
line.separator =

java.vm.specification.vendor = Sun Microsystems Inc.
user.variant =
os.name = Windows XP
sun.jnu.encoding = Cp1252
java.library.path = C:\Program Files\Java\jdk1.5.0_01\bin;.;C:\WINDOWS\system32;
C:\WINDOWS;C:\PROGRA~1\RATIONAL\RATION~1\NUTCROOT\bin;C:\PROGRA~1\RATIONAL\RATIO
N~1\NUTCROOT\bin\x11;C:\PROGRA~1\RATIONAL\RATION~1\NUTCROOT\mksnt;C:\WINDOWS\sys
tem32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program Files\Rational\common;C:\Pr
ogram Files\Rational\ClearQuest;C:\Program Files\Rational\Rose\TopLink\;C:\Progr
am Files\Rational\Rational Test;C:\Program Files\Microsoft SQL Server\90\Tools\b
inn\
java.specification.name = Java Platform API Specification
java.class.version = 49.0
sun.management.compiler = HotSpot Client Compiler
os.version = 5.1
user.home = C:\Documents and Settings\COMPAQ
user.timezone =
java.awt.printerjob = sun.awt.windows.WPrinterJob
file.encoding = Cp1252
java.specification.version = 1.5
user.name = COMPAQ
java.class.path = .
java.vm.specification.version = 1.0
sun.arch.data.model = 32
java.home = C:\Program Files\Java\jdk1.5.0_01\jre
java.specification.vendor = Sun Microsystems Inc.
user.language = en
awt.toolkit = sun.awt.windows.WToolkit
java.vm.info = mixed mode, sharing
java.version = 1.5.0_01
java.ext.dirs = C:\Program Files\Java\jdk1.5.0_01\jre\lib\ext
sun.boot.class.path = C:\Program Files\Java\jdk1.5.0_01\jre\lib\rt.jar;C:\Progra
m Files\Java\jdk1.5.0_01\jre\lib\i18n.jar;C:\Program Files\Java\jdk1.5.0_01\jre\
lib\sunrsasign.jar;C:\Program Files\Java\jdk1.5.0_01\jre\lib\jsse.jar;C:\Program
 Files\Java\jdk1.5.0_01\jre\lib\jce.jar;C:\Program Files\Java\jdk1.5.0_01\jre\li
b\charsets.jar;C:\Program Files\Java\jdk1.5.0_01\jre\classes
java.vendor = Sun Microsystems Inc.
file.separator = \
java.vendor.url.bug = http://java.sun.com/cgi-bin/bugreport.cgi
sun.cpu.endian = little
sun.io.unicode.encoding = UnicodeLittle
```

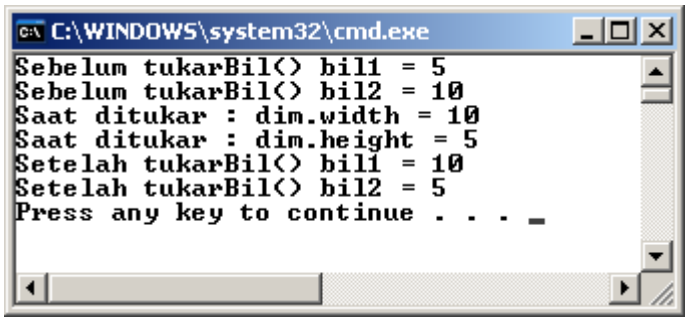
4.7. Kelas Dimension

Kelas ini memiliki dua attribute standard yaitu width dan height, sehingga cocok digunakan untuk mengolah data yang berpasangan seperti titik dua dimensi, bangunan dua dimensi, dan frame.

Contoh penggunaan kelas Dimension :

```
import java.awt.Dimension;

public class PassByReference {
  static void tukarBil(Dimension dim) {
    Integer temp;
    temp = dim.width;
    dim.width = dim.height;
    dim.height = temp;
    System.out.println("Saat ditukar : dim.width = "+ dim.width);
    System.out.println("Saat ditukar : dim.height = "+ dim.height);
  }

