DIKTAT

PEMROGRAMAN BERORIENTASI OBJEK
(Edisi Revisi I)

Oleh :
ARMANSYAH, M.KOM
NIDN. 2004108401

PROGRAM STUDI ILMU KOMPUTER
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SUMATERA UTARA
MEDAN
2019

KATA PENGANTAR

Puji syukur kepada Allah atas limpahan rahmatNya kepada punulis, hingga akhirnya penulis dapat merampungkan bahan ajar mahasiswa dalam bentuk modul ajar atau diktat. Modul ini dirancang untuk membantu mahasiswa dalam memahami konsep dan praktis algoritma pemrograman komputer secara umum, dan secara khusus Pemrograman Berorientasi Objek.

Modul ini disusun dari dasar hingga akhir sesuai dengan silabus yang dirancang untuk memperkaya referensi mahasiswa dalam memahami konsep dan praktis pemrograman berorientasi objek. Tentu saja keberadaan modul akan menjadi sangat bermanfaat jika mahasiswa membaca dan menerapkannya secara mandiri diluar perkuliahan. Harapan penulis dengan dirancangkannya modul ini adalah tercapainya kompetensi pengetahuan dan praktis pemrograman berorientasi objek oleh mahasiswa, karenanya praktek mandiri harus lebih diperbanyak untuk memenuhi capaian pembelajaran.

Penulis juga berterimakasih kepada Dekan Fakultas Sains dan Teknologi UIN Sumatera Utara Medan, secara khusus ketua Program Studi Ilmu Komputer atas dukungan dan bimbingannya, serta rekan dosen yang tidak disebutkan satu-persatu,--sehingga penulis bersemangat dalam melakukan tugas penulisan modul ini. Tentu saja ucapan terimakasih tidak lupa penulis sampaikan kepada istri karena telah begitu banyak membantu dan menemai malam-malam penulisan modul ini. Kepada seluruh mahasiswa dan pembaca, terimakasih apresiasinya atas menggunakan modul ini.

Rancangan modul ini mungkin terlalu sederhana dan banyak kekurangan baik dari sisi konten, struktur penulisan hingga modelnya. Pengembangan dan revisi akan terus diupayakan untuk meningkatkan kualitas dan mutunya dimasa mendatang. Akhirnya kepada pembaca penulis mohon maaf atas segala kekurangan, dan kepada Allah penulis mohon ampun. Demikian pengantar ini, penulis sampaikan *alhamdulillahirabbil'alamiin. Wassalamu'alaikum warahmatullahi wabarakaatuh.*

Medan, Juni 2019

Armansyah, M.Kom

DAFTAR ISI

MODUL 1

PENGANTAR BAHASA JAVA

1.1 Sejarah Singkat Java

Ide pengembangan Java sebenarnya dimulai dari 1991 atas usulan dari Patrick Naugthon, seorang insinyur di Perusahaan Sun Microsystems. Dia mengusulkan sebuah proyek kepada temannya dan CEO Scott McNealy untuk pengembangan lingkungan komputer kecil, sederhana, portabel yang dapat mengendalikan semua jenis komponen elektronik (TechMetrix Research, 1999:1). Sepanjang pengembangannya, pada tahun 1995 melalui James Gosling di bawah bendera Sun Microsystems Java mulai dikenal publik. Proyek pengembangan Java kemudian dirilis dengan menggunakan komponen inti dari platform Java Sun Microsystems. Pada era tersebut Java diberi label dengan penamaan J2SE (Java 2 Standard Edition) versi 1.0. Pada rilis terbaru (saat ini) penamaan J2 berganti nama dengan Java SE untuk versi standar, Java EE untuk versi perusahaan dan Java ME untuk perangkat bergerak dan seluler. Bahasa Java berjalan di hampir semua sistem operasi seperti Windows, Mac OS, dan berbagai versi UNIX.

Pada tahun 1997 Java Development Kit (JDK) memperbaiki beberapa masalah pada mesin kompilernya dan meningkatkan versinya menjadi versi 1.1. Pada tahun tersebut OMG (Object Management Group) yang mengatur dan menstandarisasi pemrograman berorientasi objek, resmi mengakuinya dengan Sun Microsystems sebagai satu-satunya perusahaan pemasok perangkat-perangkat Java.

Saat modul ini ditulis, versi Java saat ini sudah sampai pada versi 12 (Java SE 12.0.2). Untuk melihat dan mengunduh Java SE, pengguna dapat mengunjungi situs resmi oracle. Dalam pengembangannya hingga perilisan versi pertama, Java terus dikembangkan hingga popularitasnya meluas bukan hanya penggunaanya sebagai pemrograman komputer, tetapi juga sebagai pemrograman komponen-komponen elektronik lainnya. Konfigurasi-konfigurasi dilakukan untuk menyesuaikan Java dengan berbagai jenis platform. Sehingga Java memiliki spesifikasi tersendiri berdasarkan tujuan umum perancangannya, seperti J2EE (Java 2 Enterprise Edition) untuk Aplikasi Perusahaan, J2ME (Java 2 Micro Edition) untuk Aplikasi Seluler dan perangkat sejenisnya, dan beberapa perangkat lunak Java lainnya yang memiliki kekuatan spesifik dan disesuaikan untuk proyek-proyek spesifik juga.

Perkembangan Java mulai awal dikembangkan sangat positif dan terus berkembang. Hingga saat ini kepopuleran Java nyaris tidak terkalahkan, hal ini terlihat dari survey Java

yang cenderung stabil setiap periode survey. Hal ini tampak pada laporan dari lembaga yang aktif dalam menyoroti perkembangan berbagai bahasa pemrograman. Dari situs resminya itu laporan pada Septembe 2019 ini terlihat Java menempati urutan pertama dan tidak berbeda pada periode yang sama ditahun sebelumnya, dengan perolehan 16.661% dari seluruh jenis pemrograman yang di survey (tiobe.com, 17/9/2019).

1.2 Prospek Bahasa Java

Jika mengamati sejarah perkembangan Java hingga saat ini, prospek Java tampak terus berkembang dan akan tetap digunakan terutama oleh pengembang perangkat lunak. Sehingga, penulis menyarankan kepada para pemula untuk memulai menyukai dan mempelajari Java dengan tuntas. Selain prospeknya, ada beberapa hal yang perlu menjadi perhatian mengapa pemula (dalam hal ini adalah calon progammer) untuk memilih Java sebagai bahasa yang harus dipelajari. Beberapa hal yang dimaksud diantaranya :

1) Bahwa Java merupakan bahasa pemrograman berorientasi objek sehingga mudah untuk memperluas aplikasi menjadi yang kompleks dan besar.
2) Java dijalankan oleh Mesin Virtual *(Java Virtual Machine)* sehingga keberadaanya tidak tergantung kepada sistem operasi yang mengggunakannya, tidak seperti bahasa pemrograman lainnya.
3) Java merupakan pemrograman yang sederhana, bergantung kepada pemahaman terhadap konsep dasar objek.
4) Bahasa Java merupakan bahasa yang aman terhadap serangan virus. Ini karena sifat teknik otentikasinya didasarkan pada enkripsi kunci publik.
5) Bahasa Java merupakan Bahasa dengan arsitektur netral. Hal ini karena kehadiran *Java Runtime* yang memungkinkan kompiler java dapat menghasilkan format *file* objek yang dapat dieksekusi pada banyak prosesor.
6) Bahasa Java mendukung prinsip *multithreaded* dimana aplikasi yang dikembangkan dengan Java lebih interaktif karena dapat melakukan banyak tugas.

Selain keenam hal tersebut, hal-hal lain yang perlu diperhatikan terkait Java diantaranya bahwa Java juga merupakan bahasa yang portabel, kuat, mudah di-interpretasikan, berkinerja tinggi, terdistribusi, dan dinamis.

1.3 Editor

Setiap bahasa pemrograman membutuhkan *tools* (alat bantu) untuk menulis kode program, tidak terkecuali dengan pemrograman Java. Setiap editor tentunya membutuhkan

spesifikasi *hardware* dan *software* operasi yang sesuai dengan versi Java yang digunakan. Dalam modul ini penulis menggunakan spesifikasi *hardware* dengan prosesor Amd A9-9420 Radeon R5 3.00 GHz, 4 GB Memory (RAM), serta sistem operasi Windows 10 64 bit.

Bahasa Java sangat mudah ditulis, ada banyak editor yang dapat diunduh secara gratis di Internet. Beberapa editor yang ringan seperti Notepad dan Notpad++ dapat digunakan. Untuk mempermudah penulisan program, modul ini merekomendasikan editor diantaranya sebagai berikut ini :

1) Netbean
2) Eclipse
3) Jcreator, dan lain-lain.

Selain editor diatas, penuliskan program Java juga harus menggunakan kompiler java yang umum disebut dengan JDK. Pengguna dapat memilih sesuai kebutuhan, namun dalam praktikum di modul ini, penulis menggunakan JDK 8.

1.4 Instalasi Java

Seperti yang dikatakan diawal bahwa piranti-piranti Java dapat diunduh secara bebas gratis disitus resmi Oracle. Untuk mengunduhnya pengguna dapat membuka situs https://www.oracle.com/technetwork/java/javase/overview/index.html. Pengguna dapat mengunduh versi Java SE dengan versi yang sesuai dengan spesifikasi *hardware* dan *software* sistem operasi yang dimilikinya. Untuk Java versi lama pengaturan mesin Java membutuh konfigurasi path untuk pembacaan file java. Namun saat ini konfigurasi yang demikian tidak terlalu penting. Secara umum file mesin java akan secara otomatis terinstal pada direktory ‘C:\Program Files\java\jdk’. Jika konfigurasi dibutuhkan, lakukan pengaturan dengan mengikuti langkah-langkah berikut :

1) Klik kanan pada *'My Computer'* atau ‘*This PC’* dan pilih *'Properties'*.
2) Klik “*Advanced System Settings*”
3) Klik tombol *'Environment Variables*..” di bawah tab “*Advanced*”
4) Selanjutnya, ubah variabel *'Path'* sehingga juga berisi path ke *executable* Java. Contoh, jika *path* saat ini diatur ke 'C:\WINDOWS\SYSTEM32', maka ubah path kamu untuk membaca 'C:\WINDOWS\SYSTEM32; C:\Program Files\java\jdk\bin'.

Untuk penggunaan editor seperti Netbeans atau Eclipse saat ini, lokasi *path* kadangkala sudah terkonfigurasi secara otomatis. Sehingga pengguna tidak perlu melakukan konfigurasi-konfigurasi.

1.5 Struktur Bahasa Java

Struktur adalah ibarat rumus, formula atau bentuk dasar yang tidak boleh dilanggar. Setiap pemrograman memiliki struktur dasar, begitu juga dengan Bahasa Java. Struktur Java, jika melihat bentuknya mengadopsi struktur bahasa C/C++. Hal ini karena memang bahasa Java merupakan turunan dari bahasa C dan C++. Adapun struktur bahasa Java mengacu kepada struktur dasar *(basic syntax)* berikut :

```
class namakelas{
      public static void main(String[] args) {
      //statemen atau ekspresi program
      }
}
```

Setiap bahasa Java harus diawali dengan *keyword* **class,** dimana kata class mendefenisikan sesuatu (namakelas) sebagai tempat yang menggambarkan perilaku atau keadaan yang didukung oleh objek dan jenisnya. Sementara **namakelas** adalah nama class dari suatu program. Kode program Java disimpan dengan format **.java** yang kadangkala harus sama dengan nama kelas yang dimiliki. Setiap kode program java setidaknya harus memiliki satu **class** dan satu **method** yang disebut dengan *main* method. *Main method* umumnya berfungsi untuk menerima ekspresi-ekspres yang terbentuk pada class lain atau method lain di class yang sama.

Baik class maupun method sama-sama diawali dengan tanda kurung kurawal buka { dan diakhiri dengan kurung kurawal tutup }. Untuk lebih jelasnya silahkan amati struktur dasar bahasa Java pada kotak yang telah disajikan diatas.

MODUL 2

TIPEDATA DAN OPERATOR

2.1 Variabel

Pembahasan modul ini akan dimulai dengan pembahasan variabel, mengingat tipedata selalu terkait dengan variabel. Variabel adalah ibarat sebuah wadah yang memiliki batas kemampuan untuk menampung benda-benda didalamnya. Setiap benda tentu memiliki sifat yang berbeda pula, seperti misalnya air, batu, pasir, gas dan lain sebagainya, memiliki sifat berbeda dan wadah yang mereka gunakan untuk menampung keberadaan mereka juga berbeda kemampuan atau kapasitasnya.

Ilustrasi wadah itulah yang mewakili defenisi variabel. Dalam pemrograman komputer, variabel adalah tempat penyimpanan data yang tidak permanen dan selalu berubah. Penggunaanya memiliki aturan yang hampir sama pada setiap pemrograman. Dalam pemrograman Java, penulisan variabel mengacu pada standar penulisan berikut :

1) Variabel ditulis dengan huruf kapital saja, atau huruf akik (huruf kecil) saja,
2) Jika terdiri dari dua suku kata atau lebih, pisahkan dengan tanda garis bawah (_), dan tidak dibenarkan menggunakan spasi,
3) Tidak membenarkan awalan dalam bentuk angka dan simbol-simbol %, &, @, !, ^, #, *, (, dan,), serta beberapa simbol-simbol lain yang terlarang,
4) Tidak dibenarkan penggunaan variabel dari *keyword* yang dimiliki oleh Java

Jika dilihat dari ketentuan diatas maka penulisan variabel pada bahasa JAVA yang absah adalah sebagai berikut :

NAMA VARIABEL	KEABASAHAN
nm atau _nm	Benar
nm_siswa	Benar
NAMA	Benar
Nama karyawan	**Salah**
nm123	Benar
%kode atau 2nama	**Salah**

Penulisan variabel pada pemrograman java mirip dengan penulisan pada bahasa C, yaitu dengan menyertakan tipedata diawal nama variabel. Aturan penulisan atau bentuk umum penulisan variabel tipedata pada bahasa java adalah sebagai berikut :

```
tipe_data nama_variabel;
```

Atau dengan inisialisasi nilai dengan struktur

```
tipe_data nama_variabel = nilai;
```

Contoh

```
int bilangan;

int bilangan = 7;

float lebar = 4.5;
```

suatu variabel kadangkala dibentuk secara konstanta, dengan demikian nilai variabel tersebut tidak dapat diubah. Variabel yang diatus secara konstanta dikenalkan dengan *modifier* **final** diawal penulisan variabel, sehingga penulisan variabel konstanta ditulis lengkap dengan merujuk kepada struktur berikut :

```
final tipedata nama_variabel = nilai;
```

contoh

```
final double phi=3.14;
```

2.2 *Keywords*

Keywords merupakan daftar kata-kata kunci yang dimiliki oleh suatu bahasa pemrograman dan disimpan didalam *library*, dimana *keyword* tersebut telah dikenali oleh kompiler dan seringkali dilarang untuk digunakan kembali dalam penentuan *indentifier* (variabel, method atau fungsi) baru. Dalam bahasa java daftar *keywords* diantaranya dituliskan pada tabel berikut :

abstract	assert	boolean	Break	byte	transient
case	catch	char	Class	const	try
continue	default	do	Double	else	void
enum	extends	final	Finally	float	volatile
for	goto	if	implements	import	while

instanceof	int	interface	Long	native	switch
new	package	private	Protected	public	this
return	short	static	Strictfp	super	throw

Selain daftar *keyword* diatas, masih banyak *keyword* lain yang tidak disampaikan di sana. Pengguna bahasa java secara langsung akan menjumpainya pada saat menulis kode program.

2.3 Tipe Data

Tipe data merupakan set (rentang) nilai yang memiliki karakteristik serupa. Tipedata memiliki kemampuan dan kekurangan yang berbeda untuk menangani data. Secara umum tipedata java terdiri dari tipe bernilai (value type), tipe referensi dan tipe pointer. Tipe data tersebut dapat menerima data berupa karakter tunggal maupun majemuk, bilangan bulat, bilangan pecahan dan tipe data logika. Secara lengkap tipe data dalam java dapat dilihat pada tabel dibawah berikut ini :

Tipedata	Nilai Default	Nilai Minimum	Nilai Maksimum	Ukuran
byte	0	-128 (-2^7)	127 (2^7-1)	1 byte
short	0	-32768 (-2^15)	32767 (2^15-1)	2 byte
int	0	-2147483648 (-2^31)	2147483647 (2^31-1)	4 byte
long	0L	-9223372036854775808 (-2^63)	9223372036854775807 (2^63-1)	8 byte
float	0.0f	IEEE 754 bilangan titik mengambang		4 byte
double	0.0d	IEEE 754 bilangan titik mengambang		8 byte
Boolean	false	True atau False		1 bit
char	'\u0000'	'\u0000'	'\uffff'	2 byte

Untuk memahami penggunaan tipe data dan variabel, berikut ini disajikan contoh program dibawah ini :

```
public class VariabelTipe {
    public static void main(String args[]) {
        String strVariabel ="Helloo Java";
        int intVariabel = 100;
        float floatVariabel = 10.2f;
```

```
            char charVariabel = 'A';
            boolean boolVariabel = true;

        System.out.println(strVariabel+ ", untuk tipe string");
        System.out.println(intVariabel+ ", untuk tipe integer");
        System.out.println(floatVariabel+ ", untuk tipe float");
        System.out.println(charVariabel+ ", untuk tipe char");
        System.out.println(boolVariabel+ ", untuk tipe boolean");
       }
    }
```

Program diatas jika dieksekusi dan tidak menampilkan pesan kesalahan, akan menampilkan keluaran seperti berikut :

```
Helloo Java, untuk tipe string
100, untuk tipe integer
10.2, untuk tipe float
A, untuk tipe char
true, untuk tipe Boolean
```

2.4 Operator

Operator di dalam pemrograman, tidak terkecuali pada java merupakan simbol khusus yang digunakan untuk menentukan operasi tertentu. Operasi yang dimaksud bergantung kepada jenis operasi yang tuliskan ke program. Operasi itu dapat berupa operasi matematis, operasi perbandingan atau operasi penugasan, dan operasi-operasi lainnya. Operator bahasa java meliputi operator aritmatika, operator relasional, operator logika, operator bitwise, operator penugasan, dan operator *misc*.

