MODUL PRAKTIKUM

VISUALISASI DALAM FISIKA (SFI-131)

Denny Darmawan, M.Sc.
(darmawan@uny.ac.id)

PROGRAM STUDI FISIKA
FAKULTAS MATEMATIKA & ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI YOGYAKARTA

MODUL 1: PENGANTAR PEMROGRAMAN JAVA

Pendahuluan

Pemrograman dengan Java saat ini cukup populer terutama dalam hal mem-visualisasi-kan sebuah konsep fisika. Banyak applet Java yang tersedia bebas di internet yang secara khusus dibuat untuk membantu memahami konsep-konsep fisika secara visual.

Salah satu keunggulan dari Java adalah bahwa kode yang kita tuliskan dapat dijalankan di banyak sistem operasi bahkan di platform *mobile* seperti *handphone*, sehingga dikenal dengan semboyan "*write once, run everywhere*". Hal ini tentu akan mempermudah dalam membangun sebuah aplikasi karena programmer tidak perlu lagi memikirkan sistem operasi apa yang nantinya dipakai untuk menjalankan aplikasi yang dibangunnya.

Keunggulan lainnya adalah, sintaks Java mirip dengan bahasa pemrograman C atau C++ yang merupakan salah satu bahasa pemrograman yang cukup populer dikalangan programmer, sehingga apabila sudah menguasai bahasa C atau C++, mempelajari Java akan terasa lebih mudah.

Sekilas Java

Pada awalnya, Java merupakan hasil proyek penelitian yang dilakukan oleh perusahaan Sun Microsystem dengan kode nama Green pada tahun 1991. Target yang diharapkan dari penelitian ini adalah terciptanya bahasa pemrograman yang dapat berjalan di semua platform. James Gosling yang merupakan kontributor dalam proyek ini menamakan bahasa pemrograman baru tadi Oak. Namun, ternyata nama ini sudah dipakai oleh bahasa pemrograman lain. Akhirnya oleh staff Sun disepakai bahwa nama bahasa pemrograman yang berhasil diciptakan tadi adalah Java. Pada tahun 1995, Sun secara resmi mengumumkan kehadiran bahasa pemrograman baru ini sebagai Java versi 1.0.

Java berkembang menjadi Java versi 1.2 pada tahun 1998, dan versi ini mulai disebut sebagai Java2. Java2 hadir dalam beberapa versi (edisi):

- J2EE (Java2 Enterprise Edition): merupakan versi Java yang diterapkan untuk server
- J2SE (Java2 Standard Edition): merupakan versi Java yang diterapkan untuk desktop
- J2ME (Java2 Micro Edition): merupakan versi Java yang diterapkan untuk peralatan *mobile* seperti *handphone* dan PDA.

Mulai 2006, Sun memberi nama versi Java-nya sebagai Java EE, Java SE dan Java ME.

Setiap versi Java akan terdiri atas Java API (*Application Programming Interface*) yang berisi kumpulan pustaka yang dapat digunakan dalam pemrograman dan Java Run Time Environment (JRE) yang merupakan lingkungan untuk menjalankan kode Java. Salah satu komponen dari JRE adalah JVM (Java Virtual Machine) yang menjadikan program Java dapat dijalankan di mana saja. Untuk sistem operasi yang berbeda dibutuhkan JVM yang berbeda pula. Namun demikian, program Java yang akan dijalankan diatas JVM tetap sama (*platform independent*).

Apa Yang Diperlukan?

Agar dapat membuat program Java dan menjalankannya, maka ada 4 hal yang harus disiapkan: pengetahuan mengenai strukur pemrograman dengan Java, *text editor* untuk menuliskan kode program, *compiler*, dan JVM untuk dapat menjalankan program yang telah disusun. Apabila menggunakan sistem operasi Microsoft Windows, dapat digunakan Notepad sebagai *text editor*-nya. Bisa juga menggunakan *text editor* gratis semacam Notepad++ yang memiliki kemampuan *syntax highlighting*. Ingat bahwa text editor berbeda dengan word processor sehingga Anda tidak dapat menggunakan MS-Office atau OpenOffice untuk menuliskan kode program Anda.

Selanjutnya Anda dapat meng-install JDK (*Java Development Kit*) atau disebut juga Java-SDK (*Java Software Development Kit*) yang terdiri atas Java *compiler*, Java API dan JRE (superset dari JVM). Versi terbaru dari JDK untuk saat ini adalah JDK versi 1.6 yang bisa diperoleh di http://java.sun.com.

Umumnya, alih-alih menggunakan *text editor* untuk menuliskan kode program, programmer lebih suka memanfaatkan IDE (*Integrated Development Environment*) di mana text editor dan compiler telah diintegrasikan dalam satu aplikasi. Untuk pemrograman Java, IDE yang cukup populer adalah Netbeans yang bisa diperoleh secara bebas di www.netbeans.org atau dari website Sun. Agar dapat digunakan dengan lancar, JDK harus di-*install* terlebih dahulu sebelum meng-*install* Netbeans.

Struktur Dasar Pemrograman Java

Program sederhana dalam Java ditunjukkan oleh contoh berikut:

```
public class coba {
   public static void main(String[] args) {
        System.out.println("Hallo dunia \n");
   }
}
```

Tulis kode program di atas dengan Notepad, simpan dengan nama **coba.java** di drive C. Selanjutnya lakukan kompilasi dengan langkah berikut: buka command prompt Windows, masuk ke drive C: dengan mengetik **cd** dan tekan tombol enter. Compile file Java Anda dengan perintah **javac coba.java** (perhatikan penggunaan huruf besar dan kecil dalam program dan pemberian nama file). Setelah kompilasi akan diperoleh file baru **coba.class** yang dapat dijalankan dengan perintah **java coba** dan di layar akan muncul tulisan:

```
Hallo dunia
```

Penjelasan:

Pada program di atas, kode program diawali oleh **`public class coba,`** di sini kita membuat *class* baru dengan nama **coba**. Ingat bahwa nama file ketika Anda menyimpan kode ini harus selalu sama dengan nama dari *class*. Class ini diberi akses *public* agar dapat dipanggil oleh *class* yang lain. Class selalu dimulai oleh tanda kurung kurawal buka **{** dan diakhiri oleh tanda kurung kurawal tutup **}** (yang merupakan ciri dari *block programming*).

Selanjutnya terdapat kode **`public static void main`** di mana *main* merupakan *method* yang akan dijalankan pertama kali oleh JVM. Method *main* diberi nilai *void* yang berarti method ini tidak memberikan nilai apapun. Method *main* juga diberi sifat *static* yang berarti dapat dijalankan tanpa membuat objek terlebih dahulu (ingat bahwa Java merupakan bahasa pemrograman yang berorientasi objek). Method *main* menerima variabel *args* yang bertipe *String* sebagai parameternya. Penggunaan tanda [ ] pada tipe variabel menunjukkan bahwa *args* merupakan variabel array. Method main juga diawali oleh tanda kurung kurawal buka { dan kurung kurawal tutup }.

Di dalam method main terdapat perintah **`System.out.println()`** yang meminta komputer untuk menampilkan parameter ke layar. Di sini parameter yang ditampilkan berupa teks `Hallo dunia` dan karena berupa teks maka harus diapit oleh sepasang tanda petik ganda. Ingat bahwa semua perintah harus diakhiri dengan tanda **;** (titik koma).

Dalam perintah **`System.out.println()`** terdapat karakter **\n**. Karakter ini dikenal

sebagai *escape character* yang menyebabkan kursor akan bergeser ke baris selanjutnya (penggunaan *method* **`println()`** sendiri akan menyebabkan kursor berpindah ke baris berikutnya setelah karakter terakhir ditampilkan). Selain \n untuk berpindah baris, terdapat juga *escape character* yang lain:

\n	meminta kursor untuk berganti baris
\"	menampilkan karakter " ke layar
\\	menampilkan karakter \ ke layar
\t	menampilkan karakter tab ke layar

Karakter berupa kalimat yang berada di antara tanda petik ganda " " pada pernyataan **`System.out.println(" ")`** disebut sebagai string. Untuk menampilkan teks panjang di layar, beberapa string dapat juga dijumlahkan dengan menggunakan operator jumlahan + seperti contoh berikut:

```
public class coba {
   public static void main(String[] args) {
       System.out.println("Hallo dunia" + "apa kabar?");
   }
}
```

Jika program di atas dijalankan, maka pada layar terminal akan muncul tulisan:

```
Hallo dunia apa kabar?
```

Menuliskan Komentar

Ketika menuliskan kode program, adakalanya Anda perlu menyisipkan komentar agar orang lain yang membaca program Anda dapat memahami kode Anda dengan lebih mudah atau Anda juga memerlukannya untuk mengingat program apa yang Anda tulis. Di dalam Java, komentar didahului oleh tanda **//** atau sepasang tanda **/*** dan ***/**. Apabila komentar cukup pendek dan hanya menempati satu baris biasanya digunakan tanda // sedangkan jika komentar Anda cukup panjang dan menempati beberapa baris maka didahului oleh /* dan

diakhiri oleh */. Komentar tidak akan diproses ketika kode program di-*compile*.

Maka program sebelumnya dapat ditulis dengan komentar:

```
/* ---------------------------------------
      Contoh Program Sederhana
--------------------------------------- */

public class coba {
   public static void main(String[] args) {
       System.out.println("Hallo dunia");   // ini perintah
   }
}
```

Tugas

Buatlah program sederhana untuk menampilkan data diri Anda (nama, NIM, prodi). Simpan dengan nama file **a_nim.java** (nama *class* disesuaikan).