  public static void main(String[] args) {
        Dimension bil = new Dimension(5,10);
        PassByReference pbr = new PassByReference();
        System.out.println("Sebelum tukarBil() bil1 = " + bil.width);
        System.out.println("Sebelum tukarBil() bil2 = " + bil.height);
        pbr.tukarBil(bil);
        System.out.println("Setelah tukarBil() bil1 = " + bil.width);
        System.out.println("Setelah tukarBil() bil2 = " + bil.height);
  }
}
```

Outputnya :



D. Rangkuman

Bahasa Java diciptakan untuk memenuhi kebutuhan sistem. Beberapa konsep inti bahasa di implementasikan sebagai kelas Java, misalnya manipulasi String, multi jalinan, dan penanganan excepsi. Java memuiliki beberapa kelas dasar diantarnya adalah kelas String, kelas String Buffer, kelas math, kelas character, kelas number, dan masih banyak lagi lainya.

Latihan Soal

1. Buatlah sebuah program untuk menghitung akar-akar imajiner dengan menggunakan rumus ABC
2. Buatlah sebuah program aplikasi kalkulator sederhana
3. Buatlah program yang menerima input dari keyboard dan kemudian memprosesnya sebagai Command. Program selesai bila Command exit diinputkan.

BAB IX
MULTI THREADING

9.1. Pengertian Thread

Thread adalah objek yang mewakili satu unit eksekusi sekumpulan instruksi. Ada pula yang mendefinisikan, thread adalah sekumpulan instruksi yang dapat dieksekusi secara mandiri. Proses adalah satu unit kepemilikan sumberdaya.

Multithreading adalah suatu kemampuan yang memungkinkan beberapa thread dapat dijalankan secara bersamaan/bergantian/konkuren.

Manfaat aplikasi menggunakan multithreading yaitu agar thread-thread di suatu proses, dapat sharing kode program, data, dan sumber daya, secara lebih efisien dibanding proses-proses terpisah, sehingga kinerja aplikasi berbasis thread lebih baik dibanding aplikasi berbasis proses.

9.2. Membuat Thread

Untuk membuat thread ada dua cara yaitu:

1) Membuat kelas implementasi dari interface Runnable

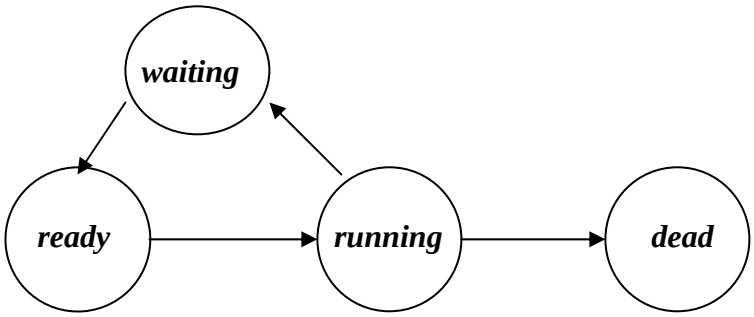
```
public class NamaThread1 implements Runnable {
  ……..
  public void run() {
    ……..
   }
  ……..
}
```

2) Membuat kelas turunan dari kelas thread