2.4.1 Operator Aritmatika

Sesuai dengan namanya, operator ini digunakan untuk melakukan perintah operasi matematis. Operator ini terdiri dari beberapa operator yang dijelaskan pada tabel di bawah berikut contohnya :

Operator	Deskripsi	Contoh
+	Menambahkan dua operan	A + B = 30
-	Mengurangi operan pertama dengan operan kedua	A - B = -10
*	Mengalikan kedua operan	A * B = 200

/	Membagi pembilang dengan de-pembilang	B / A = 2
%	Mengambil sisa bagi operan pertama dari operan kedua	B % A = 0
++	Operator menaikan nilai integer satu poin	A++ = 11
--	Operator menurunkan nilai integer satu poin	A-- = 9

Perhatikan contoh berikut, umpama sebuah bangun ruang berbentuk kerucut yang memiliki tinggi *t* 20.30 cm, dan jari-jari *r* sebesar 15 cm memiliki volume yang dihitung dengan persamaan matematika $\frac{1}{3} \times \pi \times r^2 \times t$. Kita ketahui bahwa π *(phi)* adalah konstanta bilangan pecahan 3.14. Dengan keterangan tersebut maka volume kerucut untuk kasus tersebut adalah berjumlah 4780.65 cm^3.

Perhatikan apakah program java dapat diterapkan untuk menghitung volume kerucut tersebut. Untuk itu rancanglah program java dengan mengikuti kebutuhan variabel tipe yang sudah ada, yaitu variabel jari-jari dan tinggi, serta variabel *phi* yang bersifat konstanta. Amatilah rancangan program berikut :

```
      public class VolumeKerucut {
      public static void main(String[] args) {
          final double phi=3.14;
          double t=20.30;
          double r=15;
          double volume=(phi*r*r*t)/3;

          System.out.println("Tinggi kerucut    : "+ t);
          System.out.println("Jari-jari kerucut : "+ r);
          System.out.printf("Volume kerucut    : %.2f",
volume);
        }
}
```

Program tersebut menerapkan operator matematika dengan variabel konstanta *phi*. Jika dieksekusi dan tidak menampilkan kesalahan, akan menghasilkan keluaran seperti berikut :

```
Tinggi kerucut    : 20.3
Jari-jari kerucut : 15.0
Volume kerucut    : 4780.65
```

2.4.2 Operator Relasional

Operator pada kategori ini adalah operator yang berfungsi untuk membandingkan antar dua operan. Misal A benilai 5 dan B bernilai 7, untuk melihat penjelasan penggunaan operator ini terhadap A dan B, dapat dilihat pada tabel berikut :

Operator	Deskripsi	Contoh
==	Sama dengan	(A == B) Salah
!=	Tidak sama dengan	(A! = B) Benar
>	Lebih besar dari	(A> B) Salah
<	Lebih keci dari	(A <B) Benar
>=	Lebih besar dari atau sama dengan	(A> = B) Salah
<=	Lebih kecil dari atau sama dengan	(A <= B) Benar

Lebih lanjut akan disajikan pada contoh program berikut, perhatikanlah baik-baik kode program di bawah untuk mendapatkan pemahaman yang lebih baik :

```
      public class RelasiOperan{
      public static void main(String[] args) {
          int A=5, B=7;

          System.out.println("A == B = " + (A == B) );
          System.out.println("A != B = " + (A != B) );
          System.out.println("A > B = " + (A > B) );
          System.out.println("A < B = " + (A < B) );
          System.out.println("B >= A = " + (B >= A) );
          System.out.println("B <= A = " + (B <= A) );
          }
}
```

Program tersebut diatas akan menghasilkan keluaran sebagai berikut :

```
A != B = true
A > B  = false
A < B  = true
B >= A = true
B <= A = false
```

2.4.3 Operator Logika

Operator logika merupakan operator yang memiliki hasil keluaran bernilai benar (***true***) atau salah (***false***). Umpama variabel A bernilai true dan B bernilai false, maka perhatikanlah penjelasan dari penggunaan operator logika berikut terhadi variabel A dan B pada tabel berikut ini :

Simbol	Operator	Deskripsi	Contoh
&&	AND	Bernilai benar jika kedua operan bernilai benar	(A && B) Salah
||	OR	Bernilai benar jika satu operan bernilai benar	(A||B) Benar
!	NOT	Bernilai benar jika operan salah, dan sebaliknya	!(A && B) Benar

Perhatikan contoh berikut ini, dimana variabel A dan B diatur dengan tipe logika *boolean*. Dalam program variabel A diberi nilai *true* dan B diberi nilai *false.* Nilai *true* dan *false* merepresentasi nilai benar (atau biner 1) dan salah (biner 0). Lebih lanjut perhatikan kode program berikut :

```
      public class Logika{
        public static void main(String[] args) {
        boolean A = true;
        boolean B = false;
        System.out.println("A && B    = " + (A&&B));
        System.out.println("A || B    = " + (A||B) );
        System.out.println("!(A && B) = " + !(A && B));
        }
}
```

Keluaran program memiliki kesamaan persepsi seperti yang diuraikan pada tabel operator dan contohnya. Perhatikan keluaran program diatas setelah dieksekusi berikut ini :

```
A && B    = false
A || B    = true
!(A && B) = true
```

2.3.4 Operator Bitwise

Operator jenis ini mirip dengan operator logika, hanya saja penggunaan ini lebih spesifik untuk menangani data dengan nilai bilangan biner 1 dan 0. Adapun operator jenis ini

diantaranya, and, or, xor, not, left shift, dan right shift. Untuk lebih jelasnya perhatikan contoh penggunaanya misal A dan B adalah bilangan biner, maka operator ini akan bekerja seperti tabel dibawah ini :

A	B	A&B	A|B	A^B	~A	~B
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	1	1	1	0	0	0
1	0	0	1	1	0	1

Implementasi operator yang ditunjukkan pada tabel diatas mengacu pada penjelasan tabel berikut :

Simbol	Operator	Deskripsi
&	AND	Bernilai 1 jika kedua operan bernilai 1
|	OR	Bernilai 1 jika satu saja operan bernilai 1
^	XOR	Bernilai 1 jika kedua operan berbeda, dan sebaliknya akan bernilai o jika kedua operan bernilai sama
~	NOT	Bernilai 1 jika 0, dan sebaliknya bernilai 0 jika 1
<<	Left Shift	Nilai operan kiri dipindahkan ke kiri oleh jumlah bit yang ditentukan oleh operan kanan.
>>	Right Shift	Nilai operan kiri dipindahkan ke kanan dengan jumlah bit yang ditentukan oleh operan kanan.

2.3.5 Operator Penugasan

Operator ini digunakan untuk menugaskan operator menjalankan suatu operasi tertentu sesuai dengan jenis operasinya. Adapun tugas operator ini dijelaskan pada tabel berikut :

Operator	Deskripsi	Contoh	Keterangan
=	Memberikan nilai variabel kiri dari operasi di sebelah kanan	C = A + B	
+=	Memberikan nilai tambah kepada variabel kiri dari penjumlahan variabel kiri dengan variabel kanan	C += A	C = C + A

-=	Memberikan nilai kurang kepada variabel kiri dari pengurangan variabel kiri dengan variabel kanan	C -= A	C = C - A
*=	Memberikan nilai kali kepada variabel kiri dari perkalian variabel kiri dengan variabel kanan	C *= A	C = C * A
/=	Memberikan nilai bagi kepada variabel kiri dari pembagian variabel kiri dengan variabel kanan	C /= A	C = C / A
%=	Memberikan nilai sisa bagi kepada variabel kiri dari pembagian variabel kiri dengan variabel kanan	C % A	C = C % A

Selain penggunaan operator yang tersedia pada tabel diatas, operator penugasan juga berlaku pada operasi <<=, >>=, &=, ^=, dan |=. Serta operator penting lainnya seperti yang dikoleksi dalam operator Misc. Dimana operator Misc ini merupakan operator yang berfungsi untuk melihat ukuran dari tipedata yang dimiliki oleh java. Adapula operator yang disebut dengan operator ternary yang melibatkan beberapa operator sekaligus. Pada kesempatan lain operator ini dibahas pada kesempatan lain.

Berikut akan dicontohkan beberapa operator yang disajikan pada tabel operator penugasan diatas melalui program berikut :

```
      public class Penugasan{
        public static void main(String[] args) {
        int x=6, y=7, z=0;

        z=x+y;
        System.out.println("z = x + y = " + z);
        z*=x;
        System.out.println("A || B    = " + z);
        z/=y;
        System.out.println("!(A && B) = " + z);
        }
}
```

Keluaran program, jika berhasil dieksekusi akan menampilkan keluaran sebagai berikut :

```
z = x + y = 13
z *= x    = 78
z /= y    = 11
```

2.5 *Package* Java

Package (paket) adalah *namespace* yang mengatur sekumpulan kelas (*class)* dan antarmuka yang terkait. Secara konseptual kita dapat menganggap paket mirip dengan folder yang berbeda di komputer. *Package* di dalam java tersimpan pada direktory program (kompiler) java. *Package* dapat dipanggil suatu waktu bilamana dibutuhkan. Ada banyak *package* yang tersedia di dalam java, diantaranya *java.lang*; java.util.*; java.awt.*; java.sql.*;*

Di dalam *package* tersebut terdapat banyak kelas-kelas yang siap untuk digunakan. Pada kesempatan ini, kita akan coba menggunakan salah satu *package* yang umum digunakan untuk menangani perintah input data di dalam program *console*. *Package* yang digunakan yaitu ***java.util.*;*** Tanda bintang merupakan indikator dari suatu *class* yang berada didalam *package* tersebut.

Salah satu *Class* yang memiliki fungsi menangani input data didalam *package* java.util.* adalah *Scanner*. Untuk menggunakan *class Scanner* kita perlu menyertakannya bersama *package java.util.Scanner;* dengan menggunakan perintah ***import***. Secara lengkap package ini ditulis dengan **import java.util.Scanner;** yang dicantumkan didalam *package* dan diluar *class*. Lebih lanjut akan disajikan contoh penggunaan *package* yang dapat menangani input data melalui kibor (*keyboard*).

Pada contoh-contoh program diatas, nilai variabel diinisialisasikan sehingga ketika dipanggil, variabel tersebut menjawab dengan menampilkan nilai yang sudah ditetapkan. Seringkali nilai variabel justeru diberikan melalui proses input seperti pada contoh program berikut ini :

```
package inputdata;
import java.util.Scanner;
public class InputData{
      public static void main(String[] args) {
      int x;
      double y, z;

      Scanner io = new Scanner(System.in);
      System.out.print("Input x = ");
      x = io.nextInt();
```

```
            System.out.print("Input y = ");
            y = io.nextDouble();
            z=x*3/y;
            System.out.println("Nilai z adalah = " +z);
            }
}
```

Jika program berhasil dieksekusi, user diberikan kotak input untuk variabel x dan y, umpama x = 6, dan y = 2, maka hasil komputasi untuk z adalah 9.0 seperti pada tampilan keluaran program berikut :

```
Input x = 6
Input y = 2
Nilai z adalah = 9.0
```

MODUL 3

STRUKTUR KEPUTUSAN

Suatu program bisa saja akan mengulangi suatu blok program beberapa kali atau sampai hasil yang diharapkan sesuai dengan target yang ditentukan. Misal, apakah program akan menghitung nilai suatu variabel, atau menampilkan nilai suatu variabelnya. Pengulangan blok program ini ditentukan oleh struktur keputusan berdasarkan kondisi tertentu. Di dalam pemrograman kasus seperti ini disebut dengan struktur keputusan *(decision)*. Keputusan blok program akan diulang kembali atau tidak bergantung kepada standar kondisi yang ditetapkan oleh perancang program. Java memiliki dua struktur keputusan yang umum digunakan, yakni struktur keputusan dengan seleksi atau perulangan. Pada tahap ini akan disajikan konsep dasar struktur keputusan dengan seleksi beserta contoh kasusnya.

3.1 Seleksi

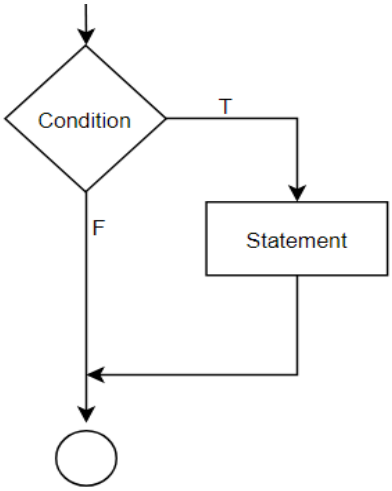
Seleksi disebut juga dengan pemilihan atau percabangan, yang memerintahkan kepada program untuk memilih satu diantara beberapa opsi berdasarkan kondisi tertentu. Ada beberapa model seleksi berdasarkan pada tingkatan atau banyaknya pilihan yang ditetapkan di dalam program. Model seleksi ini dapat digunakan sesuai kebutuhan perancang. Model seleksi yang dimaksud antara lain seleksi if tunggal, else statemen, seleksi if bertingkat, dan seleksi if bersarang.

3.1.1 Seleksi If Tunggal

Model ini sesuai dengan namanya, dimana hanya ada satu blok keputusan yang didasarkan pada satu kondisi saja. Secara sederhana, seleksi ini bermakna keputusan untuk mengeksekusi blok statemen akan dijalankan jika kondisi terpenuhi, dan jika kondisi tidak terpenuhi maka blok statemen diabaikan (dilewatkan). Makna tersebut sesuai dengan struktur program berikut :

```
if(ekspresi_boolean) {
  statement;
}
```

Untuk memahami cara kerja dari struktur diatas, amatilah bagan *flowchart* berikut :



Secara praktis, struktur keputusan selalu melibatkan operator relasi dan operator logika. Seringkali bahkan kedua operator tersebut digunakan secara bersamaan pada suatu kondisi untuk menentukan validitas batasan nilai, terutama untuk nilai-nilai data bertipe bilangan (bulat/ril) maupun bertipe teks (string ataupun char). Pembahasan operator telah disampaikan pada materi tipedata dan operator. Agar lebih jelas, penggunaan seleksi ***if*** tunggal ini akan disajikan pada contoh dibawah.

Misal suatu variabel *n* bertipe integer diinisialisasikan dengan nilai 5, maka program akan mengeksekusi apakah *n* benar-benar sama dengan 5. Pada contoh ini, kondisi akan disesuai dengan `n=5` sehingga blok program akan ditampilkan. Sebaliknya jika nilai *n* tidak sesuai dengan kondisi atau `n≠5`, maka blok program akan di abaiknya dan program akan berhenti. Perhatikan contoh berikut :

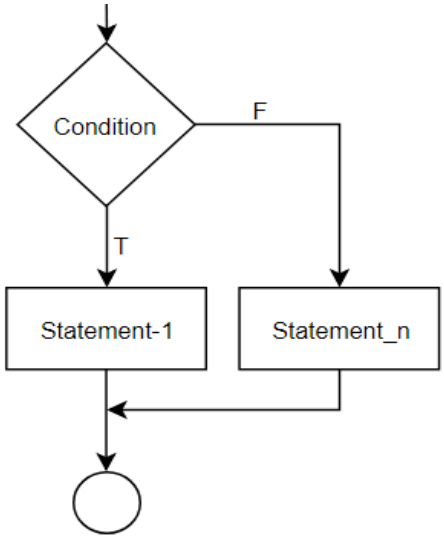
```
public class Iftunggal{
      public static void main(String[] args) {
      int n=5;
            if(n==5) {
         System.out.println("n adalah bilangan "+n);
        }
      }
}
```

Eksekusi program diatas akan menampilkan keluaran sebagai berikut :

```
n adalah bilangan 5
```

3.1.2 *Else .. Statement*

Struktur keputusan ini pada dasarnya merupakan alternatif manakala blok program (statemen) terdiri dari dua opsi, sehingga hanya akan ada satu opsi diantara kedua pilihan tersebut yang akan dijalankan (dieksekusi). Dimana blok statemen pada kondisi pertama akan dijalankan jika nilai variabel pada kondisi pertama terpenuhi, tetapi jika kondisi pertama tidak sesuai maka blok statemen tersebut akan diabaikan, selanjutnya ***else*** akan mengambil alih untuk mengeksekusi blok statemen di dalamnya. Bagan *flowchart* berikut menunjukan cara kerja struktur keputusan pada model ini. Perhatikan baik-baik bagan yang dimaksud di bawah ini :



Untuk menggunakan struktur keputusan pada model ini, perancang dapat mengikuti struktur program berikut :

```
            if(ekspresi_boolean) {
               statement_one;
            } else {
            statement_n;
}
```

Berikut ini akan disajikan contoh kasus, umpama *n* adalah integer bernilai 5, dan *k* adalah integer yang diinput melalui kibor, dan *n* akan ditambahkan dengan nilai k. Tentu nilai *n* akan berubah dari semula, selanjutnya program akan mencetak informasi sesuai dengan kondisi `n>10`. Perhatikan contoh programnya berikut :

```
      import java.util.Scanner;
      public class IfElse{
            public static void main(String[] args) {
            int k, n=5;

            Scanner io = new Scanner(System.in);
            System.out.print("Input nilai k : ");
            k=io.nextInt();

            n+=k;
            if(n>=10) {
                  System.out.println("n saat ini > 10");
            }else {
                  System.out.println("n saat ini < 10");
            }

            System.out.println("n saat ini adalah "+n);
            }
}
```

Jika dieksekusi, program diatas akan menampilkan keluaran program seperti berikut :

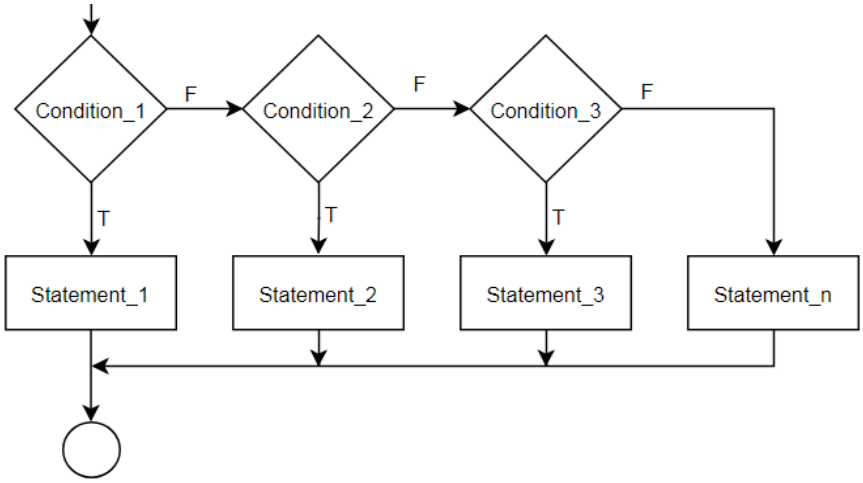
```
Input nilai k : 6
n saat ini > 10
n saat ini adalah 11
```

3.1.3 Seleksi If majemuk

Seleksi model ini merupakan pengembangan dari seleksi dengan dua pilihan blok statemen atau ***else … statement.*** Seleksi ini disebut juga dengan ***Else If Ladder*** yang bermakna tangga. Secara struktur model ini membentuk untaian tangga yang terus berkelanjutan.