Catatan:

Untuk pengguna Sistem Operasi MS-Windows, program JDK yang telah terinstall tidak dapat digunakan sebelum PATH pada sistem diarahkan ke folder bin dari paket JDK. Cara men-setting PATH pada MS-Windows-XP adalah sebagai berikut:

Masuk ke **Control Panel > System > Advance** lalu klik pada tombol **Environment Variabels.** Pada kotak **System Variabels** klik **PATH** lalu klik tombol **Edit**. Selanjutnya masukkan alamat lengkap folder bin dari paket JDK yang terinstall dengan sebelumnya memberi tanda titik koma (;). Misal:

; C:\Program Files\Java\jdk1.6.0_11\bin

Kemudian klik tombol OK. Ketikkan **javac** pada *command prompt*, apabila masih ada pesan '***javac' is not recognised,*** coba restart sistem operasi Anda.

MODUL 2: VARIABEL DAN TIPE DATA

Dalam pemrograman, tentu akan sering sekali digunakan data yang disimpan dalam variabel. Ada beberapa macam atau tipe data yang tersedia, dan sebelum variabel digunakan untuk menyimpan data, terlebih dahulu harus dideklarasikan tipe data yang akan disimpan.

Terdapat 2 macam tipe data dalam Java, yaitu referensi dan data primitif. Sebagai bahasa pemrograman yang berorientasi objek, maka data yang digunakan dalam Java akan berupa referensi terhadap objek. Namun, kita masih dapat menggunakan tipe data biasa yang umum ditemui pada bahasa pemrograman lain. Tipe data biasa ini yang kemudian disebut sebagai tipe data primitif. Tipe data ini dapat dikelompokkan ke dalam empat tipe lagi:

- Tipe data integer (bilangan bulat), yaitu: *byte*, *short, int, long*.
- Tipe data float (bilangan nyata/real), yaitu: *float, double*.
- Tipe data char (karakter), yaitu: *char*.
- Tipe data boolean, yang memiliki nilai *true* atau *false*.

Tipe data yang berbeda akan memiliki ukuran yang berbeda dan nilai batas minimal /maksimal yang berbeda pula yang ditunjukkan dalam tabel berikut:

Tipe data	Ukuran (byte)	Nilai minimal	Nilai maksimal
byte	1	-128	127
short	2	-32768	32768
int	4	-2147483648	2147483647
long	8	-9223372036854775808	9223372036854775807
float	4	± 3,4 E-38	± 3,4 E+38
double	8	± 1,7 E-308	± 1,7 E+308
char	2	\u0000	\uFFFF

Dalam pemrograman Java, variabel harus dideklarasikan dulu (yang menjadi ciri bahasa pemrograman yang *strongly typed*) dengan menyatakan nama variabel yang didahului oleh tipe data dari variabel itu. Untuk beberapa variabel yang memiliki tipe yang sama, dapat dideklarasikan dalam satu baris yang sama, dengan dipisahkan oleh tanda

koma. Sebagai contoh:

```
int a;           //variabel a memiliki tipe int
float x, y, z;   // variabel x, y dan z memiliki tipe float
char n;          // variabel n memiliki tipe char
```

Nilai atau data yang tersimpan dalam sebuah variabel dapat dinyatakan bersamaan dengan deklarasi variabelnya, atau bisa juga dimasukkan setelahnya. Untuk memasukkan nilai atau data ke dalam variabel, digunakan operator *assignment* berupa tanda sama dengan. Sebagai contoh:

```
int   a = 30, b = 2, c = 100+50;
float x = 3.5f;
char  m='Y', n='N';
```

Menyatakan Konstanta

Adakalanya kita memerlukan variabel yang berisi data yang tidak berubah (berupa konstanta). Di dalam Java, menyatakan konstanta dilakukan dengan menambahkan kata kunci **`final`** di depan tipe variabel. Sebagai contoh:

```
final double pi = 3.14;
final int na = 14;
```

Operator Numerik

Data yang tersimpan dalam variabel selanjutnya akan diolah. Untuk melakukan pengolahan data ini salah satunya dengan menggunakan operator. Ada beberapa macam operator yang dapat digunakan, diantaranya adalah: operator assignment (yang sudah ditunjukkan sebelumnya), operator logika, operator numerik, operator perbandingan dan operator *increment* dan *decrement*. Operator perbandingan, operator logika dan operator increment-decrement akan dipelajari di modul berikutnya. Operator numerik (aritmatik) ditunjukkan sebagai berikut:

Operator	Simbol
Perkalian	*
Pembagian	/
Penjumlahan	+
Pengurangan	-
Sisa (modulus)	%

Operasi Masukan

Dalam bahasa pemrograman seperti C, Pascal atau BASIC, terdapat fungsi atau pernyataan yang dapat digunakan untuk meminta masukan data dari pengguna program. Bahasa Pascal menggunakan fungsi **`readln()`** untuk meminta masukan dalam mode teks, begitu juga untuk bahasa C dikenal fungsi **`scanf()`**. Karena Java lebih dikembangkan untuk membuat program dengan antarmuka dalam bentuk grafis, fungsi untuk meminta masukan dalam modus teks tidak disediakan. Dalam hal ini programer akan memanfaatkan argumen dari metode utama (main) untuk meminta masukan dari pengguna program.

Perhatikan contoh program berikut:

```
// program menghitung luas dan keliling lingkaran
// membutuhkan masukan berupa besarnya jejari lingkaran
// dijalankan dengan: java lingkarantext jejari
// contoh: java lingkarantext 10

public class lingkarantext {
      public static void main(String args[]){
            final float pi=3.14f;
            float luas, keliling;
            float jejari = Float.parseFloat(args[ 0]);

            luas = pi*jejari*jejari;
            keliling = 2*pi*jejari;
```

```
            System.out.println("Lingkaran berjejari "+jejari+" cm");
            System.out.println("memiliki keliling "+keliling+" cm");
            System.out.println("dan luas "+luas+" cm");
      }
}
```

Penjelasan:

Ketika program di atas dijalankan dengan perintah

`java lingkarantext`

maka akan muncul pesan kesalahan (*error*), karena program di atas seharusnya disertai argumen masukan berupa angka yang menunjukkan besarnya jejari suatu lingkaran. Hal ini tentu akan membingungkan pengguna, kecuali jika diberitahu sebelumnya bahwa program tersebut merupakan program untuk menghitung luas dan keliling suatu lingkaran, dan harus dijalankan dengan memberi argumen masukan berupa nilai jejari lingkaran, yaitu

`java lingkarantext jejari`

dengan **`jejari`** diganti angka yang menunjukkan besarnya jejari sebuah lingkaran.

Angka jejari yang dimasukkan memiliki tipe String karena terdapat parameter **`String args[ ]`** pada metode **`main`**, sehingga agar dapat dilakukan operasi matematik maka data tersebut diubah ke tipe float dengan perintah **`Float.parseFloat(args[0])`**. Penggunaan variabel **`args[0]`** menunjukkan bahwa argumen pertamalah yang akan dijadikan masukan nilai jejari lingkarannya (meskipun pengguna dapat memasukkan argumen lebih dari satu yang dipisahkan oleh spasi).

Selanjutnya perhatikan contoh program kedua berikut:

```
// program menghitung luas dan keliling lingkaran
// membutuhkan masukan berupa besarnya jejari lingkaran
// masukkan diisikan melalui box dialog

import javax.swing.*;

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

```
            String input=JOptionPane.showInputDialog("Masukkan jejari
lingkaran");
            float jejari=Float.parseFloat(input);
            float luas, keliling;
            final float pi=3.14f;

            luas = pi*jejari*jejari;
            keliling = 2*pi*jejari;

            JOptionPane.showMessageDialog(null,"Lingkaran    berjejari
"+jejari+" cm \n memiliki keliling "+keliling+" cm \n dan luas
"+luas+" cm");

            System.exit(0);
      }
}
```

Penjelasan:

Program di atas dapat dijalankan dengan perintah