```
public class NamaThread2 extends Thread{
  ……..
  public void run() {
    ……..
   }
  ……..
}
```

3. Kondisi Thread

Thread dapat berada dalam 1 dari 4 kondisi berikut:



Multitasking adalah kemampuan komputer melakukan banyak tugas secara konkuren/bergantian. Pada multitasking, konkurensi melibatkan banyak proses. Karena multithreading perluasan dari multitasking, maka pada multithreading setiap proses melibatkan banyak thread.

Terdapat tiga kelas yang berkaitan dengan aplikasi multithreading :

1. Thread : membuat thread di program
2. ThreadDeath : membersihkan thread yang telah berakhir eksekusinya.
3. ThreadGroup : mengelompokkan thread-thread

Hanya ada satu interface yaitu Runnable yang digunakan untuk membuat thread.

Method-method standard yang berkaitan dengan multithreading :

a. start() → dipanggil setiap kali thread dimulai.
b. init() → gunanya sama dengan start() untuk menempatkan program inisialisasi
c. run() → berisi badan Thread
d. stop() → untuk menghentikan thread dan ditempatkan di akhir badan thread
e. Method-method standard lain adalah *wait(), notify(), notifyAll(), sleep()*, *isAlive()* dan *interrupt().*

4. Siklus Hidup Thread

a. Pertama kali thread diciptakan (**born**), selalu dalam kondisi **ready**. Method *start()* harus dipanggil untuk mengeksekusi method *run()* sehingga thread tersebut siap segera dieksekusi processor.
b. Thread yang berada pada kondisi **ready** antri menunggu giliran untuk dieksekusi, penjadwal akan melakukan dispatching kepada thread sehingga thread transisi dari kondisi **ready** ke **running** (processor sedang mengeksekusi thread tersebut).

c. Jika saat eksekusi thread memerlukan kondisi tertentu untuk dipenuhi atau karena diinterupsi, maka thread tersebut dapat memanggil method wait() atau sleep() agar thread dapat transisi dari kondisi **running** ke kondisi **waiting** (menunggu sampai masuk kembali ke kondisi ready).

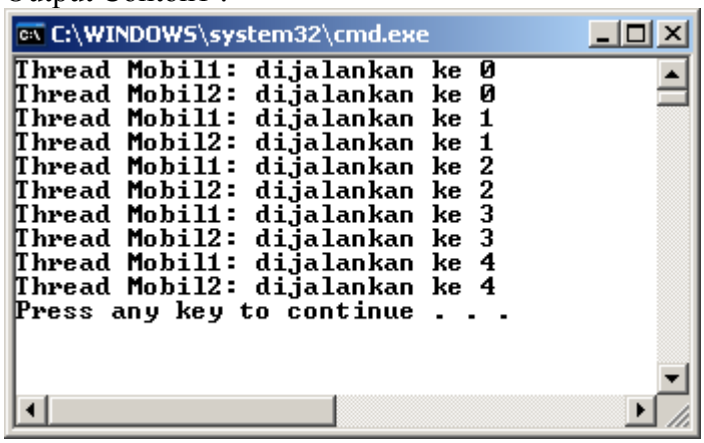
d. Thread transisi dari kondisi **running** ke kondisi **dead** ketika method run() selesai dilaksanakan atau diakhiri karena terjadi exception, selanjutnya dihapus dari memori aktif.

5. Contoh-Contoh Program

5.1. Contoh1

```
class Mobil extends Thread {
  public Mobil(String id) {
    super(id); // id adalah nama thread
  }
  public void run() {
    String nama = getName();
    for (int i=0; i<5; i++) {
      try {
        Thread.currentThread().sleep(1000); // tunggu 1000 milli detik
      }
      catch (InterruptedException ie) {
        System.out.println("Terinterupsi");
      }
      System.out.println("Thread " + nama + ": dijalankan ke " + i);
    }
  }
}

public class AplyThreadMobil {
  public static void main(String[] args) {
    Mobil mbl1 = new Mobil("Mobil1");  // membuat objek Thread1, dinamai Mobil1
    Mobil mbl2 = new Mobil("Mobil2");  // membuat objek Thread2, dinamai Mobil2
    mbl1.start();  // eksekusi thread mbl1, dengan cara mengeksekusi method run()
    mbl2.start();  // eksekusi thread mbl2, dengan cara mengeksekusi method run()
  }
}
```

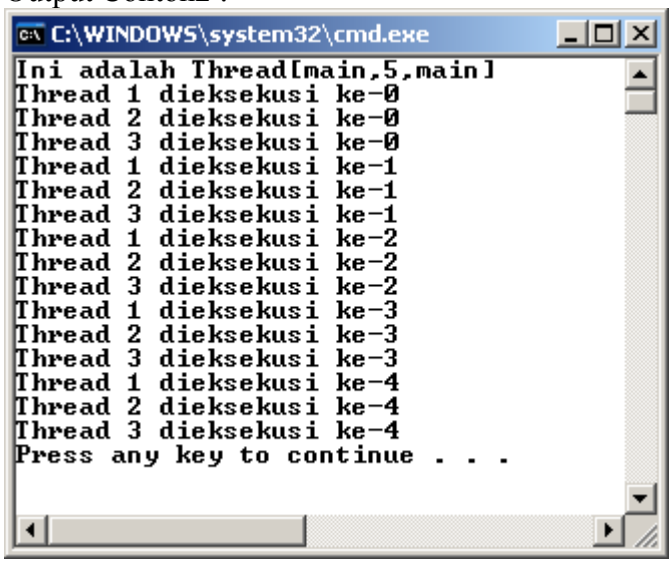
Output Contoh1 :



5.2. Contoh2