Prinsip kerja struktur ini menguji nilai variabel dikondisi pertama, jika kondisi pertama terpenuhi maka blok statemen pertama yang akan dijalankan, tetapi jika kondisi pertama tidak terpenuhi maka blok statemen tersebut diabaikan, dan program akan menguji kondisi kedua, jika nilai variabel pada kondisi kedua terpenuhi maka blok statemen didalamnya yang akan dijalankan, tetapi jika kondisi kedua tersebut juga tidak terpenuhi maka blok statemen pertama dan kedua akan dilewatkan, dan akan menguji kondisi ketiga, keempat dan seterusnya sampai mencapai *else***.** Struktur *else* merupakan alternatif manakala semua kondisi tidak terpenuhi. Kata *else* memiliki makna kecuali atau selain daripada kondisi pertama, kedua, ketiga dan seterusnya sebelum *else*. Artinya jika semua kondisi tidak

terpenuhi, maka blok statemen yang berada didalam *else* yang akan dijalankan. Perhatikanlah bagan *flowchart* berikut :



Jika mengacu kepada bagan alir diatas, struktur keputusan *else if ladder* dapat dituliskan seperti pada struktur dasar berikut :

```
if(ekspresi_boolean_1) {
    statement1;
} else if (ekspresi_boolean_2) {
    statement2;
} else if (ekspresi_boolean_3) {
    statement3;
} else {
    statement4;
}
```

Perhatikan contohkan pada studi kasus untuk menentukan jumlah beban kredit semester (beban sks) mahasiswa UIN Sumatera Utara Medan yang dapat diambil pada setiap semester. Dimana ketentuan jumlah beban sks-nya mengacu kepada Indeks Prestasi Semester (IPS) yang diperoleh pada semester sebelumnya. Adapun ketentuannya seperti pada tabel berikut :

IPS	SKS
3.50 - 4.00	24
3.00 - 3.49	22
2.50 - 2.59	20

2.00 - 2.49	18
1.50 - 1.99	16
1.00 - 1.49	14
0.00 - 0.99	10

Berdasarkan tabel IP diatas dapat dirancangkan programnya seperti berikut, perhatikan kode program secara teliti dibawah ini :

```
import java.util.Scanner;
public class Elseifladder{
	public static void main(String[] args) {
	float ips; int sks=0;

	Scanner io = new Scanner(System.in);
	System.out.print("Berapakah IP Anda ?  ");
	ips=io.nextFloat();

		if(ips>=3.50 && ips<=4.00) {
			sks=24;
		} else if(ips>=3.00 && ips<=3.49) {
			sks=22;
		} else if(ips>=2.50 && ips<=2.59) {
			sks=20;
		} else if(ips>2.00 && ips<=2.49) {
			sks=18;
		} else if(ips>=1.50 && ips<=1.99) {
			sks=16;
		} else if(ips>=1.00 && ips<=1.49) {
			sks=14;
		} else if(ips>=0.00 && ips<=0.99) {
			sks=10;
		} else {
System.out.println("Maaf, data ips yang diinput tidak
valid.");
		}
		System.out.println("Beban sks Anda adalah
"+sks);
	}
}
```

Selanjutnya hasil eksekusi program diatas akan menampilkan keluaran sebagai berikut :

```
Berapakah IP Anda ?  3.46
```

```
Beban sks Anda adalah 22
```

3.1.4 *Nested* If

Kadangkala struktur if dibentuk secara bertumpuk-tumpuk. Dimana struktur seleksi if mengandung seleksi if didalamnya. Struktur ini disebut dengan istilah *nested if* atau if bersarang. Yakni struktur seleksi yang membungkus struktur seleksi if anak. Kurang lebih model seleksi *nested if* ditulis dengan struktur seperti berikut ini :

```
if( ekspresi_boolean1) {
    statemen_if_utama;
 if(ekspresi_boolean2) {
      statement_if_within_if;
   }
}
```

Perhatikan contoh berikut dimana kondisi pertama akan menguji nilai `x=10`, jika terpenuhi maka blok statemen didalamnya akan dijalankan dengan menguji perbandingan antara nilai x dan y. Jika nilai y > x, maka blok statemen pada if kondisi (anak) pertama yang akan dijalankan, tetapi jika kondisi tersebut tidak terpenuhi, maka blok statemen pada bagian *else* yang akan dijalankan. Perhatikan program berikut :

```
public class Nestedif{
  public static void main(String[] args) {
      int x = 10, y=20;

      if(x==10){
        if(y>x){
          System.out.println("y lebih besar dari x.");
         }else {
          System.out.println("x lebih besar dari y.");
          }

           System.out.println("nilai x = "+x);
           System.out.println("nilai y = "+y);
      }
  }
}
```

Program diatas akan menampilkan keluaran seperti berikut :

```
y lebih besar dari x.
nilai x = 10
```

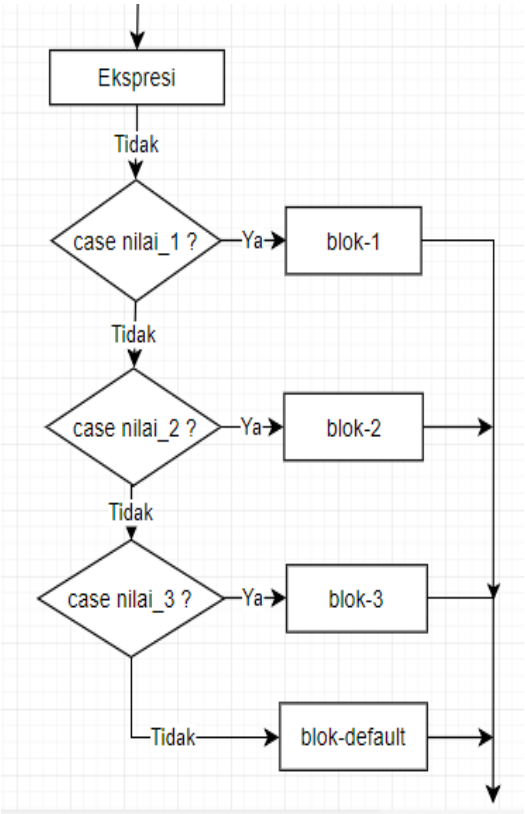
```
nilai y = 20
```

3.1.5 Seleksi dengan *switch*

Seleksi jenis ini merupakan alternatif dari seleksi if bertangga (majemuk). Kadangkala blok statemen terdiri dari banyak pilihan. Hal tersebut kurang sesuai untuk seleksi dengan *if*. Penggunaan seleksi ini mengacu kepada struktur program berikut :

```
switch(variabel){
    case nilai-1:
        blok-1;
        break;
    case nilai-2:
        blok-2;
        break;
    . . .
    default:
        default-blok;
        break;
}
```

Prinsip kerja struktur statemen *switch* adalah dengan menguji nilai variabel yang diinput (diinisialisasi) dengan daftar nilai pada *case* pertama hingga *case* terakhir. Jika nilai variabel sesuai dengan kriteria pada *case* pertama, maka blok statemen dibawahnya akan dieksekusi, dan jika variabel tersebut nilainya tidak sesuai, maka program akan menguji nilai variabel pada *case-case* selanjutnya (kedua, ketiga, keempat, dan seterusnya) hingga tidak ada daftar *case* lagi yang tersedia. Jika semua *case* sudah ditelusuri dan tidak ditemukan satu *case* pun yang sesuai, maka *default-case* yang akan mengambil alih dan blok statemennya yang akan dieksekusi. Bagan *flowchart* berikut dapat menjelaskan alur kerja dari struktur seleksi dengan *switch* berikut ini :



Berikut ini akan disajikan contoh penggunakan struktur dengan *switch*. Dalam program ini disajikan beberapa dartar operasi matematis yang diwakili dengan simbol opetaror matematika. Secara lengkap perhatikan program berikut :

```
import java.util.*;
public class Switchcase{
      public static void main(String[] args) {
      double num1, num2, result=0;

      Scanner io = new Scanner(System.in);
      System.out.print("Input 2 angka : ");
      num1=io.nextDouble();
      num2=io.nextDouble();

      System.out.print("Input operator [+,-,*,/] : ");
      char op = io.next().charAt(0);
      switch(op) {
      case '+':
      result=num1+num2;
      System.out.println("number 1 + number 2 = "+result);
      break;
      case '-':
      result=num1-num2;
      System.out.println("number 1 - number 2 = "+result);
      break;
```

```
            case '*':
            result=num1*num2;
            System.out.println("number 1 x number 2 = "+result);
            break;
            case '/':
            result=num1/num2;
            System.out.println("number 1 / number 2 = "+result);
            break;
            default:
            result=num1+num2;
      System.out.println("Maaf, operator yang Anda input tidak
      valid.");
                  break;
                  }
                  }
      }
```

Program tersebut akan menampilkan keluaran seperti berikut :

```
Input 2 angka : 6 3
Input operator [+,-,*,/] : *
number 1 x number 2 = 18.0
```

MODUL 4

STRUKTUR KEPUTUSAN (2)

4.1 Perulangan

Jika dipandang secara bagan menggunakan *flowchart*, perulangan merupakan struktur yang sama dengan struktur seleksi atau pemilihan karena menggunakan simbol yang sama. Namun secara defenisi perulangan dan seleksi adalah struktur yang berbeda. Struktur perulangan merupakan struktur untuk mengulang statemen sebanyak nilai tertentu atau sampai kondisi tidak memenuhi untuk dilakukan perulangan. Dengan demikian tampak sekali bahwa struktur perulangan menggunakan prinsip yang diterapkan pada struktur seleksi. Perbedaanya adalah struktur seleksi mengulangan kondisi untuk memilih satu diantara beberapa opsi pilihan yang sesuai dengan keadaan yang diminta. Sementara perulangan mengulang statemen sebanyak *n* tertentu. Nilai *n* adalah nilai bilangan bulat yang ditentukan sesuai kebutuhan. Perlu diingat bahwa perulangan, apapun bentuknya, tidak mampu menangani data bertipe teks char maupun string.

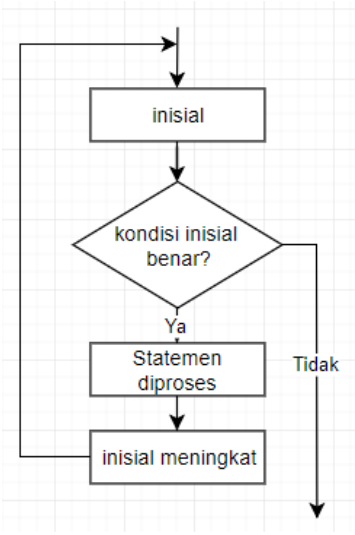
Implementasi perulangan dalam kehidupan sehari-hari terlihat pada perulangan bilangan mundur dari nilai tertinggi ke nilai rendah 0 di lampu merah di persimpangan. Bahasa java memiliki beberapa bentuk perulangan yakni perulangan *for*, *while*, dan, *do..while.* Baik perulangan *for*, *while*, dan, *do..while* ketiganya memungkinkan memiliki perulangan anak di dalamnya. Struktur ini disebut dengan perulangan bersarang *nested looping.*

4.2 Perulangan *FOR*

Perulangan model ini merupakan perulangan dengan jumlah perulangan (iterasi) yang sudah tetap, dimana banyaknya iterasi sudah dapat diprediksi berapa kali akan berulang atau kapan iterasi akan diberhentikan. Struktur dasar perulangan ***For*** mengacu pada struktur program berikut :

```
for (inisialisasi; syarat; pengubah)
statement;
```

Secara grafis, alur perulangan for ditunjukkan pada bagan flowchart berikut :



Meskipun perulangan dapat menangani data bertipe bilangan, namun untuk perulangan *for* hanya mampu menangani data bertipe bilangan bulat saja. Supaya lebih jelas, perhatikan contoh program berikut :

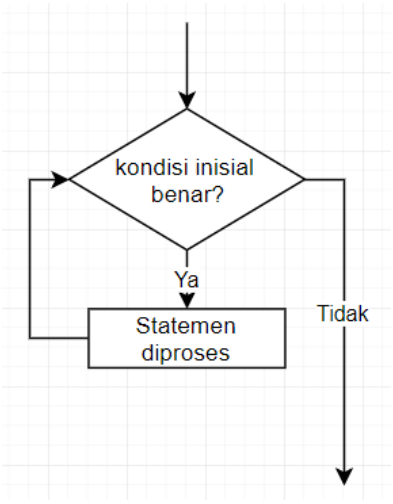
```
public class ForLooping {
      public static void main(String args[]) {
      int bilangan;
      for(bilangan=1;bilangan<=10;bilangan++)     {
            System.out.print(bilangan+" ");
           }
      }
}
```

Perhatikan keluaran berikut, dimana bilangan = 1 akan dicetak secara berulang sampai nilang bilangan tidak lebih atau =10. Keluaran akan seperti pada data berikut :

```
1 2 3 4 5 6 7 8 9 10
```

4.3 Perulangan *While*

Kata *while* dapat diartikan dengan makna ‘selama’ atau ‘sepanjang’. Maksudnya, perulangan *while* merupakan struktur algoritma dimana perulangan akan terus dijalankan selama kondisi bernilai benar. Struktur perulangan while mengacu pada *flowchart* dan struktur berikut ini :




```
        Inisialisasi
        while(syarat_kondisi)
        {
            statements;
        ekspresi_increment;
}
```

Perhatikan dengan teliti contoh berikut. Ini merupakan alternatif bilamana programmer akan menggunakan struktur perulangan. Adapun contoh yang dimaksud seperti pada kode berikut :

```
public class Tes {
  public static void main(String args[]) {
  int bilangan=1;
  while(bilangan<=10)
  {
   System.out.print(bilangan+" ");
   bilangan++;
  }
}
}
```

Jika program tersebut dijalankan, akan menampilkan keluaran sebagai berikut :

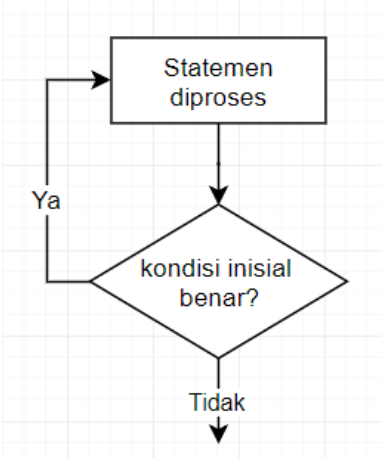
```
1  2 3 4 5 6 7 8 9 10
```

4.4 Do..While

Perulangan ini memiliki struktur yang mirip dengan perulangan *while*. Akan tetapi prinsip kerjanya sedikit berbeda. Pada perulangan *while*, blok statemen pertama dijalakan

bilamana kondisi di dalam *while* terpenuhi dan begitu seterusnya sampai kemungkinan perulangan akan dilanjutkan terus sampai tidak ada lagi kondisi yang sesuai. Sementara pada perulangan *do..while*, blok statemen dijalankan setidaknya satu kali, dan untuk mengulangi proses perulangan berikutnya blok tersebut harus melalui proses pengujian kondisi yang ditentukan di dalam *while*. Untuk menggunakan perulangan dengan struktur ini, struktur program mengacu kepada bentuk umum, berikut dengan bagan *flowchart* di bawah ini :

```
do{
   .....
   .....
   }while(condition);
```



Pada contoh program perulangan dengan struktur *while* kita telah melihat kode program seperti yang ditampilkan diatas. Alternatif untuk program tersebut dengan struktur *do..while* akan berbentuk seperti pada program berikut ini :

```
public class Tes {
public static void main(String args[]) {
int bilangan=1;
do{
      System.out.print(bilangan+" ");
      bilangan++;
}while(bilangan<=10);
}
}
```

Kita bisa melihat perbedaan struktur antara perulangan *while* dan *do..while* pada kedua program diatas. Perbedaannya tampak pada kode *while* yang terletak diawal ataupun diakhir. Sehingga dengan ini tidaklah meminggungkan bagi pemula untuk menentukan

struktur mana yang akan digunakan. Keluaran program juga tidak berbeda diantara keduanya. Manakala program diatas dijalankan, juga menampilkan keluaran yang sama baik struktur maupun tampilannya seperti berikut dibawah ini :