```
java lingkaranbox
```

Selanjutnya akan muncul kotak dialog masukan yang meminta pengguna mengisikan nilai jejari suatu lingkaran. Setelah mengisikan data, pengguna akan mendapatkan kotak dialog pesan yang menampilkan data luas dan keliling lingkaran.

Pada contoh kedua ini, pengguna diarahkan oleh kotak dialog yang secara tidak langsung memberitahukan bahwa program yang sedang dijalankannya adalah program yang akan menghitung luas dan keliling sebuah lingkaran. Hal ini berbeda dengan contoh program pertama dimana pengguna harus diberitahu terlebih dahulu sebelum menggunakan program.

Penggunaan tampilan grafis berupa kotak dialog masukan dan pesan sebenarnya merupakan fasilitas dasar pada Java. Berbeda dengan bahasa pemrograman lain yang fasilitas dasar untuk memasukkan dan menampilkan datanya berada pada mode teks.

Keuntungan dari penggunaan Java (dibandingkan dengan bahasa C, Pascal atau BASIC) dalam pembuatan program dengan antarmuka grafis adalah tersedianya *class* untuk membuat tampilan grafis yang dapat langsung dipakai oleh programer sehingga tidak perlu membuatnya sendiri.

Pada contoh program kedua, digunakan *class* **`JoptionPane`** untuk menampilkan antarmuka grafis berupa kotak dialog masukan (melalui *method* **`showInputDialog()`**) dan kotak dialog pesan (melalui *method* **`showMessageDialog()`**). *Class* **`JoptionPane`** ini terdapat dalam paket **javax.swing** sehingga perlu disertakan dalam program melalui pernyataan **`import javax.swing.*`** dimana tanda * menyatakan bahwa yang disertakan dari paket **javax.swing** adalah seluruh *class* yang tersedia di dalam paket tersebut. Apabila yang digunakan hanya *class* **`JoptionPane`** saja, maka penyertaan paket dalam program di atas dapat pula dinyatakan dengan **`import javax.swing.JOptionPane`**.

Tugas

Buatlah program yang meminta masukan berupa nilai jejari alas sebuah silinder dan tingginya kemudian menampilkan data silinder tersebut secara lengkap (berupa jejari alas, tinggi, volume dan luas permukaan silinder). Simpan dengan nama file **a_nim.java** (sesuaikan nama *class*-nya).

MODUL 3: ALUR PROGRAM (PERCABANGAN)

Operator Relasi dan Logika

Dalam pemrograman, sering kali ditemui kasus dimana suatu perintah hanya akan dilaksanakan jika kondisi atau syarat tertentu telah dipenuhi sebelumnya. Dalam pemrograman dengan Java, ada beberapa pernyataan yang bisa digunakan agar komputer mengecek dahulu apakah syarat sudah dipenuhi sebelum melakukan suatu pernyataan, yaitu: **`if`**, kombinasi **`if-else`** dan **`switch`**.

Untuk mengetahui apakah suatu syarat telah dipenuhi, diperlukan nilai benar atau salah yang dibangkitkan melalui operasi relasi atau melalui operasi logika. Operasi relasi (hubungan) memerlukan operator relasi berikut:

Operator	Keterangan
>	Lebih besar
<	Lebih kecil
>=	Lebih besar atau sama dengan
<=	Lebih kecil atau sama dengan
= =	Sama dengan
!=	Tidak sama dengan

Contoh:

`x > 3` : bernilai benar jika **`x`** lebih dari 3 dan bernilai salah jika **`x`** kurang dari 3

`a == 5` : bernilai benar jika **`a`** bernilai 5 dan salah jika **`a`** bernilai selain 5

Selain menggunakan operator relasi, nilai benar atau salah dapat juga dibangkitkan melalui operator logika berikut:

Operator	Simbol	Keterangan
AND	&&	Bernilai benar hanya jika keduanya benar
OR	||	Bernilai salah jika keduanya salah
NOT	!	Bernilai benar jika ekspresi salah dan bernilai salah jika ekspresi benar
XOR	^	Bernilai benar jika kedua ekspresi berbeda nilainya

Contoh:

`a && b` : bernilai benar hanya jika **`a`** dan **`b`** keduanya bernilai benar, selain itu salah

`a || b` : bernilai salah hanya jika **`a`** dan **`b`** keduanya bernilai salah, selain itu benar

Pernyataan `if`

Melalui pernyataan **`if`**, maka suatu pernyataan (atau kumpulan pernyataan) akan dilaksanakan apabila syarat dipenuhi. Format pernyataan **`if`** adalah:

`if` (syarat) pernyataan ;

Apabila pernyataan yang harus dikerjakan lebih dari satu, maka kumpulan pernyataan itu dikalang oleh kurung kurawal buka dan tutup (disebut blok pernyataan):

```
if (syarat) {
pernyataan1 ;
pernyataan2 ;
pernyataan3; }
```

Pada pernyataan **`if`** di atas, kumpulan atau suatu pernyataan hanya akan dikerjakan apabila kondisi dipenuhi, dan jika tidak, maka kumpulan pernyataan tersebut diabaikan. Terkadang suatu program juga menginginkan komputer mengerjakan pernyataan yang lain apabila syarat tidak dipenuhi. Dalam hal ini, kita menggunakan pernyataan **`if-else`** dengan format berikut:

`if` (syarat) pernyataan_A **`else`** pernyataan_B ;

Apabila pernyataan yang harus dikerjakan lebih dari satu, maka digunakan blok pernyataan:

```
if (syarat) { pernyataan_A1;
              pernyataan_A2; }
else { pernyataan_B1;
       pernyataan_B2; }
```

Pernyataan **if-else** juga dapat dinyatakan dalam bentuk bertingkat :

```
if  (syarat_A) { pernyataan_A1;
                 pernyataan_A2; }
else if (syarat_B) { pernyataan_B1;
                     pernyataan_B2;}
else { pernyataan_C1;
       pernyataan_C2;}
```

Pada kasus tertentu, **if-else** bertingkat dapat digantikan dengan perintah **switch** dengan format sebagai berikut:

```
switch (ekspresi)
{
      case nilai1  :  pernyataan1 ;
                      break ;
      case nilai2  :  pernyataan2 ;
                      break ;
      case nilai3  :  pernyataan3 ;
                      break ;
      default      : pernyataan ;
}
```

Pada pernyataan **switch** di atas, ekspresi harus berupa pernyataan yang bernilai integer atau karakter. Ekspresi dapat diganti dengan suatu variabel bertipe integer atau karakter dengan nilai1, nilai2, nilai3 berupa nilai yang mungkin dari variabel tersebut.

Contoh Program:

```
// program contoh penggunaan if-else

import javax.swing.*;

public class modul3java {
      public static void main (String args[]) {
            String input=JOptionPane.showInputDialog("Masukkan sebuah
                  bilangan bulat [0-100]: ");
            int angka=Integer.parseInt(input);
            String hasil;

            if (angka <= 100 && angka >= 80)
                  hasil = "angka Anda berada di antara 80 dan 100 ";
            else if (angka < 80 && angka >=60 )
                  hasil = "angka Anda berada di antara 60 dan 80 " ;
            else if (angka < 60 && angka >=40)
                  hasil = "angka Anda berada di antara 40 dan 60 ";
            else if (angka < 40 && angka >=20)
                  hasil = "angka Anda berada di antara 20 dan 40 ";
            else if (angka < 20 && angka >=0)
                  hasil = "angka Anda berada di antara 0 dan 20 ";
            else
                  hasil = "angka Anda berada di luar jangkauan
bilangan 0-100 ";

            JOptionPane.showMessageDialog(null,hasil);
            System.exit(0);
      }
}
```

Tugas

Buatlah program yang meminta masukan 3 bilangan, kemudian tentukan bilangan mana yang terkecil.

MODUL 4: ALUR PROGRAM (PERULANGAN)

Dalam membuat program, terkadang ada beberapa perintah atau pernyataan yang ingin diulang beberapa kali selama nilai suatu variabel belum memenuhi nilai tertentu. Alih-alih menuliskan pernyataan tadi secara berulang, kita dapat menyatakannya melalui pernyataan perulangan **`for`** atau **`while`** atau **`do-while`**. Sebelumnya, harus dipahami terlebih dahulu istilah operator *decrement* (penurunan) dan *increment* (kenaikan).

Operator decrement dan increment

Untuk menambah nilai suatu variabel berisi data integer dengan nilai 1 dapat dinyatakan melalui ekspresi:

```
x = x + 1 ;
```

sehingga apabila nilai awalnya nilai **`x`** adalah 3, maka setelah pernyataan di atas nilai **`x`** menjadi 4.

Di dalam Java, pernyataan tersebut dapat dituliskan secara ringkas sebagai

```
++x ;
```

yang dikenal dengan nama *pre increment operator*. Artinya, data pada suatu variabel yang dikenai operator tersebut akan ditambah nilai 1 terlebih dahulu sebelum digunakan pada ekspresi berikutnya.

Selain operator *pre increment*, terdapat juga operator *post increment*, yang dituliskan sebagai

```
x++ ;
```

Pada pernyataan di atas, isi variabel **`x`** akan dipakai terlebih dahulu pada suatu pernyataan sebelum ditambah dengan nilai 1. Operator *decrement* juga memiliki bentuk yang mirip, kecuali operasi penjumlahan diganti dengan operasi pengurangan. Selengkapnya ditunjukkan pada tabel berikut:

Operator	Nama	Keterangan
`var++`	Post increment	Nilai yang ada di sebuah variabel digunakan dahulu, kemudian baru ditambah dengan satu
`++var`	Pre increment	Nilai yang ada di sebuah variabel ditambah satu, kemudian baru digunakan
`var--`	Post decrement	Nilai yang ada di sebuah variabel digunakan dahulu, baru dikurang satu

`--var`	Pre decrement	Nilai yang ada di sebuah variabel dikurang satu, baru kemudian digunakan

Perulangan dengan *for*

Format perintah perulangan dengan menggunakan **for** adalah sebagai berikut:

```
for (ekspresi_1; ekspresi_2; ekspresi_3) pernyataan ;
```

Sedangkan bentuk perulangan dimana terdapat lebih dari satu pernyataan dapat dinyatakan dalam bentuk blok pernyataan:

```
for (ekspresi_1; ekspresi_2; ekspresi_3) {
      pernyataan1;
      pernyataan2;
      pernyataan3;
      }
```

dimana **ekspresi_1** merupakan nilai awal dari variabel yang mengontrol perulangan, **ekspresi_2** menyatakan syarat kapan perulangan akan berhenti, sedangkan **ekspresi_3** merupakan pengendali besarnya kenaikan (atau penurunan) nilai dari variabel yang mengendalikan operasi perulangan.

Contoh:

```
public class modul4a {
      public static void main(String[] args) {
            int x;
            for (x=1; x<=3; x++) System.out.println("isi x = " + x);
      }
}
```

Program di atas akan menampilkan hasil:

```
isi x=1
isi x=2
isi x=3
```

Perulangan dengan *while*

Selain dengan menggunakan pernyataan **for**, perulangan dapat juga memanfaatkan pernyataan **while**. Perbedaannya adalah, jika pada pernyataan **for** kita sudah mengetahui berapa cacah perulangannya, pada pernyataan **while** cacah perulangan akan ditentukan oleh syarat atau kondisi yang dijadikan kontrol perulangan.

Format penulisan perintah perulangan dengan **while** adalah sebagai berikut:

```
while (syarat) {
      pernyataan;
      pernyataan;
      }
```

Pada dasarnya, perulangan dengan for dapat diubah ke perulangan dengan **while**. Dengan menggunakan istilah yang dipakai pada format perulangan dengan **for**, maka jika dinyatakan ke bentuk perulangan dengan **while** adalah:

```
ekspresi_1;
while (ekspresi_2) {
      pernyataan1;
      pernyataan2;
      pernyataan3;
      ekspresi_3
      }
```

sehingga contoh program sebelumnya dapat dituliskan sebagai:

```
public class modul4b {
      public static void main(String[] args) {
            int x;
```

```
            x=1;
            while (x<=3) {
                  System.out.println("isi x = "+x);
            x++ ;
      }
}
```

Program di atas juga akan menghasilkan tampilan:

```
isi x=1
isi x=2
isi x=3
```

Perulangan dengan *do-while*

Selain menggunakan pernyataan **for** dan **while**, perulangan juga dapat dilakukan dengan pernyataan **do-while**. Pernyataan **do-while** memiliki format berikut:

```
do {
      pernyataan;
      pernyataan;
} while (syarat);
```

Perbedaan dari pernyataan **while** dengan **do-while** adalah: pernyataan **while** menguji syarat di awal, sehingga apabila syarat tidak dipenuhi, maka blok pernyataan tidak akan dikerjakan. Sedangkan pernyataan **do-while** menguji syarat di belakang, sehingga meskipun syarat tidak dipenuhi, blok pernyataan tetap akan dikerjakan satu kali. Perhatikan penggalan 2 program berikut:

```
int m=6;

while (m <= 5) {
      System.out.println("Halo");
}
```

```
int m=6;

do {
      System.out.println("Halo");
} while (m<=5);
```

Pada program di sebelah kiri, perintah mencetak string "**Halo**" di layar tidak akan dikerjakan,

karena syarat yang diminta tidak dipenuhi. Sedangkan pada program di sebelah kanan, pada layar akan tercetak string "**Halo**" meskipun syarat tidak dipenuhi. Namun, karena syarat baru diuji di akhir blok perintah, maka perintah mencetak karakter ke layar tetap dikerjakan satu kali.

Pelajari Contoh Program berikut:

```
// program contoh penggunaan alur perulangan

import javax.swing.*;

public class modul4c {
      public static void main (String args[]) {
            int k, x;
            String input=JOptionPane.showInputDialog("Masukkan sebuah
 bilangan bulat: ");
            int n = Integer.parseInt(input);

            x = 0;
            for (k=1; k<=n; k++) {
                  x=x+k; // hitungan ini berulang n kali
            }

            JOptionPane.showMessageDialog(null,"Jumlahan dari 0
hingga " +n+" adalah "+x);

            System.exit(0);
      }
}
```

Tugas 1

Buatlah program untuk menghitung nilai faktorial suatu bilangan. Simpan dengan nama file **d1_nim.java**.

Tugas 2

Buatlah program untuk menghitung nilai pangkat suatu bilangan. Simpan dengan nama file **d2_nim.java**.

Perulangan Bertingkat

Perintah perulangan (*looping*) atau iterasi dapat disusun secara bertingkat yang dikenal dengan istilah *loop* dalam *loop* (*nested loop*). Perhatikan contoh program berikut:

```
public class modul4d {
      public static void main (String args[]) {

      int m, k;
      for (m=1; m<=3; m++) {
            System.out.println("mulai iterasi ke-"+m);
            for (k=1; k<=2; k++)
                  System.out.println("ini  sub-iterasi  ke-"+k+"  dari
iterasi ke-"+m);
            }
      }
}
```

Pada program di atas, variabel **m** menjadi variabel kendali perulangan utama. Perulangan dilakukan sebanyak 3 kali. Untuk setiap perulangan, dilakukan 2 perulangan lagi (sub-iterasi) dimana variabel **k** menjadi variabel kendali perulangan di dalam perulangan utama. Apabila program dijalankan, maka di layar terminal akan diperoleh hasil:

```
mulai iterasi ke-1
ini sub-iterasi ke-1 dari iterasi ke-1
ini sub-iterasi ke-2 dari iterasi ke-1
mulai iterasi ke-2
ini sub-iterasi ke-1 dari iterasi ke-2
ini sub-iterasi ke-2 dari iterasi ke-2
mulai iterasi ke-3
```

```
ini sub-iterasi ke-1 dari iterasi ke-3
ini sub-iterasi ke-2 dari iterasi ke-3
```

Pelajari contoh berikut:

```
public class modul4e {
      public static void main (String args[]) {

      int k, m, x;
      System.out.println(" Nilai Faktorial Bilangan ");
      System.out.println(" -------------------------");

      for (k=1; k<=10; k++){
            x=1;
            for (m=1; m<=k; m++)  x=x*m;
            System.out.println(" Nilai "+k+"! adalah "+x);
            }
      }
}
```

Tugas

Berdasarkan contoh program di atas, buatlah program untuk menampilkan nilai pangkat dari bilangan 2, simpan dengan nama **d3_nim.java**. Contoh hasil program ditunjukkan sebagai berikut (Anda bisa menampilkannya di konsole, sehingga tidak perlu menggunakan kelas swing):

```
Nilai 2 pangkat 1 adalah 2
Nilai 2 pangkat 2 adalah 4
Nilai 2 pangkat 3 adalah 8
Nilai 2 pangkat 4 adalah 16
Nilai 2 pangkat 5 adalah 32
Nilai 2 pangkat 6 adalah 64
Nilai 2 pangkat 7 adalah 128
Nilai 2 pangkat 8 adalah 256
Nilai 2 pangkat 9 adalah 512
Nilai 2 pangkat 10 adalah 1024
```

MODUL 5: METHOD

Dalam bahasa pemrograman prosedural, dikenal adanya subprogram atau subroutine (dalam bahasa C dikenal dengan istilah function) yang berisi blok perintah yang dapat dipanggil di dalam program utama. Adanya subprogram ini dapat memudahkan penulis program dan orang lain yang membaca program sumber (source code) dalam memahami jalannya suatu program.

Dalam pemrograman Java, konsep subprogram mungkin agak sedikit berbeda karena Java merupakan bahasa pemrograman yang murni berorientasi objek (Object Oriented Programing, OOP, akan dibahas di modul berikutnya). Namun demikian, kita dapat memanfaatkan method (metode) yang menyusun setiap kelas dalam Java untuk membuat function dalam bahasa pemrograman prosedural.

Setiap kelas dalam program Java dapat terdiri atas beberapa method. Ketika kita meng-compile sebuah program Java dan menjalankannya, mesin Java akan menjalankan method main. Setiap method dapat mengandung masukan (input) atau argumen dan keluaran (output). Format penulisan method adalah sebagai berikut:

```
TipeKeluaran NamaMethod (tipe1 argumen1, tipe2 argumen2, ... )
{
      ekspresi1;
      ekspresi2;
      ...
}
```

Tipe keluaran dan argumen dapat berupa tipe data primitif (int, float, double, char, boolean) atau referensi (String) atau berupa **void** jika suatu method tidak menghasilkan nilai. Tipe keluaran akan ditentukan oleh nilai yang dihasilkan oleh method. Begitu pula dengan tipe argumen akan ditentukan oleh nilai yang dimasukkan sebagai argumen. Untuk mengeluarkan (mengembalikan) nilai dari dalam method, digunakan kata kunci **return**. Apabila method yang dibuat diperlakukan layaknya function dalam bahasa pemrograman prosedural, maka method tersebut harus dinyatakan dengan modifier **static**.
Perhatikan program contoh berikut:

```
// program contoh penggunaan method dalam kelas
import javax.swing.JOptionPane;

public class modul5 {

      static double pangkat (float a, int b) {
            //method untuk menghitung nilai a pangkat b
            int i;
            double m=1.0;

            for (i=1;i<=b;i++) m=m*a;
            return m;
      }

      public static void main (String args[]) {
            int n;
            float x;
            double hasil=0.0;
            String input = JoptionPane.showInputDialog("Masukkan
sebuah bilangan antara 0 dan 2: ");
            float y = Float.parseFloat(input);

            x = y - 1;
            for (n=1;n<=100;n++)
                  hasil = hasil + pangkat(x,n)*pangkat(-1,n+1)/n;

            JOptionPane.showMessageDialog(null,"Nilai logaritma
natural dari "+y+" adalah "+hasil);