```
// membuat Thread dengan cara 2
public class MyOtherThread implements Runnable {
  private Thread t;

  public MyOtherThread(String nama) {  // start ada di constructor
    t = new Thread(this, nama);
    t.start();
  }

  public void run() {
    try {
      for (int i = 0; i < 5; i++) {
        System.out.println(t.getName()+" dieksekusi ke-" + i);
        t.sleep(1);
      }
    }
    catch (InterruptedException ex) {
    }
  }
}
public class AplyMyOtherThread {
  public static void main(String argv[]) {
    MyOtherThread t1 = new MyOtherThread("Thread 1");
    MyOtherThread t2 = new MyOtherThread("Thread 2");
    MyOtherThread t3 = new MyOtherThread("Thread 3");

    System.out.println("Ini adalah " + Thread.currentThread());
  }
}
```

Output Contoh2 :

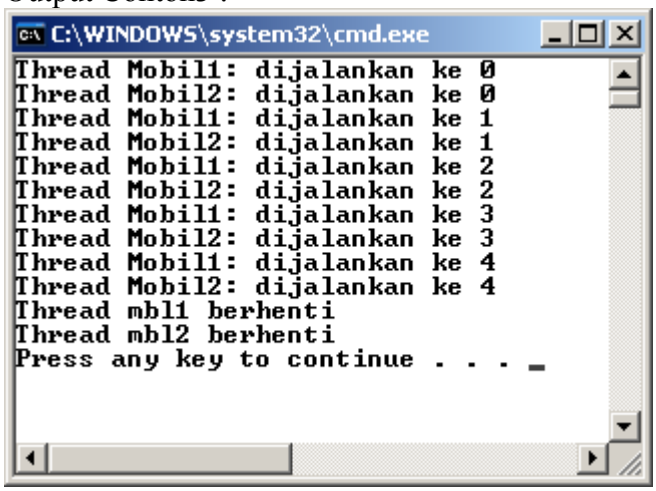


5.3. Contoh3

Kelas program berikut adalah melengkapi kelas AplyThreadMobil agar mengecek apakah kedua thread (mbl1 dan mbl2) sudah berhenti dijalankan.

```
public class AplyThreadMobil2 {
  public static void main(String[] args) {
    Mobil mbl1 = new Mobil("Mobil1");  // buat objek Thread1 dan dinamai Mobil1
    Mobil mbl2 = new Mobil("Mobil2");  // buat objek Thread2 dan dinamai Mobil2
    mbl1.start();  // eksekusi thread mbl1, dengan cara mengeksekusi method run()
    mbl2.start();  // eksekusi thread mbl2, dengan cara mengeksekusi method run()
    boolean mbl1berhenti = false;  // belum berhenti
    boolean mbl2berhenti = false;  // belum berhenti
    do {
          // cek kondisi thread mbl1
          if (!mbl1.isAlive()) {
            mbl1berhenti = true;
            System.out.println("Thread mbl1 berhenti");
          }
          // cek kondisi thread mbl2
          if (!mbl2.isAlive()) {
            mbl2berhenti = true;
            System.out.println("Thread mbl2 berhenti");
          }
    } while (!mbl1berhenti || !mbl2berhenti);
  }
}
```

Output Contoh3 :



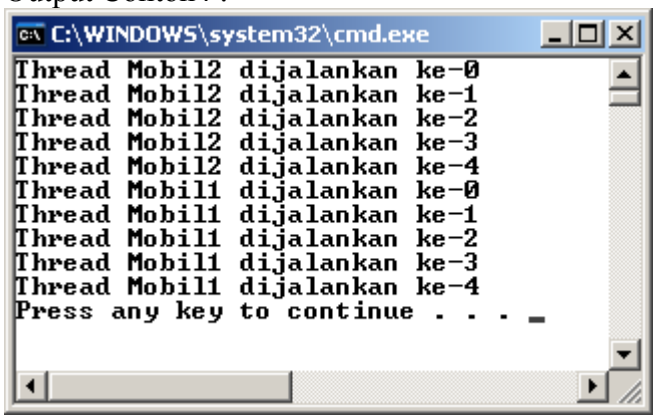
Sinkronisasi merupakan suatu upaya agar kode program tertentu dijalankan secara sekuensial sehingga kode tersebut tidak akan dijalankan oleh thread lain dalam waktu yang bersamaan. Jika sinkronisasi diterapkan dalam method, dipastikan bahwa seluruh kode di dalam method tersebut dieksekusi tanpa diinterupsi oleh yang lain, perhatikan contoh berikut ini :

5.4. Contoh4