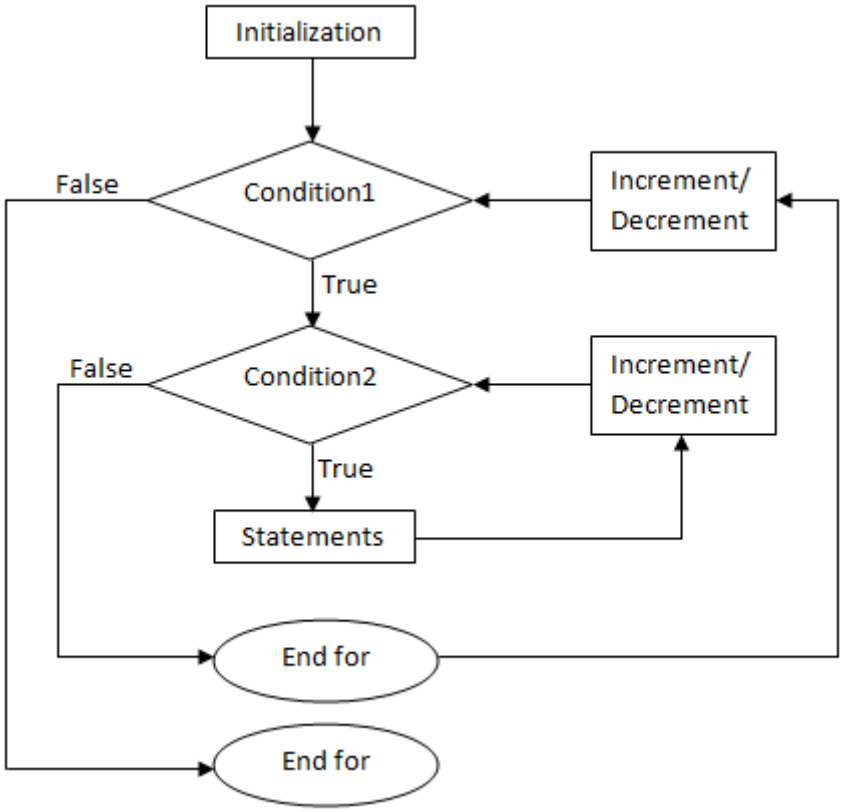
```
1  2 3 4 5 6 7 8 9 10
```

4.5 Perulangan Bersarang

Seringkali kita jumpai dimana perulangan mengandung perulangan lagi di dalamnya. Ini bisa saja terjadi pada struktur perulangan *for*, *while* maupun pada perulangan model *do..while.* Struktur perulangan seperti ini disebut dengan perulangan bersarang atau *nested looping.* Yakni perulangan yang di dalamnya terdapat perulangan lain, yang bisa saja saling terkait maupun tidak. Berikut ini disajikan struktur *nested looping* menggunakan struktur *for*, yang secara bentuk ditampilkan sebagai berikut

```
for(inisialisasi; syarat; pengubah){
  for(inisialisasi; syarat; pengubah){
    statement ;
  }
}
```

Untuk memahami prinsip kerja perulangan bersarang, berikut ini ditampilkan bagan flowchart yang relevan dengan struktur diatas. Untuk setiap model *looping*, mungkin saja berbeda dengan model yang disajikan disini. Adapun bagan yang dimaksud adalah sebagai berikut :



Sumber : programtopia.net

Dengan menggunakan struktur diatas berikut ini program contoh disajikan untuk mengimplementasikan struktur perulangan bersarang menggunakan struktur *for* sebagai berikut :

```
public class Tes {
      public static void main(String args[]) {
      for(int i=1; i<=3; i++){
            for(int j=1; j<=3; j++) {
                  System.out.format("%d * %d = %d\n",i,j,
i*j);
                  } System.out.println();
            }
      }
}
```

Jika program diatas dijalankan, maka program akan menampilkan keluaran perkalian 3 x 3 seperti berikut :

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
```

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
```

Program tersebut diatas juga dapat diimplementasikan pada perulangan model *while* dan *do..while.* Pengguna hanya cukup memperhatikan struktur dasar perulangan tersebut dan menyesuaikan nilai awal dan syarat perulangannya.

MODUL 5

LARIK

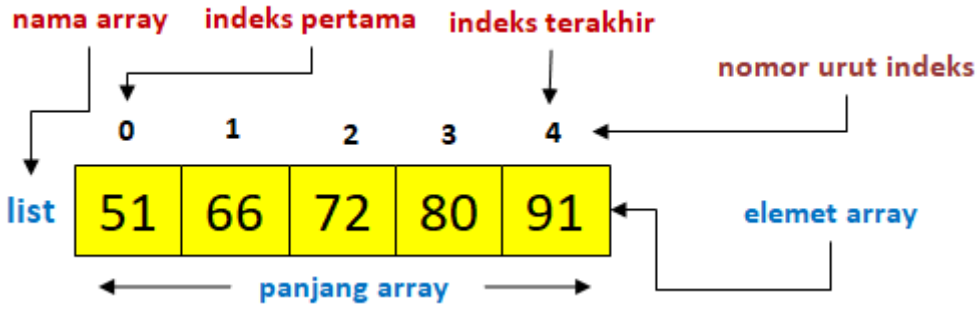
5.1 Konsep Larik

Larik atau umum disebut dengan array, merupakan himpunan data dengan indeks terurut yang memiliki tipe data yang sama. Misal, himpunan data terdiri dari buku, pensil, pena, penggaris, dan alat-alat tulis lainnya, dimana data-data tersebut menggunakan tipedata string. Data-data tersebut dapat dikoleksikan di dalam larik 'alat tulis' yang tipedatanya string.

Larik terbagi dalam beberapa struktur tergantung representasi dan kopleksitas datanya. Bilamana larik hanya memiliki sebuah urutan data saja, ini dinamakan dengan larik satu dimensi. Tetapi manakala koleksi data terbentuk dalam beberapa atau banyak koleksi data terurut, maka ini dinamakan dengan larik multi dimensi.

5.1.1 Larik Satu Dimensi

Larik satu dimensi merupakan koleksi data bertipe sama yang struktur datanya merepresentasikan sebuah kolom saja atau baris saja. Struktur larik satu dimensi seperti yang ditunjukkan pada gambar berikut :



Beberapa komponen yang dapat dilihat pada gambar struktur larik diatas, yakni nama array (larik), elemen, maupun indeks. Nama larik adalah variabel yang memiliki tipe tertentu, yang nantinya akan menampung data-data yang tipenya sesuai dengan tipe larik tersebut. Sedangkan elemen, merupakan koleksi data yang terekam didalam suatu larik, yang tersusun sedemikian rupa dengan indeks berurutan. Suatu indeks adalah bilangan bulat integer yang memiliki peran penting dalam menempatkan elemen data. Indeks akan menunjukkan posisi data bilamana indeks disebutkan. Misal, elemen 66 tersimpan pada larik *list* yang berada di posisi indek 1, dimana indek 1 adalah elemen data ke 2.

a) Deklarasi Larik

Suatu larik harus dideklarasikan diawal, baik nama lariknya, maupun tipe dan ukurannya. Supaya larik yang dibuat dapat memiliki kemampuan dalam penyimpanan data. Pendeklarasian larik tidak hanya mengenalkan tipe dan ukurannya, tetapi juga mendefenisikan bentuk dimensi lariknya. Pada larik dimensi satu, deklarasi cukup sederhana dan mudah dipahami. Adapun deklarasi larik satu dimensi mengacu kepada struktur berikut :

```
tipedata[] namalarik; atau
tipedata []namalarik; atau
tipedata namalarik[];
```

b) Instansiasi

Prosess instansiasi ini merupakan proses penentuan ukuran larik, namun sering juga sekaligus menginisialisasikan data-data larik di dalamnya. Instansiasi suatu larik dapat menggunakan struktur berikut :

```
tipedata namalarik[]=new tipedata[ukuran];
```

Disini kita telah memiliki contoh struktur larik yang disajikan pada gambar diatas. Dimana larik diberi nama *list* dengan tipe integer jumlah 5 data yang juga bertipe integer bilangan bulat. Maka instansiasi lariknya dapat dituliskan seperti berikut ini :

```
int list[]= new int[5];
```

Inisialisasi data dapat dituliskan seperti pada struktur berikut :

```
list[0]=51;
list[1]=66;
list[2]=72;
list[3]=80;
list[4]=91;
```

Inisialisasi elemen data larik juga dapat menggunakan struktur yang langsung bersinggungan dengan deklarasi lariknya seperti pada struktur berikut :

```
int list[]={51,66,72,80,91};
```

Untuk melengkapi struktur larik diatas, berikut ini akan disajikan dalam program dengan nama class **ArrayDimsatu** dibawah ini :

```
public class ArrayDimsatu {
  public static void main(String args[]) {
  int list[] = {51,66,72,80,91};

  for(int i=0; i<list.length; i++){
  System.out.println("Elemen ke " +(i+1)+ " : "+list[i]);
        }
        }

  }
```

Program diatas akan menampilkan keluaran program sebagai berikut :

```
Elemen ke 1 : 51
Elemen ke 2 : 66
Elemen ke 3 : 72
Elemen ke 4 : 80
Elemen ke 5 : 91
```

c) Mengakses elemen larik

Elemen suatu larik dapat diakses secara tunggal hanya dengan menyebutkan nomor indeks didalam kurung siku "`[]`" yang menunjuk kepada suatu elemen data. Perhatikan pula nama larik harus dicantumkan agar kompiler mengenali variabel larik yang mana yang akan merespon permintaan. Misal, kita ingin mengakses elemen data yang bernilai 72 di dalam larik *list*, maka ini dilakukan dengan menyebut nama larik dan nomor indeks yang menunjukkan posisi data tersebut dengan struktur seperti `list[2]`. Secara lengkap ditulis dengan struktur berikut :

```
System.out.println("Elemen ke " + " : " +list[2]);
```

Dengan demikian maka keluaran untuk ekspresi tersebut akan menampilkan data

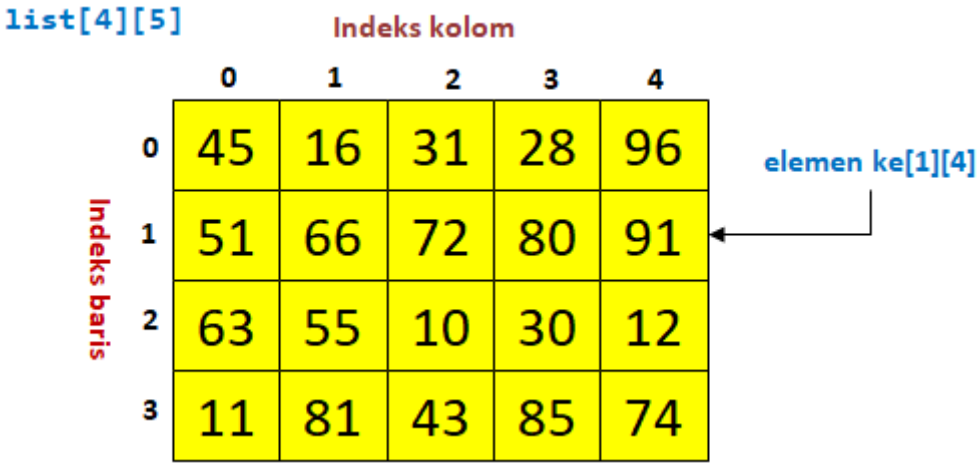
```
Elemen ke  : 72
```

5.1.2 Larik dua dimensi

Ini sedikit berbeda dengan larik satu dimensi. Pada larik satu dimensi hanya merujuk kepada satu arah vektor, apakah vektor baris saja atau vektor kolom saja. Nah, pada larik dua dimensi kedua unsur (baris dan kolom) tersebut melengkapi larik ini. Pada larik dua dimensi

indeks baris selalu ditulis disebelah kiri, selanjutnya indeks kolom ditulis di sisi kanan pada bagian deklarasi lariknya.

Indeks baris dan kolom pada larik dua dimensi, merupakan komponen yang merepresentasikan larik ini sebagai suatu matriks atau tabel. Untuk memahami struktur tipe larik dua dimensi, dibawah ini disajikan struktur larik di dalam memori komputer sebagai berikut :



Perhatikan pada gambar, kita memiliki larik dengan nama *list* yang diikuti dengan keterangan `[4][5],` nilai 4 mewakili jumlah baris dan nilai 5 mewakili kolom. Dengan demikian *list* merupakan larik berbentuk matrik (tabel) yang memiliki kemampuan menyimpan elemen data larik sebesar 4 baris dan 5 kolom. Terlihat juga bahwa elemen-elemen data larik adalah bilangan-bilangan yang berada di dalam kotak. Untuk mengakses elemen data yang bernilai 91, ini bisa ditulis dengan bentuk `list[1][4].`

a) Deklarasi array dua dimensi

Dalam bahasa java, deklarasi larik dua dimensi mengacu kepada deklarasi larik satu dimensi, namun pada larik ini dimensi (baris dan kolom) harus disertakan. Struktur larik dua dimensi seperti berikut :

```
tipedata namalarik[ukuran baris][ukuran kolom];
```

Suatu larik dua dimensi merujuk kepada bentuk matriks maupun tabel, yang pada dirinya terdapat baris dan kolom yang elemen-elemen datanya diakses oleh indeks yang mewakili baris dan kolom. Gambar larik diatas merupakan contoh struktur larik dua dimensi di dalam memori. Gambar larik yang ditunjukkan pada gambar diatas dapat dituliskan dengan struktur java seperti dibawah ini :

```
int list[4][5];
```

Struktur diatas juga dapat diinisialisasikan datanya secara langsung seperti pada contoh larik dimensi satu diatas. Kurang lebih inisialisasi datanya bisa seperti berikut ini :

```
int list[][]={{45,16,31,28,96},{51,66,72,80,91},
            {63,55,10,30,12},{11,81,43,85,74}};
```

Perlu diingat bahwa deklarasi larik yang menyertakan inisialisasi nilai di dalamnya, tidak harus mencantumkan ukuran lariknya. Perhatikan pada contoh deklarasi larik satu dan dua dimensi diatas, inisialisasi nilai tidak mencantumkan ukuran panjang lariknya, kecuali inisialisasinya mengacu kepada prinsip instansiasi objek. Secara lengkap larik dua dimensi yang disajikan diatas akan berbentuk program berikut :

```
public class ArrayDimdua {
   public static void main(String args[]) {
        int list[][]={{45,16,31,28,96},
              {51,66,72,80,91},
              {63,55,10,30,12},
              {11,81,43,85,74}};

        for(int i=0; i<4; i++){
              for(int j=0;j<5;j++) {
              System.out.print(list[i][j]+" ");
        }System.out.println();
        }
   }
}
```

Jika dijalankan program diatas akan menampilkan elemen data larik yang membentuk seperti matrik ordo atau tabel seperti berikut ini :

```
45 16 31 28 96
51 66 72 80 91
63 55 10 30 12
11 81 43 85 74
```

b) Mengakses elemen larik

Akses elemen larik dimensi dua sama juga seperti mengakses elemen pada larik dimensi tunggal. Akan tetapi pada larik dimensi dua, unsur kolom dan baris harus dicantumkan. Misal, apabila kita ingin mengakses elemen yang bernilai 10, maka kita perlu mencantumkan indeks baris dan kolom yang menunjukkan elemen data 10. Kita lihat elemen 10 berada diposisi indek baris ke-3 dan kolom ke-3, maka ini akan ditulis dengan struktur **`list[2][2];`** kenapa? Karena indeks baris dan indeks kolom diawali dengan nilai 0, maka indeks baris dan kolom ke 3 menjadi posisi 2.

5.2 Larik multidimensi

Karakteristik larik multidimensi adalah memiliki indeks baris dan indeks kolom dari sisi-sisi yang membentuk dimensi. Bentuk larik multidimensi yang paling sederhana adalah larik dua dimensi, yang hanya memiliki indeks baris dan kolom dari sisi dua arah saja. Larik multidimensi bisa saja lebih komplek, dengan sisi-sisinya bisa dipandang dari sisi atas, bawah, samping kiri maupun kanan.

Pada pemrograman konsol (berbasis teks) larik multidimensi sulit dipandang dan sulit dipahami, karena sisi-sisi yang mewakili dimensi tidak mudah diidentifikasi, begitu juga pada pemrograman berbasis grafik. Larik multidimensi lebih mudah dipahami pada pemrograman komputer grafik karena keluaran program dapat dilihat dari sisi yang beragam. Berikut ini contoh penerapan larik multidimensi dengan merujuk kepada tiga dimensi. Untuk larik tiga dimensi menggunakan struktur java berikut ini :

```
int list[3][4][2];
```

Selanjutnya, dengan menggunakan struktur diatas, kode program java untuk larik tiga dimensi secara lengkap sebagai berikut :

```
public class Tes{
   public static void main(String args[]){
      int list[][][] = new int[3][4][2];
      int i, j, k, num=1;

      for(i=0; i<3; i++){
          for(j=0; j<4; j++){
              for(k=0; k<2; k++){
                  list[i][j][k] = num;
                  num++;
              }
          }
```

```
            }

             for(i=0; i<3; i++){
                for(j=0; j<4; j++){
                    for(k=0; k<2; k++){
                        System.out.print("list[" +i+ "][" +j+
      "][" +k+ "] = " +list[i][j][k]+ "\t");
                    }
                    System.out.println();
                }
                System.out.println();
            }
        }
}
```

Jika program tersebut dijalankan, akan menampilkan keluaran program berikut :

```
list[0][0][0] = 1     list[0][0][1] = 2
list[0][1][0] = 3     list[0][1][1] = 4
list[0][2][0] = 5     list[0][2][1] = 6
list[0][3][0] = 7     list[0][3][1] = 8

list[1][0][0] = 9     list[1][0][1] = 10
list[1][1][0] = 11    list[1][1][1] = 12
list[1][2][0] = 13    list[1][2][1] = 14
list[1][3][0] = 15    list[1][3][1] = 16

list[2][0][0] = 17    list[2][0][1] = 18
list[2][1][0] = 19    list[2][1][1] = 20
list[2][2][0] = 21    list[2][2][1] = 22
list[2][3][0] = 23    list[2][3][1] = 24
```

Kita bisa melihat keluaran program diatas, sulit untuk memahami bentuknya karena tidak tersusun. Bila kita cetak tanpa nama lariknya, akan tampil seperti berikut :

```
1     2
3     4
5     6
7     8

9     10
11    12
13    14
15    16

17    18
19    20
21    22
23    24
```

Kedua bentuk keluaran tersebut sama-sama sulit dipandang dari sisi-sisi yang memiliki dimensi. Inilah kenapa multidimensi pada pemrograman konsol lebih baik disajikan dalam bentuk dua dimensi saja, karena sisi-sisinya mudah dipahami. Akan tetapi pemahaman untuk larik multidimensi mutlak harus dipahami sebagai dasar untuk memahami bagaimana data disimpan di dalam memori komputer.