            System.exit(0);
      }
}
```

Penjelasan:

Program di atas akan meminta pengguna untuk memasukkan sebuah bilangan, kemudian menghitung nilai logaritma naturalnya menggunakan bentuk deret:

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \cdots = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{x^n}{n}$$

Namun, bentuk deret ini hanya konvergen pada interval nilai -1 < x < 1, sehingga hanya akurat untuk menentukan nilai logaritma natural antara 0 dan 2. Keakuratannya juga akan ditentukan oleh banyaknya suku deret yang dilibatkan dalam perhitungan. Semakin banyak akan memberikan nilai yang semakin akurat.

Dalam program di atas, method untuk menghitung nilai pangkat dinyatakan terlebih dahulu. Method ini memiliki dua argumen (masukan) yaitu ***a*** yang bertipe float dan ***b*** yang bertipe integer. Selanjutnya hasil dari perhitungan dalam method pangkat disimpan dalam variabel ***m*** yang dinyatakan bertipe double, maka method pangkat dinyatakan bertipe double juga. Nilai yang tersimpan dalam variable m selanjutnya dikembalikan (dikeluarkan) sebagai nilai keluaran method pangkat melalui pernyataan **return m**.

Tugas1

Buatlah program untuk menentukan nilai pangkat exponensial (e^x). Gunakan bentuk deret

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \cdots = 1 + \sum_{n=1}^{\infty} \frac{x^n}{n!}$$

Tugas2

Buatlah program untuk menentukan nilai sinus suatu sudut. Gunakan bentuk deret

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{x^{(2n-1)}}{(2n-1)!}$$

MODUL 6: PEMROGRAMAN BERORIENTASI OBJEK

Bahasa pemrograman Java merupakan contoh bahasa pemrograman yang murni berorientasi objek, berbeda dari bahasa C, Pascal, atau Fortran yang berorientasi prosedur (bersifat prosedural). Pemrograman berorientasi prosedur disusun dengan konsep bahwa program akan terdiri atas beberapa prosedur (subprogram) yang dapat dipanggil dalam program utama. Hal ini berbeda dengan pemrograman berorientasi objek yang disusun dengan konsep bahwa program akan terdiri atas beberapa objek yang dapat diciptakan dari suatu kelas dan mewarisi sifat-sifat dan kelakuan yang dimiliki oleh kelas tersebut.

Objek dan Kelas

Dalam pemrograman berorientasi objek, sebelum dapat menciptakan satu atau beberapa objek, kita harus menciptakan kelas dari objek tersebut. Kelas dapat dipandang sebagai cetakan atau template dari objek. Dari kelas yang merupakan konsep objek abstrak dapat dibentuk **instance** yang merupakan konsep objek nyata. Istilah instance (contoh) dan objek dalam pemrograman berorientasi objek umumnya dapat saling menggantikan. Sebagai contoh, kita dapat membuat kelas `mahasiswa` sebagai objek umum. Dari kelas `mahasiswa` ini, kita dapat menciptakan ***mahasiswa1***, ***mahasiswa2***, ***mahasiswa3*** dan seterusnya yang merupakan objek nyatanya.

Atribut dan Behaviour (perilaku)

Setiap kelas yang dibuat akan memiliki dua bagian utama, yaitu atribut dan behaviour. Atribut di dalam kelas akan dinyatakan oleh variabel. Objek-objek (instance) yang diciptakan dari kelas ini dapat memiliki nilai yang berbeda untuk variabel yang sama, yang kemudian disebut sebagai ***variabel instance***. Ketika nilai variabel suatu objek diubah, berarti atribut dari objek tersebut berubah. Sebagai contoh, kelas *mahasiswa* yang sudah dibuat dapat memiliki atribut nama, nomor_induk, tahun_angkatan dan nilai_ipk. Maka masing-masing instance dapat memiliki nilai atribut yang berbeda. Misal untuk atribut nama, *mahasiswa1* memiliki nama “Budi”, *mahasiswa2* memiliki nama “Nina”, dan mahasiswa3 memiliki nama “Tono”. Nilai variabel ini tersimpan di masing-masing objek. Namun dapat juga variabel dari semua objek memiliki nilai yang sama dan nilai ini tersimpan di kelasnya. Misal, kelas mahasiswa yang dibuat memiliki atribut tahun_angkatan bernilai 2005, maka

semua objek yang dibentuk dari kelas ini akan otomatis memiliki variabel tahun_angkatan bernilai 2005. Apabila nilai ini diubah, maka nilai variabel angkatan untuk semua objek yang dibentuk dari kelas mahasiswa akan ikut berubah. Variabel seperti ini disebut sebagai ***variabel kelas***.

Selain atribut, kelas juga memiliki behaviour yang ditunjukkan oleh method (metode). Setiap objek yang dibentuk dari kelas ini dapat mengakses metode yang dimiliki oleh kelas tersebut. Terdapat 2 tipe method, yaitu ***method kelas*** dan ***method instance***. Method kelas digunakan dan bekerja hanya pada kelas itu sendiri, sementara method instance merupakan method yang bekerja pada objeknya dan diperoleh dari instance kelas.

Perhatikan contoh program berikut:

```
public class mahasiswa {

      //atribut kelas mahasiswa
      String nama, nim, prodi;
      Boolean aktif = false;
      static String angkatan = "2005";

      //perilaku kelas mahasiswa
      void daftarKuliah() {
            System.out.println("telah mendaftar kuliah... ");
            System.out.println("berstatus mahasiswa aktif...");
            aktif = true;
      }

      void kuliah() {
            System.out.println("sedang mengikuti perkuliahan...");
            System.out.println("belajar sungguh-sungguh...");
      }
}
```

Pada program di atas, kita membuat kelas **`mahasiswa`** yang memiliki atribut umum: nama, nim, prodi, ipk, angkatan dan aktif. Untuk atribut angkatan diisikan nilai default 2005. Karena

variabel angkatan bersifat **static**, maka apabila ada objek yang mengganti nilai variabel angkatan, maka nilai ini juga akan dikenakan pada objek yang lain.

Perilaku yang dimiliki kelas ini adalah **daftarKuliah()** dan **kuliah()**. Karena tidak ada yang bersifat static, maka kedua method merupakan method instance. Program ini dapat di*compile* namun tidak dapat dijalankan karena tidak memiliki method **main()**.

Perhatikan contoh berikut yang merupakan program contoh sebelumnya namun diberi mehod **main()** agar dapat dijalankan. Dalam program ini, objek nyata mulai dibentuk dari objek abstraknya melalui pernyataan **new**. Agar memudahkan, simpan sebagai kelas **mahasiswaku**:

```
public class mahasiswaku {

      String nama, nim, prodi;   //atribut kelas mahasiswaku

      //perilaku kelas mahasiswa
      void kuliah() {
            System.out.println("sedang mengikuti perkuliahan...");
            System.out.println("belajar sungguh-sungguh...");
      }

      public static void main(String args[]){
            mahasiswa mhs1 = new mahasiswa();    //membuat objek mhs1
            mahasiswa mhs2 = new mahasiswa();    //membuat objek mhs2

            mhs1.nama = "Budi";
            mhs1.nim  = "037";
            mhs2.nama = "Nina";
            mhs2.nim  = "026";

            System.out.println("Data mahasiswa 1");
            System.out.println("Nama : "+mhs1.nama);
            System.out.println("NIM  : "+mhs1.nim);
            mhs1.kuliah();
```

```
            System.out.println("\n Data mahasiswa 2");
            System.out.println("Nama : "+mhs2.nama);
            System.out.println("NIM  : "+mhs2.nim);
            mhs2.kuliah();
      }
}
```

Pewarisan (Inheritance)

Dalam pemrograman berorientasi objek, selain konsep penting bahwa program akan terdiri atas objek yang memiliki atribut dan perilaku yang dimiliki kelasnya, terdapat konsep penting lainnya yang berupa konsep pewarisan. Dalam konsep ini, suatu kelas dapat mewarisi kelas lainnya. Kelas yang mewariskan disebut ***superkelas***, sedangkan kelas yang diwarisi disebut ***subkelas***. Arti mewarisi di sini adalah bahwa subkelas dapat menggunakan seluruh variabel dan method yang ada di dalam superkelas tanpa perlu menambahkan atau membuat variabel dan method yang sama dengan variabel dan method yang sudah ada di dalam superkelas. Dalam pemrograman Java, pewarisan ini akan dinyatakan dengan pernyataan ***extends***.

Perhatikan contoh berikut:

```
public class cahfisika extends mahasiswa {

      String prodi = "Fisika";

      void kuliah() {
            System.out.println("sedang      mengikuti      perkuliahan
visualisasi fisika...");
            System.out.println("belajar   sungguh-sungguh   pemrograman
Java...");
      }

      public static void main(String args[]){

            cahfisika mhs1 = new cahfisika();    //membuat objek mhs1
```

```
        cahfisika mhs2 = new cahfisika();    //membuat objek mhs2

        mhs1.nama = "Budi";
        mhs1.nim  = "037";
        mhs2.nama = "Nina";
        mhs2.nim  = "026";

        System.out.println("Data mahasiswa 1");
        System.out.println("Nama : "+mhs1.nama);
        System.out.println("NIM  : "+mhs1.nim);
        System.out.println("Prodi: "+mhs1.prodi);
        System.out.println("Angkatan: "+mhs1.angkatan);
        mhs1.daftarKuliah();
        System.out.println("Status aktif: "+mhs1.aktif);
        mhs1.kuliah();