```
class SinkronisasiKeluaran {
  public static synchronized void info(String nama) {
    for (int i=0; i<5; i++) {
      try {
        Thread.sleep(1000); // non aktif 1000 milli detik
      }
      catch(InterruptedException ie) {
        System.out.println("Terinterupsi");
      }
      System.out.println("Thread " + nama + " dijalankan ke-"+i);
    }
  }
}
class Mobil2 extends Thread {
  public Mobil2(String id) {
    super(id);
  }
  public void run() {
    String nama = getName();
```

```
    SinkronisasiKeluaran.info(nama);
  }
}
public class AplySinkron {
  public static void main(String[] args) {
    Mobil2 mbl1 = new Mobil2("Mobil1");
    Mobil2 mbl2 = new Mobil2("Mobil2");
    mbl1.start();
    mbl2.start();
  }
}
```

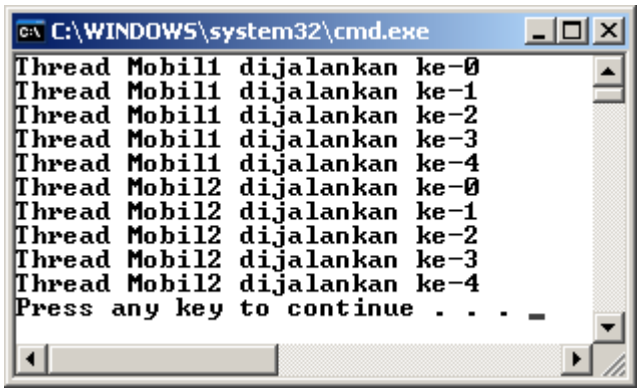
Output Contoh4 :



Terlihat bahwa thread Mobil2 dijalankan duluan adalah kebetulan, suatu saat yang dijalankan bisa Mobil1 duluan, untuk memastikan yang dijalankan duluan secara sekuensial adalah Mobil1, maka sebelum method start() harus dipanggil terlebih dahulu method setPriority(), sehingga kelas AplySinkron menjadi :

```
public class AplySinkron {
  public static void main(String[] args) {
    Mobil2 mbl1 = new Mobil2("Mobil1");
    Mobil2 mbl2 = new Mobil2("Mobil2");
    mbl1.setPriority(6);
    mbl1.start();
    mbl2.start();
  }
}
```

Outputnya menjadi :

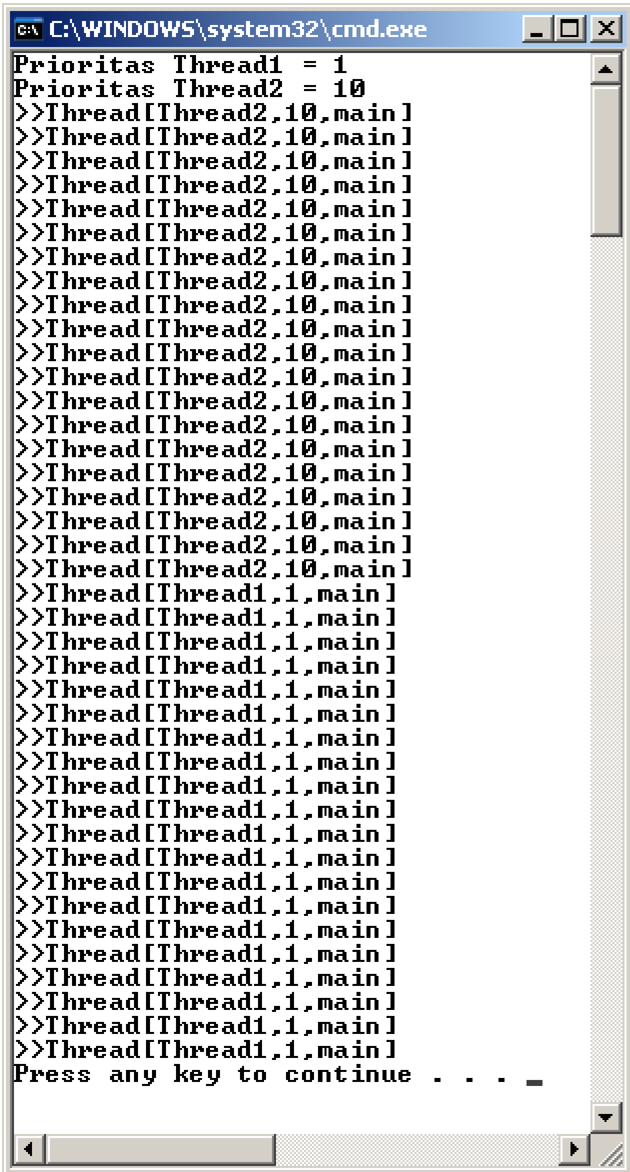


5.5. Contoh5

Contoh berikut adalah variasi lain dari pemanfaatan method setPriority() :