5.3 Meng-*input* Elemen Larik

Elemen larik bisa juga diinput secara manual, pada umumnya memang elemen larik ditambahkan secara manual. Untuk menambahkan elemen larik secara manual, kita harus benar-benar memahami struktur dan cara kerjanya. Dibawah ini kita akan melihat bagaimana elemen data larik diinput secara manual. Misal pada kasus ini kita akan membuat program untuk menghitung nilai rerata dari elemen data masukan berikut :

```
import java.util.*;
public class Tes {
   public static void main(String args[]) {
         int n, number, avg=0, sum=0;

         Scanner io = new Scanner(System.in);
         System.out.print("Input bilangan n : ");
         n = io.nextInt();

         //instansiasi larik
               int arr[]=new int[n];
         for(number=0; number<n; number++) {
               System.out.print("Input bilangan : ");
               arr[number]=io.nextInt();

               sum+=arr[number];
         }
         avg = sum/number;
         System.out.format("Rerata nilai : %d",avg);
   }
}
```

Selanjutnya keluaran program tersebut akan menampilkan keluaran rerata nilai input yang diberikan. Keluaran program akan tampak seperti berikut ini :

```
Input bilangan n : 5
Input bilangan : 89
Input bilangan : 78
Input bilangan : 67
Input bilangan : 56
```

```
Input bilangan : 93
Rerata nilai : 76
```

5.4 Larik dengan foreach

Ada kalanya suatu larik dapat ditampilkan dengan menggunakan fungsi *foreach* dimana fungsi ini merupakan struktur perulangan larik yang lebih sederhana. Struktur foreach mengacu kepada struktur berikut :

```
    for(variabel_indeks : variabel_larik){
                statement;
  }
```

Contoh berikut ini menampilkan struktur larik dimensi tunggal yang menerapkan fungsi *foreach* untuk menambahkan elemen data larik ke dalam lariknya. Secara jelas dapat dilihat pada program berikut ini :

```
import java.util.*;
public class Tes {
      public static void main(String args[]) {
      int[] arr= new int[5];
      int sum=0;
      Scanner io=new Scanner(System.in);
      for(int i:arr) {
      System.out.print("Input bilangan : ");
      arr[i]=io.nextInt();

      sum+=arr[i];
      }
      System.out.format("Total bilangan  : %d",sum);
      }
}
```

Program tersebut akan menghitung total nilai bilangan bulat. Dengan persamaan `sum+=arr[i],` akan tampak keluaran program sebagai berikut :

```
Input bilangan ke1 : 6
Input bilangan ke1 : 4
Input bilangan ke1 : 3
Input bilangan ke1 : 8
Input bilangan ke1 : 5
Total bilangan  : 26
```

MODUL 6

PEMODELAN

6.1 Unified Modeling Language

Unified Modeling Language (UML) sering dijumpai pada pemrograman-pemrograman berorientasi objek. Keberadaan UML dalam pemrograman adalah sebagai notasi standar untuk memodelkan gambaran atau keadaan dunia nyata suatu sistem. Pemodelan UML diterapkan sebagai tahap awal dalam proses pengembangan perangkat lunak yang merujuk kepada program berorientasi objek. Melalui UML ini, pengembang dapat menentukan, menguraikan secara visual, membangun dan mendokumentasikan berbagai komponen sistem perangkat lunak. Dimana UML digunakan untuk, (1) menggambarkan diagram domain permasalahan, (2) disain perangkat lunak yang diusulkan, atau, (3) juga implementasi perangkat lunak yang sudah selesai (siap pakai). Di dalam buku yang ditulis Robert Cecil Martin (2002:1), Fowler menuliskan bahwa ketiga hal tersebut merupakan penjelasan terkait (1) Konseptual, (2) Spesifikasi, dan, (3) Implementasi--suatu perangkat lunak, entah itu sebelum, sedang, atau setelah dikembangkan. Ketiga tujuan tersebut menegaskan munculnya suatu diagram UML, yang merupakan suatu notasi grafis untuk menggambarkan diagram konsep perangkat lunak secara sederhana maupun secara lengkap.

Diagram spesifikasi dan implementasi berkaitan dengan bagaimana suatu perangkat lunak dibangun, dengan kata lain, kedua diagram tersebut berhubungan dengan kode sumber (kode program). Melalui diagram spesifikasi perancang dapat memahami entitas atau properti apa saja yang dibutuhkan dalam suatu perangkat lunak. Hal yang sama juga pada diagram implementasi yang berfungsi menerjemahkan bagaimana perangkat lunak ditulis dalam kode sumber. Inilah kenapa diagram ini memiliki ambiguitas dan bersifat formalitas meskipun tidak terlalu membahayakan.

Berbeda dengan diagram spesifikasi dan implementasi, diagram konseptual tidak bersinggungan dengan kode program, melainkan hanya menjelaskan konsep dan abstraksi yang berkaitan dengan permasalahan manusia. Misal ini menjelaskan bagaimana suatu kelas memiliki hubungan generalisasi dengan kelas yang lain, dan sebatasmana hubungan kelas-kelas tersebut.

6.2 Jenis Diagram

Dalam buku yang sama, Robert menuliskan ada tiga jenis diagram utama, yakni diagram statis, dinamis, dan, diagram fisik. Dimana diagram statis menggambarkan struktur logis

elemen perangkat lunak yang tidak berubah dengan menggambarkan kelas, objek, dan struktur data, dan hubungan yang ada di antara mereka. Diagram dinamis menunjukkan bagaimana entitas perangkat lunak berubah selama eksekusi dengan menggambarkan aliran eksekusi, atau cara entitas berubah. Sementara diagram fisik menunjukkan struktur fisik entitas perangkat lunak yang tidak berubah dengan menggambarkan entitas fisik seperti file sumber, pustaka, file biner, file data, dan lain-lain., dan hubungan yang ada di antara mereka.

Jika merujuk kepada fungsinya, diagram statis, diagram dinamis, dan diagram fisik relevan dengan jenis pemodelan diagram berikut ini :

1) Diagram Struktural
2) Diagram Tingkah Laku
3) Diagram Arsitektur

6.2.1 Digram Struktural

Diagram strutktural *(Structure Diagrams)* merupakan diagram pemodelan struktural yang merujuk kepada diagram statis dengan sifat-sifat yang statis pula, dalam konteks perancangannya. Diagram ini memberikan informasi kepada pengguna untuk menerjemahkan suatu diagram ke dalam kode program. Secara umum, digram ini terdiri dari beberapa jenis diagram diantaranya diagram kelas, diagram komponen, diagram penempatan, diagram objek, diagram paket, diagram profil, diagram struktur komposit.

Diagram struktural mewakili kerangka kerja untuk sistem, dan kerangka kerja ini adalah tempat di mana semua komponen lain ada. Oleh karena itu, diagram kelas, diagram komponen dan diagram penempatan adalah bagian dari pemodelan struktural. Diagram tersebut mewakili elemen dan mekanisme untuk mengumpulkannya. Karena sifatnya yang statis, pada kelompok diagram struktural ini, diagram kelas adalah diagram yang paling banyak digunakan.

6.2.2 Diagram Tingkah Laku

Diagram tingkah laku *(Behavioral Diagrams)* ini merupakan jenis diagram dinamis. Diagram ini menggambarkan interaksi dalam sistem. Ini mewakili interaksi antara diagram struktural. Diagram tingkah laku menunjukkan sifat dinamis dari sistem. Yang termasuk kedalam diagram ini diantaranya diagram aktivitas, diagram interaksi, *use case diagram*, diagram urutan, diagram keadaan, diagram komunikasi, diagram ikhtisar interaksi, dan, diagram waktu.

6.2.3 Diagram Arsitektur

Diagram arsitektur mewakili kerangka keseluruhan sistem. Ini berarti, diagram struktural dan diagram tingkah laku memungkinkan menjadi satu-kesatuan yang sistem perangkat lunak. Diagram arsitektur dapat didefinisikan sebagai cetak biru *(blueprint)* dari keseluruhan sistem. Diantara banyaknya jenis diagram, diagram paket *(package diagram)* termasuk di dalam kelompok diagram ini, meskipun dia juga memungkinkan ada pada diagram struktural.

6.3 Menggunakan Diagram

UML identik dengan pemodelan diagram, dimana baik level konsep, spesifikasi, maupun implementasi dituangkan dalam bentuk diagram. Tiga jenis pemodelan diagram telah disajikan diawal, yang dapat digunakan sesuai kebutuhan bergantung kepada level mana yang ingin disampaikan. Beberapa diantara pemodelan tersebut akan dibahan pada bagian ini.

6.3.1 Diagram Kelas

Diagram kelas atau *Class Diagram* pada dasarnya adalah representasi grafis dari tampilan statis sistem dan mewakili berbagai aspek aplikasi. Sementara kumpulan diagram kelas mewakili keseluruhan sistem. Dan diagram kelas juga merepresentasikan abstraksi yang menentukan struktur umum dan perilaku suatu objek.

a) Struktur dan sintak diagram kelas

Struktur Diagram kelas terdiri dari tiga bagian utama, yakni nama kelas, atribut, dan operasi atau tingkahlaku. Diagram kelas direpresentasikan pada gambar berikut :

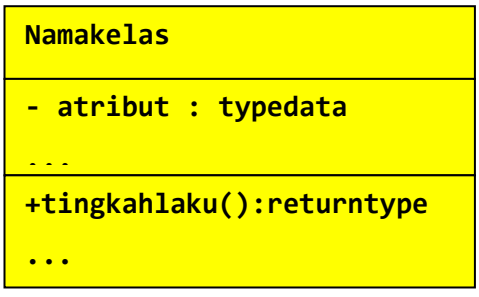


Diagram kelas memungkinkan memiliki hubungan dengan kelas yang lain, terutama pada sistem yang besar dengan banyak kelas terkait, dengan kelas-kelas dikelompokkan bersama untuk membuat diagram kelas. Hubungan antar kelas bisa saja berbeda, dan umumnya ditunjukkan dengan simbol panah yang menuju kelas tertentu.

b) Notasi sintaks

Didalam merancang diagram kelas, perancang harus memperhatikan notasi sintak, yang merujuk kepada beberapa syarat penting seperti *modifier*, sifat, dan tipe atribut maupun method. Modifier merupakan status atribut maupun method apakah mereka dibentuk secara public, private atau protected. Sementara sifat atribut dan method merujuk kepada sifat statis atau dinamis. Suatu atribut mengharuskan suatu tipe dibentuk sesuai batasan dan kriterianya, sementara method memungkinkan memiliki tipe nilai pengembalian *(return type)* atau tidak sama sekali *(void).* Suatu modifier memiliki keunggulan tersendiri, secara detil disampaikan pada tabel berikut :

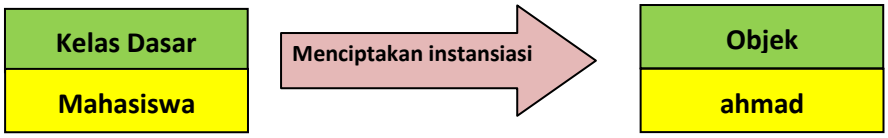
Modifier	Simbol	Hak akses
Public	+	Dapat diakses oleh objek diluar kelas
Private	-	Hanya dapat diakses oleh objek di kelas yang sama
Protected	#	Hanya dapat diakses oleh objek di kelas yang sama dan kelas-kelas turunannya
No modifier		Pleksibel, dan umumnya dapat di deklarasi secara global

Untuk merancang diagram kelas, mengikuti aturan-aturan diantaranya :

1) Secara struktur, nama kelas berada dikotak pertama, atribu dikotak kedua dan operasi terletak di kotak ketiga
2) Nama kelas harus bermakna, dan umumnya menggunakan awalan huruf kapital
3) Tidak menggunakan simbol spasi, dan direkomendasika tidak terlalu panjang
4) Atribut dan method harus diidentifikasi dengan jelas
5) Jenis data (tipe) atribut diberikan setelah titik dua (atribut : tipedata)
6) Setiap elemen dan hubungannya harus diidentifikasi terlebih dahulu
7) Jumlah properti harus ditentukan seminimal mungkin karena semakin banyak properti akan membuat diagram menjadi rumit
8) Mudah dimengerti oleh pembuat program atau penulis kode.

Selanjutnya berikut ini akan disajikan contoh perancangan diagram kelas. Seperti yang disampaikan diatas, atribut dan method harus diidentifikasi terlebih dahulu. Kita ambil contoh . *Misal, seorang mahasiswa memiliki status - nim, nama, kelas, serta memiliki aktivitas (tingkah laku) menginput data, mencetak data dan merevisi data. Objek dari kasus*

ini akan diturunkan dari kelasnya. Agar lebih mudah memahaminya, perhatikan proses instansiasi objek yang diturunkan dari suatu kelas berikut ini :



Selanjutnya perhatikan daftar properti dan method berikut merujuk kepada contoh kasus diatas :

Properti	Method
nim	inputdata
nama	revisidata
kelas	cetakdata

Nilai Properti	Method
00611033	inputdata
Ahmad Muzakir	revisidata
Algoritma1	cetakdata

Daftar properti dan method diatas, kemudian dapat dirancang dalam bentuk diagram kelas sesuai dengan notasi-notasi yang berlaku. Perlu diingat bahwa kelas mewakili konsep yang merangkum keadaan (state yang diwakili oleh atribut-atribut) dan perilaku (operasi). Setiap atribut memiliki tipe. Setiap operasi memiliki tanda, sementara nama kelas adalah satu-satunya informasi wajib. Setelah mengetahui ini maka diagram kelas untuk contoh diatas meliputi rancangan seperti berikut ini :

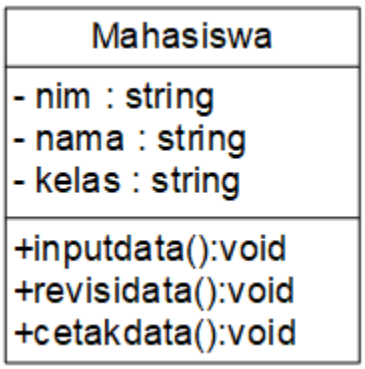
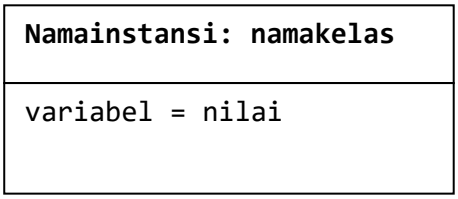


Diagram kelas diatas, selanjutnya dapat diterjemahkan ke dalam bahasa pemrograman, umpama dalam bahasa java berikut ini :

6.3.2 Diagram Objek

Diagram objek *(object diagram)* disebut juga dengan instansi diagram, yang menggambarkan keadaan nyata. Diagram objek menunjukkan bagaimana suatu sistem akan

terlihat seperti pada waktu tertentu *(runtime)*. Hal ini karena diagram objek menyertakan data sebagai proses instansiasi, yang berlaku untuk menjelaskan hubungan kompleks antara objek. Secara grafis, diagram objek mirip dengan diagram kelas, hanya saja diagram objek hanya terdiri dari nama instansi (objek), nama kelas, dan instansinya. Secara struktur objek ditulis seperti pada format diagram objek berikut ini :

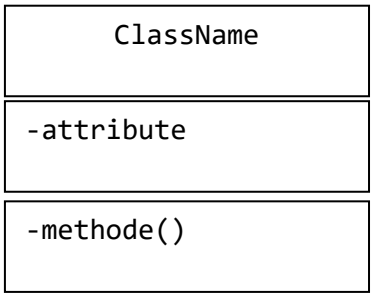


MODUL 7

CLASS, OBJECT, DAN METHOD

7.1 Class

Java merupakan Bahasa pemrograman Berorientasi Objek, karenanya dia mendukung fitur-fitur yang umum disebutkan dengan istilah-istilah : *Abstraction, Encapsulation, Class, Object, Instance, Method, Inheritance, Polymorphism,* dan *Message Parsing.* Begitu banyak fitur yang didukungnya kita perlu membahas satu-persatu. Pada tahap ini akan dibahas praktis Class, Object, dan Methode terlebih dahulu. Class (kelas) merupakan blok kode program yang memiliki beberapa komponen yang saling membutuhkan. Komponen tersebut berupa data (attribute), objek, dan operasi (method). Baik attribut maupun visibily methodnya dapat diatur secara public, private ataupun protected. Secara visual, class dan komponen-komponen miliknya dibentuk dalam ***class diagram*** seperti pada struktur berikut :



Struktur class diagram diatas menjelaskan bahwa :

1) ClassName adalah nama class
2) Attribute merupakan variabel yang diatur sesuai dengan tipe yang dimiliki
3) Method() adalah operasi atau tingkahlaku yang mungkin terjadi didalam blok class.