        System.out.println("\n Data mahasiswa 2");
        System.out.println("Nama : "+mhs2.nama);
        System.out.println("NIM  : "+mhs2.nim);
        System.out.println("Prodi: "+mhs2.prodi);
        System.out.println("Angkatan: "+mhs2.angkatan);
        System.out.println("Status: "+mhs2.aktif);
        mhs2.kuliah();
    }
}
```

Bagaimana hasil keluaran program di atas jika variabel angkatan bagi mhs1 secara manual diisikan angka 2008? Akan terlihat bahwa variabel angkatan milik mhs2 otomatis juga berubah. Hal ini dikarenakan variabel angkatan dideklarasikan bersifat static.

Pada program di atas, kelas **`cahfisika`** merupakan subkelas dari kelas **`mahasiswa`**. Karena mewarisi semua variabel dan method yang dimiliki oleh kelas **`mahasiswa`**, maka semua variabel dan method yang ada di kelas **`mahasiswa`** dapat langsung digunakan tanpa harus dibuat terlebih dahulu di kelas **`cahfisika`**.

Jika diperhatikan, kelas **`mahasiswa`** memiliki method **`kuliah()`**. Namun, pada kelas

cahfisika, didefinisikan method **kuliah()** yang berbeda dengan method **kuliah()** yang ada di kelas **mahasiswa**. Ketika method **kuliah()** dipanggil, maka yang dijalankan adalah method **kuliah()** yang ada di kelas **cahfisika** dan seolah-olah method **kuliah()** milik kelas **mahasiswa** yang merupakan superkelas dari kelas **cahfisika** diabaikan pada jalannya program. Proses ini dikenal dengan istilah ***overriding*** (penindihan).

Konstruktor

Dalam pemrograman berorientasi objek, dikenal istilah konstruktor yang tidak lain merupakan sebuah method yang digunakan untuk memberikan nilai awal (menginisialisasi) sebuah instance (objek nyata) ketika instance tersebut dibuat. Meskipun merupakan sebuah method, konstruktor berbeda dengan method biasa, yaitu: konstruktor memiliki nama yang sama dengan nama kelas, tidak memiliki nilai kembalian sehingga tidak memiliki tipe kembalian, dan tidak dapat dipanggil secara langsung melainkan dipanggil otomatis saat objek baru dibuat.

Perhatikan program berikut yang merupakan modifikasi kelas **mahasiswaku** sebelumnya:

```
public class mahasiswaku {

      String nama, nim, prodi;   //atribut kelas mahasiswa ku

      //perilaku kelas mahasiswa
      void kuliah() {
            System.out.println("sedang mengikuti perkuliahan...");
            System.out.println("belajar sungguh-sungguh...");
      }

      //konstruktor
      mahasiswaku(String Nama, String NIM, String Prodi){
            nama = Nama;
            nim = NIM;
            prodi = Prodi;
      }
```

```
    public static void main(String args[]){
    mahasiswaku mhs1 = new mahasiswaku("Budi","037","Fisika");
    mahasiswaku mhs2 = new mahasiswaku("Nina","026","Kimia");

        System.out.println("Data mahasiswa 1");
        System.out.println("Nama : "+mhs1.nama);
        System.out.println("NIM  : "+mhs1.nim);
        System.out.println("Prodi: "+mhs1.prodi);
        mhs1.kuliah();

        System.out.println("\n Data mahasiswa 2");
        System.out.println("Nama : "+mhs2.nama);
        System.out.println("NIM  : "+mhs2.nim);
        System.out.println("Prodi: "+mhs2.prodi);
        mhs2.kuliah();
    }
}
```

MODUL 7: PEMROGRAMAN GRAFIS DENGAN APPLET

Dalam modul sebelumnya, program Java yang dibuat dijalankan dengan interpreter Java melalui jendela perintah (*command prompt*) atau terminal. Apabila program Java dinyatakan dalam bentuk applet, maka program tersebut dapat dijalankan melalui web browser yang mendukung Java (Firefox, Internet Explorer).

Applet dibangun dari subkelas kelas Applet yang merupakan bagian dari paket java.applet. Bagian utama dari applet memiliki format sebagai berikut:

```
public class namaKelas extends java.applet.Applet {
      pernyataan1;
      pernyataan2;
      pernyataan3;
}
```

Aplikasi Java yang dibuat pada modul sebelumnya selalu mengandung method main() yang merupakan method utama yang akan dieksekusi oleh interpreter Java ketika aplikasi tersebut dijalankan. Untuk applet, method main() tidak diperlukan, namun terdapat beberapa method yang akan dieksekusi oleh browser ketika applet dijalankan, yaitu: init(), start() dan paint(). Ketiga method tersebut sudah tersedia dalam kelas Applet, dan akan diwariskan ke subkelas yang kita buat. Namun, secara default, ketiga method tadi tidak akan melaksanakan apapun, sehingga kita perlu menindih (*overrride*) method tersebut agar applet bekerja seperti yang kita inginkan. Sebagai contoh perhatikan program berikut:

```
import java.awt.Graphics;

public class cobaApplet extends java.applet.Applet {
      public void paint(Graphics g){
            g.drawString("sedang belajar java",10,10);
      }
}
```

Setelah di*compile* dengan compiler Java (melalui perintah **`javac namakelas.java`**), selanjutnya dibuat file html yang akan memanggil applet di atas dengan format sebagai berikut:

```
<html>
<applet code="cobaApplet.class" width=300 height=50>
</applet>
</html>
```

simpan program di atas dengan nama coba.html, selanjutnya jalankan melalui web browser yang ada.

Selain menggunakan web browser yang mendukung applet Java, file html di atas dapat juga dijalankan melalui appletviewer yang telah tersedia dalam paket JDK melalui perintah:

```
appletviewer coba.html
```

Penjelasan:

Kelas cobaApplet yang dibuat di atas terdiri atas sebuah objek (g) yang dibuat dari kelas Graphics yang terdapat pada paket java.awt. Agar dapat digunakan, maka kelas Graphics harus di sertakan dalam program melalui pernyataan: **`import java.awt.Graphics;`**

Kelas di atas terdiri atas sebuah method, yaitu paint() yang menindih method paint() yang ada dalam kelas Applet. Method ini memiliki sebuah argumen berupa objek Graphics, dinyatakan oleh g. Kelas Graphics memiliki method drawString() yang dapat menampilkan deretan karakter (string) di dalam jendela applet. String ini ditampilkan dari koordinat (10,10). Untuk dapat membayangkan posisi titik (10,10), perlu diketahui bahwa layar bertipe SVGA memiliki resolusi sebesar 1024x768 piksel.

Applet tidak dijalankan langsung oleh interpreter java (dengan perintah **`java namakelas`** seperti yang dilakukan untuk aplikasi yang dibuat dalam modul sebelumnya), namun disisipkan dalam file html melalui tag **`<applet> </applet>`**, dimana file html inilah yang selanjutnya akan dijalankan melalui web browser. Perhatikan bahwa antara file html yang memanggil kelas applet yang dibuat dengan kelas appletnya harus berada pada folder yang sama. Nilai **`width`** dan **`height`** pada tag applet menyatakan ukuran dari jendela

appletnya.

Grafis dalam Java

Kelas Graphics yang tersedia dalam paket java.awt memiliki beberapa method untuk menggambar pada applet yang dibuat. Method-method untuk menggambar akan meminta argumen berupa posisi titik awal, titik akhir dan sudut pada applet. Titik (0,0) akan merujuk ke posisi di pojok kiri atas pada layar jendela applet.

Beberapa method yang tersedia untuk menampilkan grafis 2 dimensi di antaranya adalah sebagai berikut:

Garis:

dinyatakan dengan method **`drawLine(x_awal, y_awal, x_akhir, y_akhir)`**

contoh:

```
import java.awt.Graphics;
public class AppletGaris extends java.applet.Applet {
      public void paint(Graphics g){
            g.drawLine(20,20,100,100);
      }
}
```

Segi empat:

dinyatakan dengan method **`drawRect(x,y,lebar,tinggi)`** yang akan menggambar bentuk segi empat berbentuk rangka (bidangnya tidak diisi). Dapat juga dinyatakan sebagai **`fillRect(x,y,lebar,tinggi)`** yang akan menggambar bentuk segi empat yang bidangnya diisi. X dan y merupakan posisi sudut kiri atas dari bentuk segi empatnya.

Contoh:

```
import java.awt.Graphics;
public class AppletSegiempat extends java.applet.Applet {
      public void paint(Graphics g){
            g.drawRect(20,20,100,100);
            g.fillRect(200,20,100,100);
      }
}
```

Oval

dinyatakan dengan method **`drawOval(x,y,lebar,tinggi)`** untuk oval tanpa diisi dan **`fillOval(x,y,lebar,tinggi)`** untuk oval yang diisi. X dan y menyatakan sudut kiri atas dari kotak semu yang mengelilingi oval tersebut (bayangkan oval dibentuk dari kotak persegi lalu sudut-sudutnya dilengkungkan), sedangkan lebar dan tinggi menyatakan lebar dan tingginya kotak semu tersebut. Untuk nilai lebar dan tinggi yang sama, akan diperoleh bentuk lingkaran.

Contoh:

```
import java.awt.Graphics;
public class AppletOval extends java.applet.Applet {
      public void paint(Graphics g){
            g.drawOval(20,20,150,100);
            g.fillOval(200,20,100,100);
      }
}
```

Lengkung

dinyatakan dengan method

`drawArc(x,y,lebar,tinggi,sudutMulai,derajatLengkung)` untuk garis lengkung yang tidak terisi dan **`fillArc(x,y,lebar,tinggi,sudutMulai,derajatLengkung)`** untuk garis lengkung yang terisi. X dan y memiliki konsep nilai yang sama seperti ketika menggambar bentuk oval.

Contoh:

```
import java.awt.Graphics;
public class AppletOval extends java.applet.Applet {
      public void paint(Graphics g){
            g.drawArc(20,20,150,100,90,180);
            g.fillArc(200,20,100,100,30,330);
      }
}
```

MODUL 8: APPLET ANIMASI

Kerangka utama dari program untuk applet animasi adalah sebagai berikut:

```
import java.applet.*;
import java.awt.*;

public class nama_applet extends Applet implements Runnable {

//definisikan variabel
Thread Runner;
int xpos;
.....