```
public class MultiThread2 extends Thread {
  public MultiThread2(String s) {
    this.setName(s);
  }

  public MultiThread2(String s, int pri) {
    this.setName(s);
    this.setPriority(pri);
  }

  public void run() {
    for(int i = 0; i < 20; i++) {
      System.out.println(">>" + Thread.currentThread());
    }
  }
  public static void main(String argv[]) {
    MultiThread2 t1 = new MultiThread2("Thread1", Thread.MIN_PRIORITY);
    MultiThread2 t2 = new MultiThread2("Thread2", Thread.MAX_PRIORITY);
    System.out.println("Prioritas Thread1 = " + t1.getPriority());
    System.out.println("Prioritas Thread2 = " + t2.getPriority());
    t1.start();
    t2.start();
  }
}
```

Output Contoh5 :



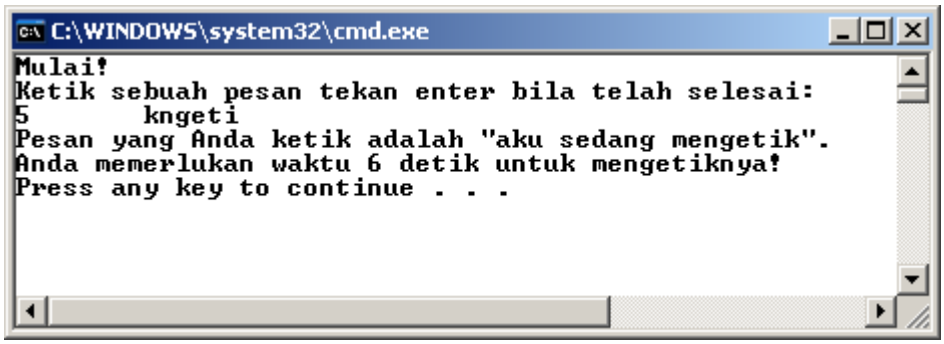
```
C:\WINDOWS\system32\cmd.exe
Prioritas Thread1 = 1
Prioritas Thread2 = 10
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread2,10,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
>>Thread[Thread1,1,main]
Press any key to continue . . .
```

Contoh6 :

```
import java.io.*;
public class CepatKetik {
  private static boolean siap = false;
  private static String pesan;
```

```
  public static void main(String argv[]) {
    int detik = 0;
    Thread t = new Thread() {
      public void run() {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader buf = new BufferedReader(isr);
        System.out.println("Ketik sebuah pesan tekan enter bila telah selesai:");
        try {
          pesan = buf.readLine();
        }
        catch (IOException ex) {
          System.err.println(ex);
        }
        siap = true;
      }
    };
    System.out.println("Mulai!");
    t.start();
    while (!siap) {
      try {
        Thread.sleep(1000);
      }
      catch (InterruptedException ex) {
        System.err.println("Tidurku terganggu!");
      }
      System.out.print("\r" + (++detik) + "\t");
    }
    System.out.println("\rPesan yang Anda ketik adalah \"" + pesan +"\".");
    System.out.println("Anda memerlukan waktu " + detik + " detik untuk mengetiknya!");
  }
}
```

Output contoh6 :



Program contoh6 ini menuunggu user untuk mengetikkan suatu string yang dihitung kecepatan dalam mengetiknya sampai ditekan tombol enter.

SOAL LATIHAN

1. Apa yang dimaksud dengan Thread, Proses dan Multithreading?
2. Sebutkan dan jelaskan kelas-kelas dan interface yang ada dalam multithreading!
3. Buatkan kelas turunan dari kelas thread!