Didalam struktur class diagram visibility attribut dan method ditentukan sesuai kebutuhan dan maksud perancangan. Visibility merupakan tingkat hak akses yang diberikan kepada attribut dan method. Visibility ini memungkinkan kedua komponen tersebut dapat digunakan secara bebar atau tidak. Secara umum visibility attribut dan method di dalam class dapat terdiri dari beberapa diantaranya sebagai berikut :

VISIBILITY	SIMBOL	HAK AKSES
Friendly	Tidak didefenisikan	Dapat diakses di subclass yang sama yang beradap ada package yang sama pula
Public	+	Semua class dapat mengakses variabel dan method yang visibility modifiernya public

Private	-	Variabel dan method dengan visibility private tidak dapat diakses oleh class lain.
Protected	#	Visibility jenis ini attribut dapat diakses oleh method-method pada class-class tertentu

Dalam prakteknya, bahasa Java mendukung prinsip class yang menjadi ciri khas pemrograman berorientasi objek. Class dalam bahasa Java ditulis dengan menggunakan struktur berikut :

```
<modifier> class NamaClass {
   // badan class
}
```

Berdasarkan struktur class diatas, penulisan class mengacu kepada aturan-aturan berikut :

1) Modifier dapat dibentuk secara ***public*** maupun default
2) Class adalah keyword yang wajib disertakan
3) NamaClass disebut juga dengan identifier, dan penulisannya diawali dengan huruf kapital; tidak boleh diawali dengan angka; tidak boleh mengandung spasi; dan sebaiknya NamaClass tidak terlalu panjang
4) // badan class adalah blok (bagian) yang digunakan untuk menulis kode program. Kode program dapat berupa pendeklarasian (instansiasi) baik attribute (variabel), method, konstruktor, maupun instruksi lainnya.
5) Class dibuka dan ditutup dengan tanda kurung kurawal `{ //badan class }`

Secara umum class adalah antarmuka darimana suatu object dibuat sesuai dengan rancangan classnya. Misal jika classnya adalah kelas Hewan, tentu objectnya bisa berupa buruh, dan harimau, kucing dan lain sebagainya. Aturan-aturan penulisan class seperti yang diuraikan diatas, maka penciptaan class dapat dicontohkan seperti berikut :

```
class Manusia {
int usia;
float berat;
float tinggi;
 }
```

Untuk kelas yang dibentuk secara public, dapat ditulis seperti contoh berikut :

7.2 Attribute

Pada dasarnya *Attribute* sudah disinggung pada modul yang membahas tentang *variabel, tipedata* dan *operator*. Attribute merupakan variabel yang *visibility*-nya dapat dibentuk sebagai *public, private* atau *protected* sesuai dengan kebutuhan dan maksud perancangan. Peletakan attribut secara umum ditempatkan pada class, tetapi juga dapat ditempatkan di dalam badan method. Tetapi direkomendasikan pada tubuh class supaya dapat diakses oleh entitas lain. Tetapi aksesibilitas attribute bergantung kepada visibilitas yang diberikan kepada attribut. Penggunaan attribute mengacu kepada sintaksis berikut ini :

```
<modifier> tipetada namaattibute;
```

Atau dengan inisialisasi nilai attribute seperti struktur umum berikut :

```
<modifier> tipetada namaattibute=nilaiattribute;
```

Contoth

```
public int usia;
private string id=”admin123”;
float tinggi = 167.89;
```

7.3 Object

Object merupakan komponen turunan dari class. Suatu object memiliki beberapa karakteristik penting, yaitu : 1) Status; 2) Perilaku; 3) Identitas. ***Status*** object merepresentasikan data atau nilai yang dimilikinya seperti objek bola yang memiliki status (nilai) warna, berat, bentuk, sementara ***Perilaku*** object mewakili tingkah laku, fungsionalitas, operasi suatu object, seperti object bola yang memiliki perilaku memantul, menggelinding, dan berhenti. Sedangkan ***Identitas*** merupakan sesuatu yang dimiliki oleh object, yang diperoleh atau diterapkan melalui status object, seperti identitas yang dimiliki oleh bola seperti, “hitam”, 1.8 Kg, dan “bulat”.

Identitas object digunakan secara internal oleh JVM untuk mengidentifikasi setiap objek secara unik. Di dalam bahasa Java pembentukan *object* dari suatu *class* dibentuk dengan mengikuti struktur berikut :

```
NamaClass namaobject = new NamaClass();
```

Contoh

```
Manusia tubuh = new Manusia();
```

7.4 Method

Dalam beberapa pemrograman, method identik dengan fungsi dan procedur terutama seperti pada pemrograman pascal yang memiliki kedua fitur tersebut. Method didalam permograman objek adalah parameter yang berfungsi untuk menerapkan tingkahlaku suatu objek. Method dapat dibentuk didalam suatu class, sementara setiap class bisa memiliki beberapa method diamin. Di dalam Java, method dapat ditulis dengan menggunakan struktur berikut ini :

```
<modifier> nilaibaliktipemethod namaMethod (daftar parameter) {
        // badan method
}
```

Dimana,

- Modifier : sama seperti modifier pada class dan attribute
- nilaibaliktipemethod : memungkinkan method memiliki nilai balik
- namaMethod : nama dari method yang digunakan
- daftar parameter : attribute (variabel) dan tipe datanya, attribute bisa tunggal bisa lebih lebih dari satu variabel
- // badan method : merupakan bagian dimana ekspresi method dituliskan

7.5 Konstruktor

Constructor atau konstruktor digunakan untuk melakukan inisialisasi variable-variabel instan class serta melakukan persiapan-persiapan yang diperlukan oleh suatu objek untuk dapat beroperasi dengan baik. Untuk menggunakan konstruktor kita perlu mengikuti aturan berikut :

1) Nama konstruktor harus sama dengan nama *class*-nya
2) Menyertakan modifier didepan nama konstruktor
3) Tidak harus memiliki nilai balik tipe
4) Konstruktor bukanlah *abstract*, *static*, *final*, dan *synchronized*

Konstruktor digunakan setiap kali objek diciptakan, paling tidak satu konstruktor dipanggil untuk menetapkan nilai awal ke anggota data dari kelas yang sama. Pemanggilan konstruktor umumnya menggunakan keyword **new.** Dalam bahasa Java, konstruktor dibagi dalam dua jenis konstruktor, yaitu :

1) Konstruktor *Default*

Dikatakan konstruktor default jika konstruktor tidak memiliki argumen berupa parameter-parameter konstruktor. Untuk class yang tidak disertai dengan konstruktor, secara default kompiler akan memberikan konstruktor default. Penggunaan konstruktor default mengacu pada sintaksis berikut :

```
ClassName() {
            // badan konstruktor
            }
```

2) Konstruktor dengan parameter

Konstruktor dengan parameter *(Parameterized Constructor)* dapat diterapkan untuk memberikan nilai yang berbeda pada suatu objek. Konstruktor dengan parameter dapat digunakan dengan struktur berikut :

```
ClassName(daftar parameter) {
   Parameter1;
   Parameter2;
}
```

7.6 Enkapsulasi

Di dalam bahasa Java, Enkapsulasi merupakan mekanisme membungkus data (variabel / attribut) dan tingkahlaku (method) bersama sebagai satu kesatuan. Dalam enkapsulasi, variabel class akan disembunyikan dari class lain, dan hanya dapat diakses melalui method kelas mereka sendiri. Keadaan ini membuat enkapsulasi disebut sebagai proses menyembunyikan data. Di dalam Java, enkapsulasi dibentuk dengan: (1) deklarasi attribute class secara private, dan (2) membentuk method ***setter*** (mengatur) dan ***getter*** (mengambil) secara public untuk memungkinkan dapat memodifikasi dan melihat data (variabel). Misal, suatu class memiliki attribut (variabel) nama dan nilai, maka pada class tersebut harus ada method **setNama()** dan **getNama()** serta **setNilai()** dan **getNilai().** Ini merupakan karakteristik dalam penerapan enkapsulasi. Untuk menerapkan enkapsulasi di dalam Java menggunakan struktur dengan modifier attribute seperti berikut :

```
class ClassName{
    private tipe namaattribute1;
```

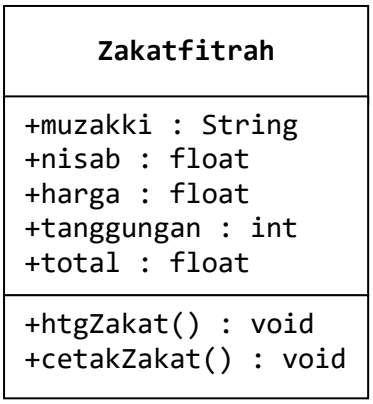
```
    private tipe namaattribute2;
}
```

7.7 Kata Kunci This

Kata kunci (*keyword)* **this** pada bahasa Java dapat digunakan di dalam methode atau konstruktor class. Kata kunci ***this*** bekerja sebagai referensi ke object saat ini, yang method atau konstruktornya sedang dipanggil. Kata kunci ini dapat digunakan untuk merujuk ke setiap anggota objek saat ini dari dalam instansi method atau konstruktor. Untuk menggunakan keyword this dapat menggunakan struktur **this.variable;**

7.8 Menerapkan *Class, Object,* dan *Method*

Setelah memahami konsep class dan komponen-komponen yang termasuk dalam anggota class diatas, mari kita terapkan satu-persatu. Ikutilah langkah-langkah praktikum untuk memudahkan memahami konseptual dan praktisnya.



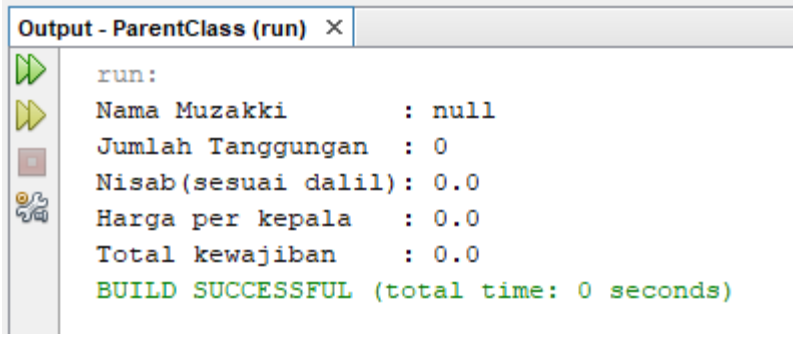
Sekarang kita telah memiliki class diagram Zakatfitrah yang memiliki beberapa data diantaranya muzakki (orang yang wajib zakat), tanggungan (jumlah tanggungan per kepala rumah tangga yang wajib dibayar), nisab (banyaknya Kilogram beras yang ditentukan berdasarkan dalil Al qur'an dan Sunnah = 2.7 Kg), dan harga (per kepala diambil dari harga beras/Kg x nisab). Dari class diagram tersebut, kita akan mencoba menerapkannya kedalam program Java bagaimana menghitung beban Zakat yang harus dibayarkan oleh kepala rumah tangga sebagai muzakki. Untuk itu ikutilah langkah-langkah praktikum sebagai berikut :

a. Memciptakan Class

1) Jalankan Program NetBean, lalu buat project baru dengan nama Zakatfitrah
2) Hapus seluruh kode yang muncul di editor, selanjutnya ketik ulang kode program berikut

```
1.  package zakat;
2.  class Zakatfitrah{
3.      String muzakki;
4.      float nisab;
5.      float hrgperkepala;
6.      int tanggungan;
7.      float total;
8.
9.      public static void main(String [] args){
10.         Zakatfitrah fitrah = new Zakatfitrah();
11.         System.out.println("Nama Muzakki       : "+fitrah.muzakki);
12.         System.out.println("Jumlah Tanggungan  :
    "+fitrah.tanggungan);
13.         System.out.println("Nisab(sesuai dalil): "+fitrah.nisab);
14.         System.out.println("Harga per kepala   :
    "+fitrah.hrgperkepala);
15.         System.out.println("Total kewajiban    : "+fitrah.total);
16.     }
17. }
```

3) Simpan perubahan program, Ctrl + S
4) Jalankan program tekan Run Project (F6)
5) Jika program berhasil maka akan menampilkan keluaran sebagai berikut



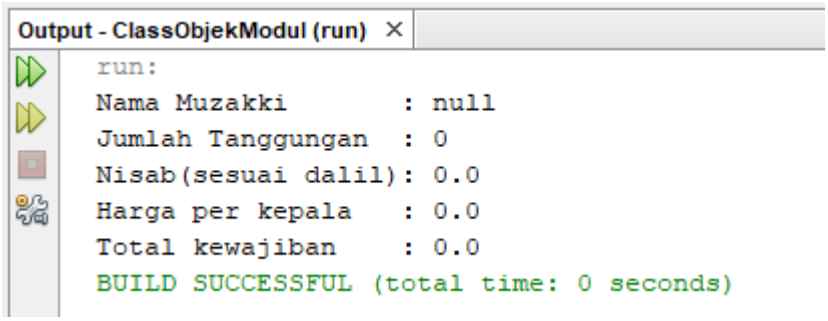
Perhatikan keluaran program diatas, tampak bahwa seluruh data masih kosong. Perhatikan juga bahwa kode program diatas terdiri dari satu class, dimana baik attribute maupun method (main) berada dalam satu kelas yang sama. Sekarang kita coba dengan memisahkan antara data dengan objek di class yang berbeda.

6) Buat class baru dengan nama Muzakki, sempurnakanlah kode program seperti pada kolom berikut ini

```
1.  package zakat;
2.  class Muzakki {
3.      String muzakki;
4.      int tanggungan;
5.      float nisab;
6.      float hrgperkepala;
```

```
7.      float total;
8.  }
9.
10. public class Zakatfitrah{
11.     public static void main(String [] args){
12.         Muzakki fitrah = new Muzakki();
13.
14.
    fitrah.total=(fitrah.nisab*fitrah.hrgperkepala)*fitrah.tanggunga
    n;
15.         System.out.println("Nama Muzakki       :
    "+fitrah.muzakki);
16.         System.out.println("Jumlah Tanggungan  :
    "+fitrah.tanggungan);
17.         System.out.println("Nisab(sesuai dalil):
    "+fitrah.nisab);
18.         System.out.println("Harga per kepala   :
    "+fitrah.hrgperkepala);
19.         System.out.println("Total kewajiban    :
    "+fitrah.total);
20.     }
21. }
```

7) Simpan perubahan program kemudian jalankan kembali, jika berhasil maka keluaran akan seperti pada gambar berikut



Setelah beberapa percobaan yang kita lakukan, keluaran program masih kosong. Kenapa bisa demikian ? Tentu saja, karena attribut yang digunakan belum diberikan nilai (nilai belum diinisialisasi). Selanjutnya kita akan mencoba menginisialisasi nilai objek.

b. Inisialisasi

Inisialisasi merupakan proses memberikan nilai kepada objek melalui variabel-variabel (attribute) class. Dalam prakteknya, inisialisasi nilai objek dapat dilakukan melalui 3 cara : 1) Inisialisasi dengan Variable References; 2) Dengan Method; 3) Dengan konstruktor

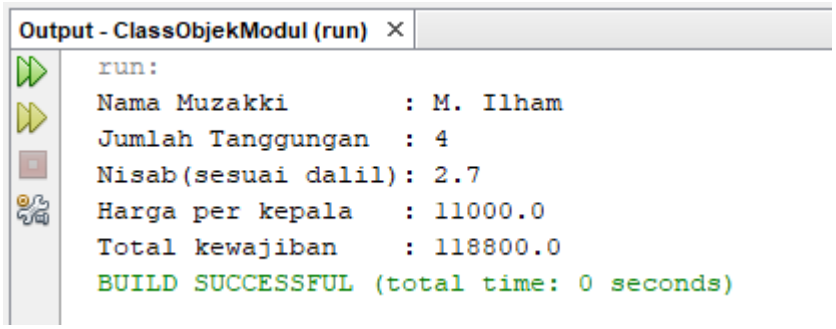
1. Insialisasi Object Melalui *Variable References*

Untuk menginisialisasi nilai objek dengan cara ini (*references*) dapat dilakukan melalui pendeklarasi attribute dan tipe di class global, dan dapat pula dilakukan melalui badan main method. Untuk menginisialisasi nilai object melalui *Variable Reference* dapat mengikuti lakukan langkah-langkah praktikum berikut ini :

1) Buat file baru, klik Menu File - New File
2) Pada kolom *Categories* pilih **Java**, pada kolom *File Tipes* pilih **Java Class**
3) Pilih Next, lalu ganti NewClass menjadi Muzakki, lalu tekan Finish
4) Hapus seluruh kode program yang ada, lalu ketik ulang kode program berikut

```
1.  package zakat;
2.  class Muzakki {
3.      String muzakki;
4.      int tanggungan;
5.      float nisab;
6.      float hrgperkepala;
7.      float total;
8.  }
9.
10. public class Zakatfitrah{
11.     public static void main(String [] args){
12.         Muzakki fitrah = new Muzakki();
13.         fitrah.muzakki="M. Ilham";
14.         fitrah.tanggungan = 4;
15.         fitrah.nisab = (float) 2.7;
16.         fitrah.hrgperkepala = 11000;
17.
    fitrah.total=(fitrah.nisab*fitrah.hrgperkepala)*fitrah.tanggunga
    n;
18.         System.out.println("Nama Muzakki       :
    "+fitrah.muzakki);
19.         System.out.println("Jumlah Tanggungan  :
    "+fitrah.tanggungan);
20.         System.out.println("Nisab(sesuai dalil):
    "+fitrah.nisab);
21.         System.out.println("Harga per kepala   :
    "+fitrah.hrgperkepala);
22.         System.out.println("Total kewajiban    :
    "+fitrah.total);
23.     }
24. }
```

5) Simpan perubahan program, Ctrl + S
6) Kompilasi dan jalankan program, jika program berhasil maka akan menampilkan keluaran seperti pada gambar berikut

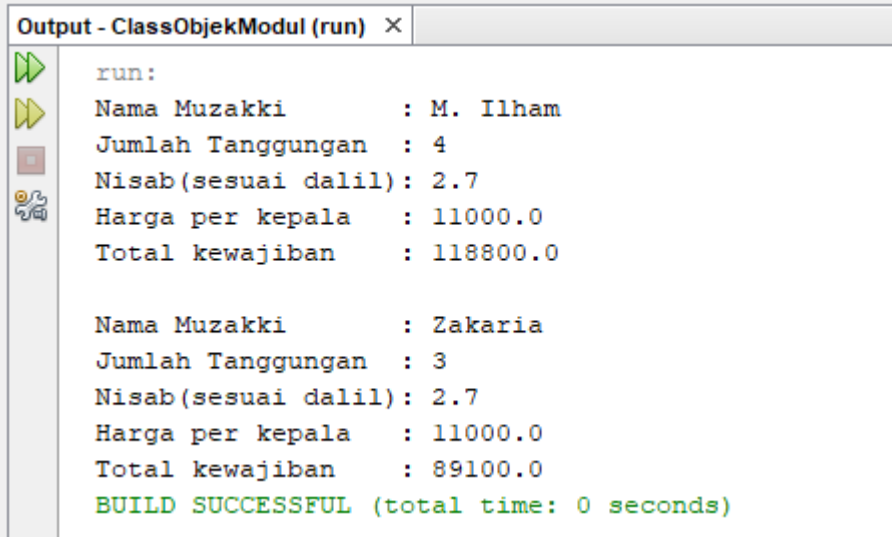


7) Jika kita ingin menambahkan object baru, kita cukup melakukan hal yang sama seperti pada program diatas, diantaranya seperti berikut :

```
1.  package zakat;
2.  class Muzakki {
3.      String muzakki;
4.      int tanggungan;
5.      float nisab;
6.      float hrgperkepala;
7.      float total;
8.  }
9.
10. public class Zakatfitrah{
11.     public static void main(String [] args){
12.         Muzakki fitrah = new Muzakki();
13.         Muzakki zakki = new Muzakki();
14.         fitrah.muzakki="M. Ilham";
15.         fitrah.tanggungan = 4;
16.         fitrah.nisab = (float) 2.7;
17.         fitrah.hrgperkepala = 11000;
18.
    fitrah.total=(fitrah.nisab*fitrah.hrgperkepala)*fitrah.tanggungan
    ;
19.         System.out.println("Nama Muzakki       :
    "+fitrah.muzakki);
20.         System.out.println("Jumlah Tanggungan  :
    "+fitrah.tanggungan);
21.         System.out.println("Nisab(sesuai dalil): "+fitrah.nisab);
22.         System.out.println("Harga per kepala   :
    "+fitrah.hrgperkepala);
23.         System.out.println("Total kewajiban    : "+fitrah.total);
24.
25.         // Inisialisasi object yang lain
26.         zakki.muzakki = "Zakaria";
27.         zakki.tanggungan = 3;
28.         zakki.nisab = (float) 2.7;
29.         zakki.hrgperkepala = 11000;
30.
    zakki.total=(zakki.nisab*zakki.hrgperkepala)*zakki.tanggungan;
31.         System.out.println("\nNama Muzakki       :
    "+zakki.muzakki);
32.         System.out.println("Jumlah Tanggungan  :
```

```
    "+zakki.tanggungan);
33.         System.out.println("Nisab(sesuai dalil): "+zakki.nisab);
34.         System.out.println("Harga per kepala   :
    "+zakki.hrgperkepala);
35.         System.out.println("Total kewajiban    : "+zakki.total);
36.     }
37. }
```