//definisikan fungsi start()
public void start() {
if (runner==null) {
        runner = new Thread(this);
        runner.start();
        }
}

//definisikan fungsi stop()
public void stop() {
if (runner !=  null){
        runner.stop();
        runner = null;
        }
}

//definisikan fungsi run()
//jalannya animasi berada di method ini
public void run() {
```

```
        .....
        .....
}

//definisikan bentuk gambar disini
public void paint(Graphics g){
        .....
        .....
        }
}
```

Perhatikan contoh program berikut:

```
//contoh applet animasi dengan Java

import java.awt.*;
import java.applet.*;

public class animasi extends java.applet.Applet implements Runnable
{
      Thread runner;
      int xpos;
      int ux;

public void start() {
//    if (runner == null){
            runner = new Thread(this);
            runner.start();
//    }
}

public void stop() {
//    if (runner != null)
//          runner.stop();
```

```
      runner = null;

}

public void run() {
      while (true) {
            for (xpos = 5; xpos<=105; xpos+=5){
                  ux = xpos;
                  repaint();
                  try { Thread.sleep(100); }
                  catch (InterruptedException e){ }
                  }
            }
}

public void paint(Graphics g){
      update(g);
}

public void update(Graphics g){
      g.setColor(Color.white);
      g.fillRect(0,0,300,300);
      g.setColor(Color.black);
      g.fillRect(xpos,20,20,20);
}
}
```

MODUL 8: APPLET (LANJUTAN)

Pengaturan Font Dalam Applet

Pada modul sebelumnya, telah dipelajari method **`drawString()`** yang terdapat dalam kelas Graphics untuk mencetak string ke dalam applet. Font yang digunakan masih menggunakan font default yang disediakan oleh kelas Graphics. Font tersebut dapat diatur sesuai dengan yang dibutuhkan dengan menggunakan kelas Font yang terdapat pada paket awt. Terlebih dahulu, objek font harus diciptakan sebelum digunakan untuk mengatur font dari string yang akan dicetak pada applet. Perhatikan contoh berikut:

```
import java.awt.*;

public class cobafont extends java.applet.Applet {
     public void paint(Graphics g){
          //menciptakan objek font
          Font ft = new Font("TimesRoman",Font.PLAIN,20);
          Font fa = new Font("Arial",Font.BOLD,20);
          Font fc = new Font("Courier",Font.ITALIC,20);
          Font f  = new Font("Verdana",Font.ITALIC + Font.BOLD,20);

          //mencetak String di applet
          g.setFont(ft);
          g.drawString("Ini TimesRoman 20 pt",5,25);
          g.setFont(fa);
          g.drawString("Ini Arial 20 pt cetak tebal",5,45);
          g.setFont(fc);
          g.drawString("Ini Courier 20 pt cetak miring",5,65);
          g.setFont(f);
          g.drawString("Ini TimesRoman 20 pt cetak tebal dan
miring",5,85);
     }
}
```

Jalankan applet di atas dengan memanggilnya melalui file html dan pelajari bagaimana langkah untuk mengatur font yang akan dicetak ke layar applet.

Pengaturan Warna

Selain font, warna string maupun warna bentuk geometri yang dicetak di layar applet dapat diatur. Pengaturan warna ini dilakukan melalui method setColor() yang akan meminta masukan objek warna. Sebagai contoh, pernyataan:

```
g.setColor(Color.warna)
```

akan mencetak string maupun gambar dengan warna yang sesuai dengan kode warna. Kode warna standar yang disediakan kelas Color adalah: **`white`**, **`black`**, **`gray`**, **`red`**, **`green`**, **`blue`**, **`yellow`**, **`orange`**, **`pink`**, **`magenta`**, **`cyan`**, **`lightGray`** dan **`darkGray`**. Perhatikan contoh berikut:

```
import java.awt.*;

public class cobawarna extends java.applet.Applet {
     public void paint(Graphics g){

          Font ft = new Font("TimesRoman",Font.BOLD,20);
          g.setFont(ft);
          g.setColor(Color.red);
          g.drawString("Ini Kotak",20,25);
          g.setColor(Color.green);
          g.fillRect(10,50,100,100);
     }
}
```

Antarmuka Pengguna (User Interface)

Paket awt yang disediakan dalam Java juga berisi kelas komponen yang dapat digunakan untuk membangun sebuah antarmuka pengguna (*user interface*), yang antara lain berupa jendela/frame, menu, tombol, checkbox, scrollbar, list, label, textfield beserta pengelolaan *event*. Untuk menambahkan komponen antarmuka pengguna ini ke dalam

applet, digunakan method **`add()`**.

Label

merupakan string teks yang tidak dapat diedit dan digunakan untuk memberikan informasi kepada pengguna mengenai suatu komponen. Format dari label adalah **`Label(string)`** dimana string merupakan kumpulan karakter yang muncul sebagai label pada applet.

```
import java.awt.*;

public class cobalabel extends java.applet.Applet {
     public void init(){
          setLayout(new FlowLayout());
          setFont(new Font("TimesRoman",Font.ITALIC,16));
          add(new Label("ini label1",Label.LEFT));
          add(new Label("ini label2",Label.CENTER));
          add(new Label("ini label3",Label.RIGHT));
     }
}
```

Method **`setLayout()`** digunakan untuk menentukan bagaimana komponen yang di add-kan tampil di dalam applet. Salah satu pilihannya adalah **`FlowLayout()`** dimana setiap penambahan komponen akan diletakkan di samping komponen sebelumnya, dan jika melewati batas ukuran applet akan diletakkan pada baris berikutnya.

Button (Tombol)

merupakan komponen yang akan memicu suatu aksi ketika di*klik*. Label yang ada pada button dinyatakan dalam string yang disisipkan pada pernyataan **`Button(label)`**.

```
import java.awt.*;

public class cobatombol extends java.applet.Applet {
     public void init(){
          setLayout(new FlowLayout());
          setFont(new Font("TimesRoman",Font.ITALIC,16));
```

```
            add(new Button("Ok"));
            add(new Button("Batal"));
            add(new Button("Tidak"));
      }
}
```

Check Box

merupakan komponen antarmuka pengguna yang digunakan untuk memilih atau tidak memilih suatu keadaan. Format dari check box adalah **`Checkbox(String)`**. Jika diinginkan agar suatu keadaan dipilih atau tidak dipilih secara default, digunakan format **`Checkbox(String, null, boolean)`** dengan boolean dinyatakan **`true`** jika terpilih dan **`false`** jika tidak terpilih. Apabila tidak dinyatakan, maka nilai defaultnya adalah **`false`**.

```
import java.awt.*;

public class cobacheckbox extends java.applet.Applet {
      public void init(){
            setLayout(new FlowLayout());
            setFont(new Font("TimesRoman",Font.ITALIC,16));
            add(new Checkbox("Matematika"));
            add(new Checkbox("Fisika",null,true));
            add(new Checkbox("Kimia"));
      }
}
```

Radio Button

mirip dengan Check Box, yang membedakan adalah: pada Check Box beberapa keadaan dapat dipilih pada saat yang sama, sedangkan pada Radio Button keadaan yang terpilih hanya dibatasi satu saja. Untuk menampilkan Radio Button, sebelumnya objek/instance dari CheckboxGroup harus diciptakan terlebih dahulu, selanjutnya komponen Radio Button dinyatakan dengan format **`Checkbox(String, namaObjek, boolean)`**.

```
import java.awt.*;
```

```
public class cobatombolradio extends java.applet.Applet {
     public void init(){
          setLayout(new FlowLayout());
          setFont(new Font("TimesRoman",Font.ITALIC,16));
          CheckboxGroup pilihan = new CheckboxGroup();
          add(new Checkbox("Matematika",pilihan,false));
          add(new Checkbox("Fisika",pilihan,true));
          add(new Checkbox("Kimia",pilihan,false));
     }
}
```

Menu Pilihan

merupakan komponen berbentuk pull-down yang digunakan untuk memilih satu dari beberapa pilihan yang disediakan. Mirip dengan Radio Button, sebelum menambahkan pilihan pada menu dengan method **`addItem(String)`**, objek/instance dari menu pilihan harus diciptakan terlebih dahulu.

```
import java.awt.*;

public class cobamenupilih extends java.applet.Applet {
     public void init(){
          setFont(new Font("TimesRoman",Font.ITALIC,16));
          Choice pilihan = new Choice();
          pilihan.addItem("Matematika");
          pilihan.addItem("Fisika");
          pilihan.addItem("Kimia");
          add(pilihan);
     }
}
```

Text Field

merupakan komponen antarmuka pengguna yang digunakan untuk meminta masukan. Text Field seolah merupakan kebalikan dari Label dimana jika label menampilkan suatu string pada applet, Text field meminta masukan melalui applet. Format Text Field adalah

TextField(lebarfield) dimana **lebarfield** diisi nilai maksimum karakter minimum yang terlihat.