8) Simpan perubahan program, dan jalankan kembali program. Jika program sukses, keluaran akan seperti pada gambar berikut



2. Insialisasi Object Melalui Method

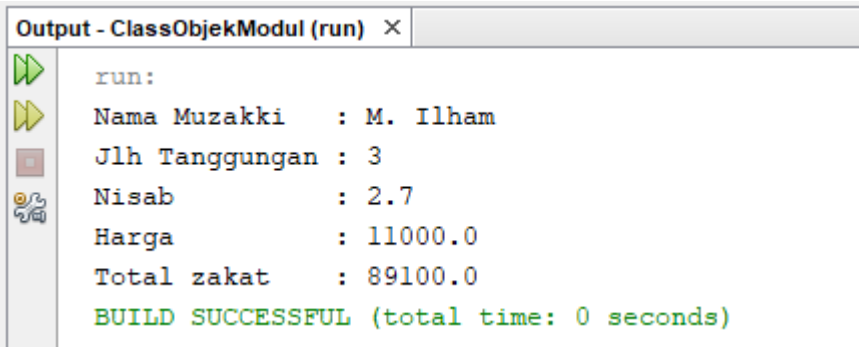
Selain melalui variabel referensi, inisialisasi nilai object juga dapat dilakukan melalui method. Inisialisasi nilai object dengan method dapat mengikuti langkah-langkah praktikum berikut :

1) Buat file baru, klik Menu File - New File
2) Pada kolom *Categories* pilih **Java**, pada kolom *File Tipes* pilih **Java Class**
3) Pilih Next, lalu ganti NewClass menjadi Datamuzakki, lalu tekan Finish
4) Hapus seluruh kode program yang ada, lalu ketik ulang kode program berikut

```
1.  package zakat;
2.  class Datamuzakki {
3.      String muzakki;
4.      int tanggungan;
5.      float nisab;
6.      float hrgperkepala;
7.      float total;
8.
9.      void htgZakat(String zakki, int tgg, float nsb, float
    hrg, float ttl){
10.         muzakki = zakki;
11.         tanggungan = tgg;
```

```
12.         nisab = nsb;
13.         hrgperkepala = hrg;
14.         total = ttl = (nsb*hrg)*tgg;
15.     }
16.
17.     void cetakData(){
18.         System.out.println("Nama Muzakki   : "+muzakki);
19.         System.out.println("Jlh Tanggungan : "+tanggungan);
20.         System.out.println("Nisab          : "+nisab);
21.         System.out.println("Harga          : "+hrgperkepala);
22.         System.out.println("Total zakat    : "+total);
23.     }
24. }
25.
26. public class Zakatfitrah{
27.     public static void main(String [] args){
28.     Datamuzakki fitrah = new Datamuzakki();
29.
30.     fitrah.htgZakat("M. Ilham", 3, (float) 2.7, 11000, 0);
31.     fitrah. cetakZakat();
32.     }
33. }
```

5. Simpan perubahan program, selanjutnya jalankan maka akan menampilkan hasil yang sama seperti data pada keluaran program diatas



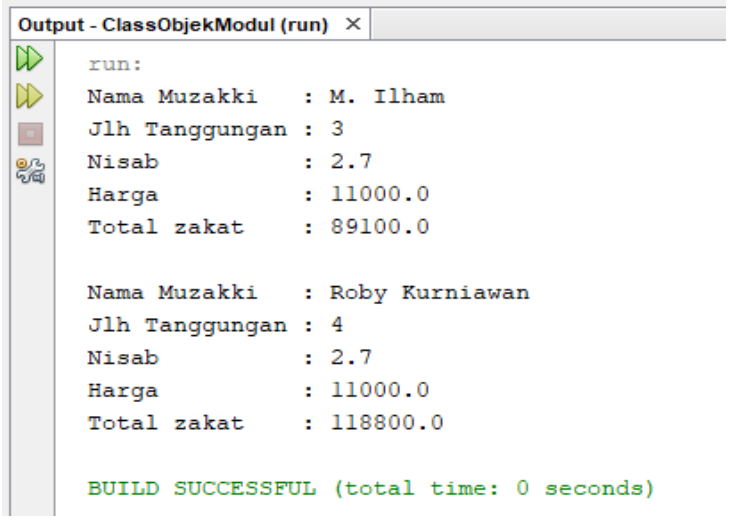
3. Inisialisasi Object Melalui *Constructor*

Ini merupakan cara yang juga sering digunakan oleh perancang program terutama pada bahasa Java. Inisialisasi nilai object melalui konstruktor ini mirip dengan cara melalui method, hanya saja cara ini menggunakan nama class yang memimiliki data. Ada beberapa aturan penggunaan konstruktor seperti yang telah dijelaskan diatas. Langkah-langkah praktikumnya bisa diikuti seperti berikut :

1) Buat file baru, klik Menu File - New File
2) Pada kolom *Categories* pilih **Java**, pada kolom *File Tipes* pilih **Java Class**
3) Pilih Next, lalu ganti NewClass menjadi Datamuzakki, lalu tekan Finish
4) Hapus seluruh kode program yang ada, lalu ketik ulang kode program berikut

```
1.  package zakat;
2.  public class Zakatfitrah {
3.      String muzakki;
4.      int tanggungan;
5.      float nisab;
6.      float hrgperkepala;
7.      float total;
8.
9.      Zakatfitrah(String zakki, int tgg, float nsb, float hrg,
    float ttl){
10.         this.muzakki=zakki;
11.         this.tanggungan = tgg;
12.         this.nisab = nsb;
13.         this.hrgperkepala= hrg;
14.         this.total = ttl =tgg * (nsb * hrg);
15.     }
16.
17.     void cetakZakat(){
18.         System.out.println("Nama Muzakki   : "+muzakki);
19.         System.out.println("Jlh Tanggungan : "+tanggungan);
20.         System.out.println("Nisab          : "+nisab);
21.         System.out.println("Harga          : "+hrgperkepala);
22.         System.out.println("Total zakat    : "+total);
23.         System.out.println();
24.     }
25.
26.     public static void main (String [] args){
27.         Zakatfitrah fitrah = new Zakatfitrah("M. Ilham", 3,
    (float) 2.7, 11000, 0);
28.         Zakatfitrah fitrah1 = new Zakatfitrah("Roby Kurniawan",
    4, (float) 2.7, 11000, 0);
29.         fitrah.cetakZakat();
30.         fitrah1.cetakZakat();
31.     }
32. }
```

5) Simpan perubahan program, dan jalankan. Hasilnya dapat dilihat seperti pada gambar



4. Konstruktor Overloading

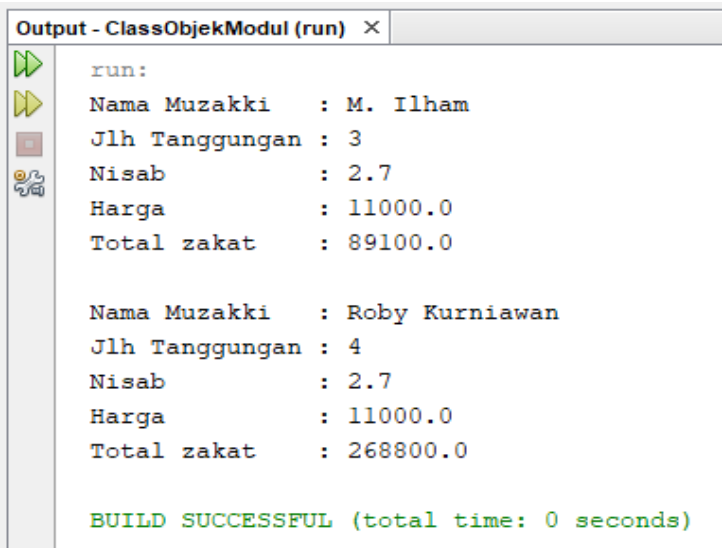
Ini merupakan variasi dari inisialisasi object melalui konstruktor, dimana konstruktor overloading persis seperti pada method overloading, dimana konstruktor yang sama di dalam satu kelas tetapi memiliki perilaku yang berbeda. Untuk lebih jelasknya, lakukan praktikum berikut agar lebih mudah memahaminya.

1) Buat file baru, klik Menu File - New File
2) Pada kolom *Categories* pilih **Java**, pada kolom *File Tipes* pilih **Java Class**
3) Pilih Next, lalu ganti NewClass menjadi Datamuzakki, lalu tekan Finish
4) Hapus seluruh kode program yang ada, lalu ketik ulang kode program berikut

```
1.  package zakat;
2.  public class Zakatfitrah3 {
3.      String muzakki;
4.      int tanggungan;
5.      float nisab;
6.      float hrgperkepala;
7.      float total;
8.
9.      Zakatfitrah3(String zakki, int tgg, float nsb, float hrg,
    float ttl){
10.         this.muzakki=zakki;
11.         this.tanggungan = tgg;
12.         this.nisab = nsb;
13.         this.hrgperkepala= hrg;
14.         this.total = ttl =tgg * (nsb * hrg);
15.     }
16.
17.       Zakatfitrah3(String zakki, int tgg, float nsb, float hrg,
    float sedekah, float ttl){
18.         this.muzakki=zakki;
19.         this.tanggungan = tgg;
20.         this.nisab = nsb;
21.         this.hrgperkepala= hrg;
22.         this.total = ttl =tgg * (nsb * hrg) + sedekah;
23.     }
24.     void cetakZakat(){
25.         System.out.println("Nama Muzakki   : "+muzakki);
26.         System.out.println("Jlh Tanggungan : "+tanggungan);
27.         System.out.println("Nisab          : "+nisab);
28.         System.out.println("Harga          : "+hrgperkepala);
29.         System.out.println("Total zakat    : "+total);
30.         System.out.println();
31.     }
32.
33.     public static void main (String [] args){
34.         Zakatfitrah3 fitrah = new Zakatfitrah3("M. Ilham", 3,
    (float) 2.7, 11000, 0);
35.         Zakatfitrah3 fitrah1 = new Zakatfitrah3("Roby
    Kurniawan", 4, (float) 2.7, 11000, 150000,0);
```

```
36.          fitrah.cetakZakat();
37.          fitrah1.cetakZakat();
38.      }
39.  }
```

5) Simpan perubahan program, dan jalankan, maka keluaran program akan seperti pada gambar berikut



5. Latihan

Sebuah perusahaan distributor peralatan rumah tangga memiliki beberapa karyawan yang digaji dengan nilai yang sama yaitu Rp. 50000. Setiap karyawan mendapat tambahan dari tunjangan yang berbeda sebagai berikut :

1) Marketing mendapat tunjangan dari penjualan :
   - 20% jika penjualan > 100000,
   - 15% jika penjualan >=50000,
   - 10% jika penjualan >=21000, dibawah nilai 21000 tidak mendapat tunjangan penjualan
2) Akuntan mendapat tunjangan harian 8000, dan tunjangan kinerja 5000 untuk masa kerja >=2 tahun, dan untuk masakan lebih kecil dari itu tidak dapat tunjangan

Rancanglah program untuk menghitung dan menampilkan gaji kedua jenis jabatan karyawan tersebut, dengan mengadobsi teori dan praktis modul 4 ini.

MODUL 8

PEWARISAN

8.1 Konsep Pewarisan

Pewarisan atau yang sering disebut dengan istilah *inheritance* merupakan suatu mekanisme dimana subclass (kelas anak) dapat mewarisi sifat-sifat yang dimiliki oleh parent class (kelas induk). Pewarisan dapat berupa *attribute* yang dimiliki oleh parent class maupun tingkahlakunya. Pewarisan dibagi dalam beberapa tipe, yakni pewarisan *single*, *multilevel*, *hierarchical*, *multiple*, dan *hybrid*. Di dalam java, pewarisan multiple dan hibrid tidak dapat dilakukan.

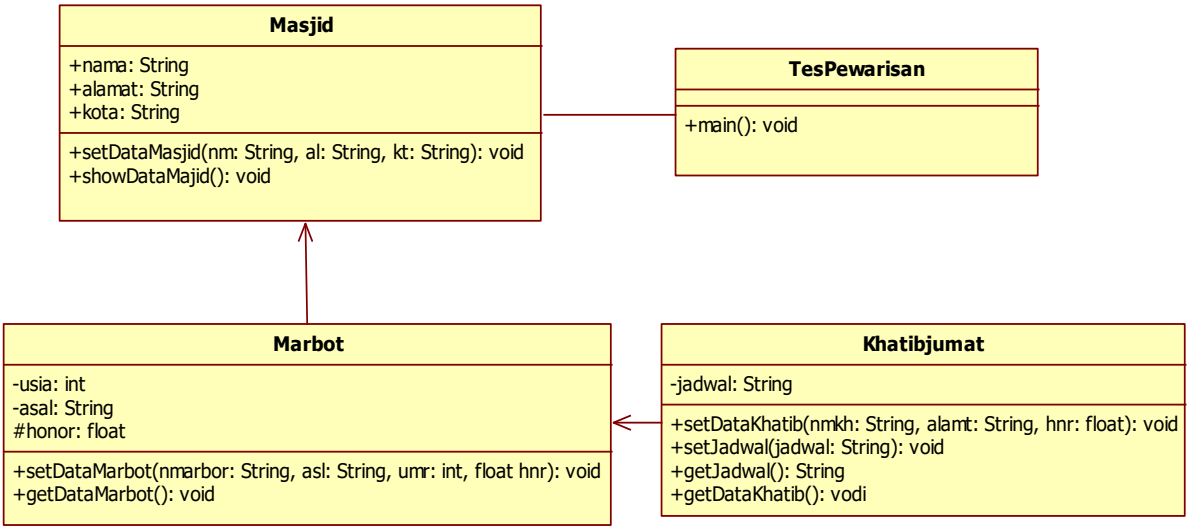
Pewarisan class di dalam bahasa Java ditandai dengan simbol **extends** yang diikuti dengan nama class induk yang mewariskannya. Secara struktur, pewarisan menerapkan struktur sebagai berikut :

```
Subclass extends Superclass {
// tubuh class
}
```

Dimana *subclass* dan *superclass* merupakan nama identifier (nama class) yang mewarisi dan mewariskan sifat-sifat, baik attribute (variabel) maupun tingkahlaku (operasi/method).