```
import java.awt.*;

public class cobatext extends java.applet.Applet {
     public void init(){
          // setLayout(new FlowLayout());
          setFont(new Font("TimesRoman",Font.PLAIN,16));
          add(new Label("Masukkan nama :"));
          add(new TextField(30));
          add(new Label("Masukkan NIM:"));
          add(new TextField(10));
     }
}
```

MODUL 10: APPLET (LANJUTAN II)

Pengaturan Letak Komponen

Susunan komponen-komponen antar muka pengguna yang ditambahkan pada applet dapat diatur melalui method **`setLayout()`**. Terdapat beberapa pengaturan dasar yang disediakan oleh paket **awt**, diantaranya adalah **`FlowLayout()`**, **`GridLayout()`** dan **`GridBagLayout()`**. Untuk menyatakan pengaturan layout komponen pada suatu applet, harus diciptakan dahulu *instance* dari jenis pengaturan layout tersebut. Sebagai contoh, jika pengaturan yang dipilih adalah **`FlowLayout()`**, maka dinyatakan sebagai:

```
public void init() {
      setLayout(new FlowLayout());
}
```

pengaturan jenis **`FlowLayout()`** di atas akan menempatkan setiap komponen baru yang ditambahkan ke samping kiri dari komponen sebelumnya. Apabila komponen yang ditambahkan tidak dapat menempati baris yang ditempati komponen sebelumnya pada suatu applet, maka komponen baru tersebut akan ditambahkan pada baris berikutnya. Contoh programnya dapat dilihat pada modul sebelumnya.

Selain **`FlowLayout()`**, jenis pengaturan yang sering digunakan adalah **`GridLayout()`** dan **`GridBagLayout()`**:

GridLayout

Pengaturan jenis **`GridLayout()`** akan menempatkan komponen yang ditambahkan ke dalam susunan baris dan kolom dengan cacah sel yang sudah ditentukan. Sebagai contoh, jika dinyatakan **`GridLayout(3,2)`**, maka komponen akan disusun ke dalam grid berukuran 3x2 (3 baris dan 2 kolom). Perhatikan contoh program berikut:

```
import java.awt.*;

public class cobatombol extends java.applet.Applet {
     public void init(){
          setLayout(new GridLayout(3,2));
          setFont(new Font("TimesRoman",Font.ITALIC,16));
          add(new Button("Tombol_1"));
          add(new Button("Tombol_2"));
          add(new Button("Tombol_3"));
          add(new Button("Tombol_4"));
          add(new Button("Tombol_5"));
          add(new Button("Tombol_6"));
     }
}
```

GridBagLayout

Pengaturan dengan jenis **`GridBagLayout()`** akan memberikan susunan yang mirip dengan pengaturan **`GridLayout()`**, namun kita dapat mengatur ukuran dari sel yang berada dalam suatu grid yang dibuat. Pengaturan ukuran sel yang ditempati suatu komponen ini akan ditentukan oleh *constraint* (kekangan) yang didefinisikan untuk komponen tersebut. Variabel-variabel yang digunakan dalam constraint diantaranya adalah:

`int gridx, gridy` : untuk menentukan posisi komponen pada grid. **`gridx`** menentukan posisi pada kolom grid dan **`gridy`** menentukan posisi pada baris grid sehingga sel yang menempati bagian pojok kiri atas akan memiliki nilai (0,0) dan sel berikutnya pada baris yang sama akan memiliki nilai (1,0).

`int weightx, weighty` : untuk menentukan proporsi baris dan kolom yang ditempati oleh suatu komponen.

`int gridwidth, gridheight` : untuk menentukan jumlah baris dan kolom yang ditempati oleh suatu komponen.

`int fill`: untuk menentukan apakah komponen-komponen pada grid akan menempati semua ruang yang yang disediakan. Pilihannya adalah BOTH jika mengisi seluruh ruang applet, HORIZONTAL jika hanya ruang pada arah horisontal saja yang diisi dan VERTICAL jika yang diisi hanya ruang pada arah vertikal saja.

Ketika menggunakan **`GridBagLayout()`** sebagai pilihan pengaturan layout

komponen, penentuan nilai *constraint* harus dilakukan dengan cermat karena penentuan *constraint* untuk sebuah komponen akan mempengaruhi komponen yang lain sehingga layout yang diperoleh terkadang justru tidak sesuai dengan yang diharapkan. Untuk itu akan mempermudah jika terlebih dahulu dibuat sketsa dari tampilan susunan komponen pada applet.

Pada layout dengan jenis **`GridBaglayout()`**, sebelum komponen ditambahkan pada applet dengan method **`add()`**, nilai-nilai *constraint* ditentukan terlebih dahulu melalui sebuah method bantu yang digunakan untuk menentukan bagaimana bentuk deklarasi dari nilai *constraint* suatu komponen yang ditambahkan. Perhatikan contoh program berikut :

```
import java.awt.*;

public class cobagrid extends java.applet.Applet {
    GridBagConstraints konstrain = new GridBagConstraints();

    void addGB(Component komponen, int x, int y) {
      konstrain.gridx = x;
      konstrain.gridy = y;
      add(komponen, konstrain);
    }

   public void init() {
      setLayout(new GridBagLayout());
      konstrain.weightx = 1.0;
      konstrain.weighty = 1.0;
      konstrain.fill = GridBagConstraints.BOTH;
      int x, y;
      addGB(new Button("atas"), x=1, y=0);
      addGB(new Button("kanan"), x=0, y=1);
      addGB(new Button("tengah"), x=1, y=1);
      addGB(new Button("kiri"), x=2, y=1);
      addGB(new Button("bawah"), x=1, y=2);
   }
}
```

Pada program di atas, method **addGB()** digunakan sebagai method bantu untuk menentukan bentuk deklarasi penentuan nilai *constraint* untuk komponen yang ditambahkan. Pada method tersebut, hanya ditentukan constraint untuk menentukan posisi komponen dalam grid saja. Kita dapat menambahkan nilai constraint yang lain, misal seberapa banyak suatu komponen menempati baris dan kolom pada grid melalui variabel **gridheight** dan **gridwidth** seperti pada contoh program di berikut. Perhatikan bahwa program di bawah hanyalah modifikasi dari program sebelumnya.

```
import java.awt.*;

public class cobagrid extends java.applet.Applet {
    GridBagConstraints konstrain = new GridBagConstraints();

    void addGB(Component komponen, int x, int y, int w, int h) {
      konstrain.gridx = x;
      konstrain.gridy = y;
      konstrain.gridwidth = w;
      konstrain.gridheight= h;
      add(komponen, konstrain);
    }

   public void init() {
      setLayout(new GridBagLayout());
      konstrain.weightx = 1.0;
      konstrain.weighty = 1.0;
      konstrain.fill = GridBagConstraints.BOTH;
      int x, y, w, h;
      addGB(new Button("atas"), x=0, y=0, w=3, h=1);
      addGB(new Button("kanan"), x=0, y=1, w=1, h=2);
      addGB(new Button("tengah"), x=1, y=1, w=1, h=1);
      addGB(new Button("kiri"), x=2, y=1, w=1, h=1);
      addGB(new Button("bawah"), x=1, y=2, w=2, h=1);
   }
}
```

Penanganan Aksi

Aksi yang dilakukan pada suatu komponen, misal dengan diklik, diketik atau disorot, akan membangkitkan event. Untuk mendeteksi event, terdapat dua cara yang dapat dipilih:

Menggunakan method Action()

Event dideteksi melalui method **`action()`** dengan algoritma berikut:

```
public boolean action(Event e, Object arg) {
   // aktivitas komponen

   if (e.target instanceOf TextField) {
      String isian = mytextField.getText();
      ...
      return true;
   } else return false;
   if (e.target instanceOf Button) {
       ...
      return true;
   } else return false;
}
```

Pada format di atas, diciptakan objek e yang merupakan objek event yang akan mewakili terjadinya event, sedangkan objek arg yang nerupakan objek argumen akan ditentukan berdasarkan komponen yang sesuai:

- Untuk Button (tombol), aksi terjadi ketika diklik. Argumennya adalah string yang menjadi label tombol tersebut.
- Untuk Checkbox, aksi terjadi ketika box dicentang. Argumennya selalu true.
- Untuk Choice, aksi terjadi ketika salah satu pilihan diklik. Argumennnya adalah string label pada item yang tersedia.
- Untuk TextField, aksi terjadi ketika tombol <enter> ditekan.

Menggunakan antar muka ActionListener

Event dideteksi dengan sebelumnya mengimplementasikan **`ActionListener`** pada applet yang dibuat. Selanjutnya setiap komponen yang akan dideteksi ditambahkan method **`addActionListener()`**.

Pelajari contoh program berikut yang menerapkan cara kedua untuk menangani *event* :

```
// program konversi suhu Celcius ke Fahrenheit atau Kelvin

import java.awt.*;
import java.awt.event.*;

public class konversi extends java.applet.Applet implements
ActionListener {
      TextField celcius = new TextField(8);
      String satuan="Suhu Konversi :";
      double suhu = 0.0;
      Label konversi = new Label(satuan);
      Label nilai = new Label(String.valueOf(suhu));
      Button tombolF = new Button("Fahrenheit");
      Button tombolK = new Button("Kelvin");

   public void init() {
      setLayout(new GridLayout(3,2));
      setFont(new Font("TimesRoman",Font.BOLD,14));
      add(new Label("Suhu Celcius :"));
      add(celcius);
      add(tombolF);
      tombolF.addActionListener(this);
      add(tombolK);
      tombolK.addActionListener(this);
      add(konversi);
      add(nilai);
   }
```

```
   public void actionPerformed(ActionEvent e) {
      double c = Double.parseDouble(celcius.getText());
      double f = 9.0/5.0*c + 32.0;
      double k = c + 273.0;
      if (e.getSource() == tombolK) {
         suhu = k;
         satuan= "Suhu Kelvin :";
         }
      if (e.getSource() == tombolF) {
         suhu = f;
         satuan= "Suhu Fahrenheit :";
         }
      repaint();
   }

   public void paint(Graphics g) {
      konversi.setText(String.valueOf(satuan));
      nilai.setText(String.valueOf(suhu));
   }
}
```

Tugas

Buatlah program kalkulator sederhana yang memiliki dua masukan (melalui komponen TextField), kemudian tombol operasi aritmatika (+, -, x, /) dan tampilan hasil melalui komponen Label.