8.2 Menerapkan Pewarisan

Setelah mempelajari teori siangkat tentang perawarisan diatas, maka berikut ini diberikan contoh program untuk memudahkan peserta praktikan dalam memahami konsep dan praktis pewarisan class. Dalam contoh ini diberikan beberapa class yang memiliki asosiasi hubungan antar class, yakni class Masjid yang mewariskan sifat-sifatnya kepada class anak yaitu class Marbot dan class Khatibjumat. Untuk lebih jelasnya, silahkan ikuti langkah-langkah praktikum berikut ini, berdasarkan acuan dari diagram class berikut :



1) Buka NetaBean, lalu buat project baru dengan nama TesPewarisan
2) Selanjutnya tambahkan file baru, klik File - New File, pada kolom Categori pilih Java, dan pada kolom File Type pilih Java Class, Next
3) Ganti NewClass dengan Masjid, selanjutnya hapus seluruh kode yang ada, dan ketik ulang kode berikut

```
1.  package tespewarisan;
2.  class Masjid{
3.      public String nama;
4.      public String alamat;
5.      public String kota;
6.
7.      public void setDataMasjid(String nm, String al, String kt){
8.          this.nama = nm;
9.          this.alamat = al;
10.         this.kota = kt;
11.     }
12.
13.     public void getDataMasjid() {
14.         System.out.println("Nama Masjid  : "+nama);
15.         System.out.println("Alamat       : "+alamat);
16.         System.out.println("Kota         : "+kota);
17.   }
18. }
```

4) Selanjutnya tambahkan kembali file baru, ikuti langkah 2, ganti NewClass dengan nama Marbot

5) Kemudian hapus semua kode yang ada, dan tulis kembali kode program berikut

```
1.  Package tespewarisan;
2.  class Marbot extends Masjid{
3.      private int usia;
4.      private String asal;
5.      protected float honor;
6.
7.      public void setDataMarbot(String nmarbot, String asl, int
    umr, int hnr){
8.          this.nama = nmarbot;
9.          this.asal = asl;
10.         this.usia = umr;
11.         this.honor = hnr;
12.     }
13.
14.     public void getDataMarbot(){
15.         System.out.println("Nama Marbot  : "+nama);
16.         System.out.println("Asal Marbot  : "+asal);
17.         System.out.println("Usia Marbot  : "+usia+ " tahun");
18.         System.out.println("Honor Marbot : "+honor);
19.     }
20. }
```

6) Tambahkan kembali file baru, ikuti langkah nomor 2, dan ganti NewClass dengan nama Khatibjumat

7) Hapus semua kode yang ada, dan tulis ulang kode berikut

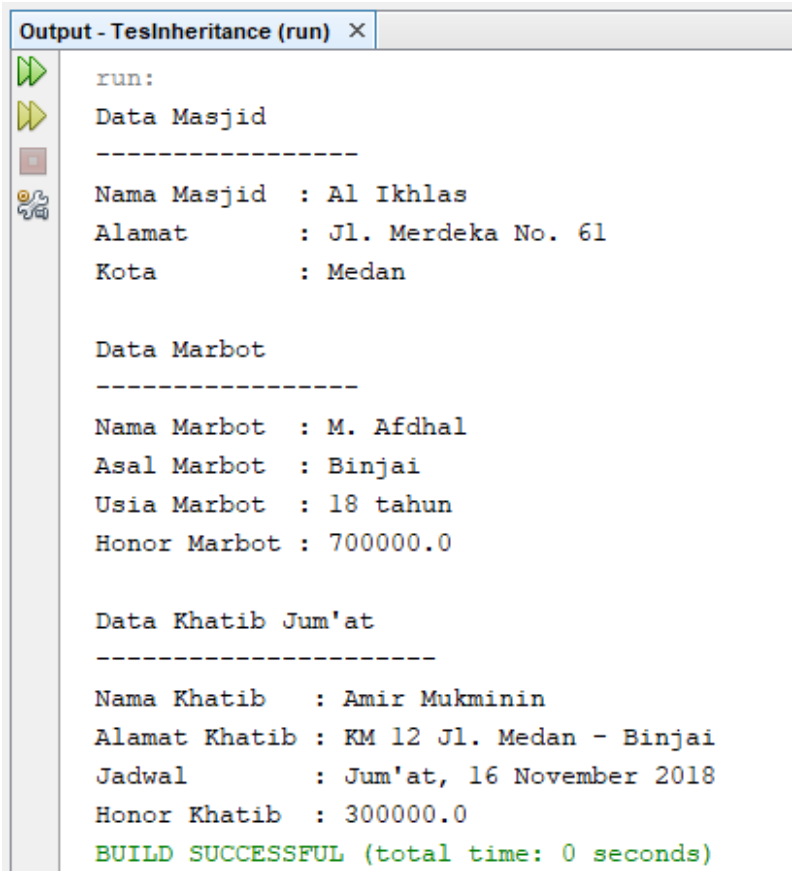
```
1.  package tespewarisan;
2.  class Khatibjumat extends Marbot {
3.      private String jadwal;
4.
5.      public void setDataKhatib(String nmkh, String alamt, float
    hnr){
6.          this.nama = nmkh;
7.          this.alamat = alamt;
8.          this.honor = hnr;
9.      }
10.
11.     public void setJadwal(String jadwal){
12.         this.jadwal = jadwal;
13.     }
14.
15.     public String getJadwal(){
16.         return jadwal;
17.     }
18.
19.     public void getDataKhatib(){
20.         System.out.println("Nama Khatib   : "+nama);
21.         System.out.println("Alamat Khatib : "+alamat);
```

```
22.         System.out.println("Jadwal        : "+jadwal);
23.         System.out.println("Honor Khatib  : "+honor);
24.     }
25. }
```

8) Langkah selanjutnya, buka file TesPewarisan yang sudah dibuat diawal, kemudian sesuaikan kode program berikut

```
1.  package tespewarisan;
2.  public class TesPewarisan {
3.      public static void main(String[] args) {
4.          Masjid masjid = new Masjid();
5.          Marbot marbot = new Marbot();
6.          Khatibjumat khatib = new Khatibjumat();
7.
8.          //Data MasjidClass
9.          System.out.println("Data Masjid");
10.         System.out.println("-----------------");
11.         masjid.setDataMasjid("Al Ikhlas","Jl. Merdeka No. 61",
    "Medan");
12.         masjid.getDataMasjid();
13.         System.out.println();
14.         //Data Marbot
15.         System.out.println("Data Marbot");
16.         System.out.println("-----------------");
17.         marbot.setDataMarbot("M. Afdhal", "Binjai", 18, 700000);
18.         marbot.getDataMarbot();
19.         System.out.println();
20.         //Data Khatib Jum'at
21.         System.out.println("Data Khatib Jum'at");
22.         System.out.println("----------------------");
23.         khatib.setJadwal("Jum'at, 16 November 2018");
24.         khatib.setDataKhatib("Amir Mukminin", "KM 12 Jl. Medan -
    Binjai", 300000);
25.         khatib.getDataKhatib();
26.     }
27. }
```

9) Simpan perubahan semua file, Ctrl + Shift+S, selanjutnya jalankan program. Jika program sukses akan menampilkan keluaran program seperti berikut



8.3 Tugas

Suatu perusahaan distributor alat elektronik memiliki beberapa karyawan yang bertugas sebagai marketing dan kasir, yang sama-sama mendapatkan gaji pokok 500000, kedua karyawan memiliki kesempatan untuk mendapatkan gaji tambahan diantaranya : 1) Marketing yang diberikan 10% dari total penjualan, sementara untuk 2) Kasir mendapatkan gaji tambahan dari jumlah jam kerja lembur. Berdasarkan kasus diatas, rancanglah program untuk menghitung total gaji karyawan tersebut.

MODUL 9

POLIMORPISME

9.1 Konsep Polimorpisme

Didalam pemrograman objek, terutama pada Java polimorpisme (*Polymorphism*) merupakan sebuah konsep yang dengannya kita dapat melakukan satu tindakan dengan cara yang berbeda. Polimorfisme berasal bahasa Yunani yang diambil dari kata *poly* dan *morphs*. Dimana kata *poly* berarti banyak dan *morph* berarti bentuk. Jadi polimorfisme dapat diarti suatu mekanisme dimana objek dapat berperilaku dalam banyak bentuk.

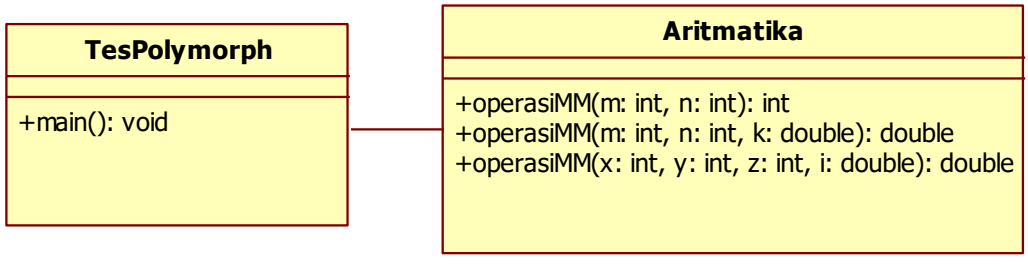
Polimorfisme di Java dapat diterapkan dalam dua cara yakni polimorfisme pada waktu kompilasi dan polimorfisme pada saat *runtime*. Polimorpisme pada waktu kompilasi dapat diterapkan menggunakan method *overloading*, sedangkan polimorpisme pada saat *runtime* dapat diterapkan dengan method *overriding*.

9.1.1 Overloading

Overloading merupakan mekanisme dimana suatu class dapat memiliki beberapa method yang sama namun dengan parameter berbeda. Perbedaan parameternya membuat objek dapat berperilaku berbeda dari yang lainnya. Dalam pemrograman berorientasi objek, polimorpisme pada waktu kompilasi dapat menerapkan method overloading ini.

c. Latihan

Untuk memahami konsep polimorpisme dengan method overloading, berikut ini diberikan contoh programnya. Sebelum menjalankan praktikumnya, ada baiknya memperhatikan struktur diagram class yang digambarkan pada bagan berikut ini :



Berdasarkan diagram class diatas, lakukanlah langkah-langkah praktikum berikut :

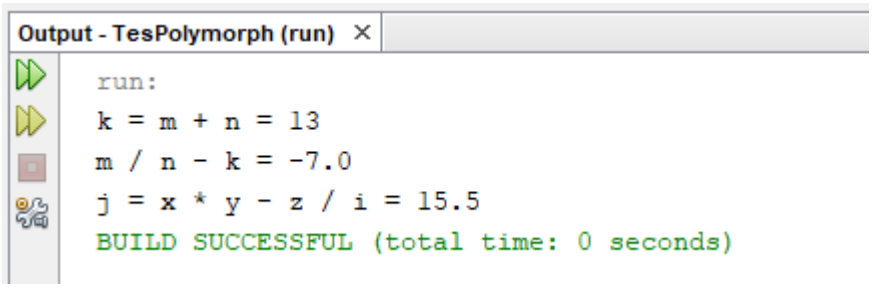
1) Buka NetBean, dan buat Project baru dengan nama TesPolymorph
2) Selanjutnya tambahkan file baru, dengan nama Aritmatika
3) Hapus semua kode yang ada, selanjutnya tulis ulang kode program berikut :

```
1. package tespolymorph;
2. class aritmatika{
3.     public int operasimm(int m, int n){
4.         int k = m + n;
5.         return k;
6.     }
7.
8.     public double operasimm(double m, double n, int k){
9.        return (m / n - k);
10.       }
11.
12.       public double operasimm(int x, int y, int z, double
   i){
13.          double j = x * y - z / i;
14.          return j;
15.       }
16. }
```

4) Simpan perubahan program, selanjutnya pada halaman public class TesPolymorph, ketik program berikut :

```
1.  package tespolymorph;
2.  public class TesPolymorph {
3.      public static void main(String[] args) {
4.          Aritmatika tmb = new Aritmatika();
5.          Aritmatika krg = new Aritmatika();
6.          Aritmatika kl = new Aritmatika();
7.
8.          System.out.println("k = m + n = "+tmb.operasiMM(4,
    9));
9.          System.out.println("m / n - k = "+krg.operasiMM(20,
    4, 12));
10.         System.out.println("j = x * y - z / i =
    "+kl.operasiMM(6, 3, 5, 2));
11.     }
12. }
```

5) Simpan program, Ctrl + shift + S, lalu jalankan program. Jika program sukses, keluaran akan seperti pada gambar berikut :

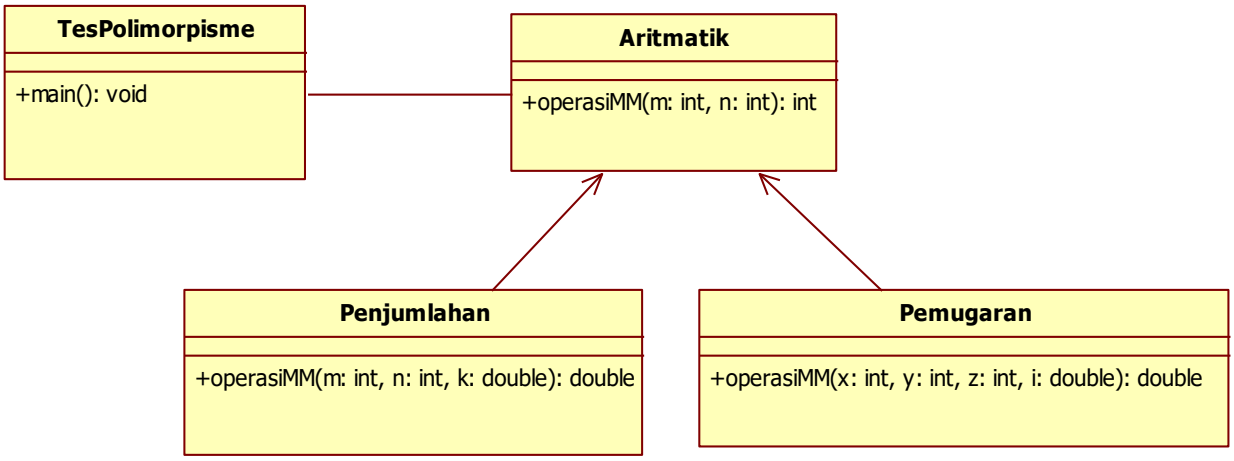


9.1.2 Overriding

Overriding merupakan mekanisme dimana beberapa class dapat menggunakan method yang sama namun dengan cara yang berbeda. Method overriding menerapkan prinsip pewarisan class. Karena prinsip ini pula, class anak tidak hanya dapat menggunakan method yang diwariskan, tetapi attribute yang dimiliki oleh class induk. Polimorpisme pada saat *runtime* dapat menerapkan method overriding ini, dengan inilah objek yang diciptakan dapat mengubah bentuknya.

a. Latihan

Untuk memahami konsep dan praktis polimorpisme saat *runtime* dengan method overriding ini, berikut disajikan contoh dengan menggunakan contoh kasus yang telah disajikan pada polimorpisme pada waktu kompilasi dengan method overloading diatas. Untuk itu perhatikan dahulu diagram class berikut :



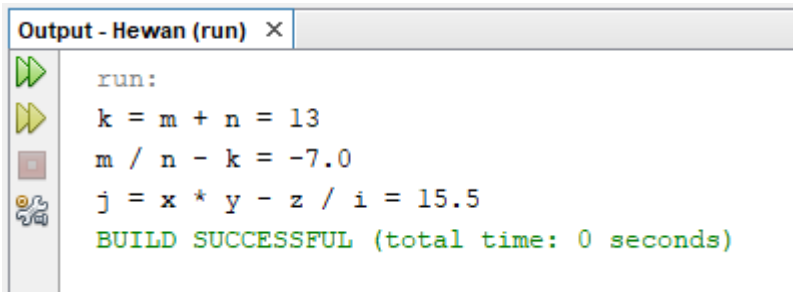
Selanjutnya, berdasarkan diagram class pewarisan diatas, maka rancangan program dapat ditulis seperti pada program berikut. Untuk itu lakukanlah langkah-langkah praktikum berikut ini :

1) Buka NetBean, dan buat Project baru dengan nama TesPolimorpisme
2) Selanjutnya tambahkan file baru, dengan nama Aritmatik
3) Hapus semua kode yang ada, selanjutnya tulis ulang kode program berikut :

```
1.  package tespolimorpisme;
2.  class Aritmatik{
3.      public int operasiMM(int m, int n){
4.          int k = m + n;
5.          return k;
6.      }
```

```
7.  }
8.
9.  class Pemugaran extends Aritmatik{
10.     public double operasiMM(double m, double n, int k){
11.        return (m / n - k);
12.     }
13. }
14.
15. class Pembagian extends Aritmatik{
16.     public double operasiMM(int x, int y, int z, double i){
17.        double j = x * y - z / i;
18.        return j;
19.     }
20. }
21.
22. public class TesPolimorpisme {
23.     public static void main (String [] args){
24.         Aritmatik tmb = new Aritmatik();
25.         Pemugaran krg = new Pemugaran();
26.         Pembagian bg = new Pembagian();
27.
28.          System.out.println();
29.         System.out.println("k = m + n = "+tmb.operasiMM(4, 9));
30.         System.out.println("m / n - k = "+krg.operasiMM(20, 4,
    12));
31.         System.out.println("j = x * y - z / i = "+bg.operasiMM(6,
    3, 5, 2));
32.          System.out.println();
33.     }
34. }
```

4) Simpan perubahan program, Ctrl + Shift + S, lanjutkan dengan menjalankan program, Jika program berjalan sukses, maka yang tampil dikeluaran akan seperti pada gambar berikut



9.2 Tugas Mandiri

Suatu perusahaan distributor alat elektronik memiliki beberapa karyawan yang bertugas sebagai marketing dan kasir, yang sama-sama mendapatkan gaji pokok 500000, kedua karyawan memiliki kesempatan untuk mendapatkan gaji tambahan diantaranya : 1) Marketing yang diberikan 10% dari total penjualan, sementara untuk 2) Kasir mendapatkan

gaji tambahan dari jumlah jam kerja lembur. Berdasarkan kasus diatas, rancanglah program untuk menghitung total gaji karyawan tersebut.

MODUL 10

EXCEPTION HANDLING

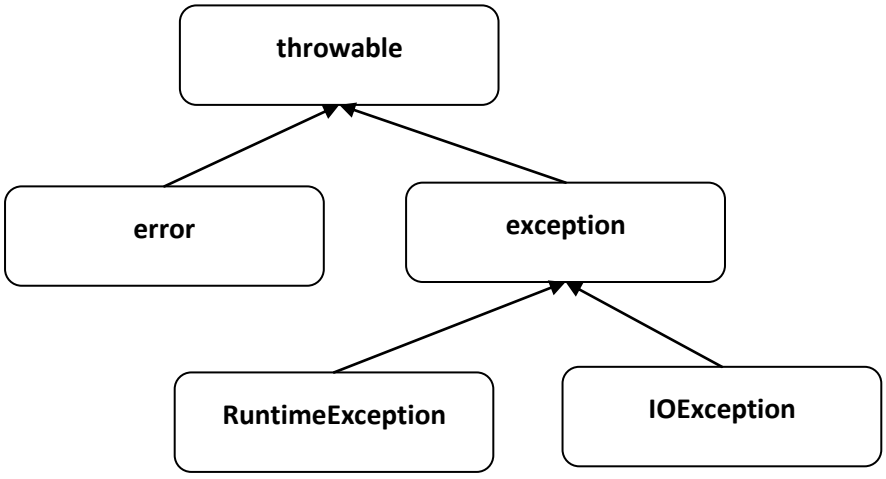
10.1 Error dan Exception

Seringkali program mengalami *error* ketika dijalankan. *Error* disini biasanya menampilkan pesan-pesan yang menjadi penyebab kerusakannya. Kondisi ini dianggap tidak normal karena menampilkan keluaran diluar harapan pengguna. Dalam pemrograman komputer, ketidaknormalan tersebut dinamakan dengan *exception* yang umumnya disebabkan karena beberapa faktor seperti, (1) input pengguna yang tidak valid; (2) kegagalan perangkat; (3) kehilangan koneksi jaringan; (4) keterbatasan fisik (kehabisan memori disk); (5) kesalahan kode; dan, (6) membuka file yang tidak tersedia.

Ketidaknormalan *(error)* program ini tidak dapat dipulihkan, itu sebabnya program menjadi berhenti, dengan kata lain aliran instruksi program menjadi tidak dapat diteruskan dengan ditandai oleh pesan *error* di layar. Agar program tidak dipandang *error*, maka program dapat dirancang sebagai *exception* (pengecualian), sehingga program tetap stabil meskipun instruksi program tidak dapat diteruskan.

10.1.1 Hirarki *Exception* Java

Didalam java, *exception* dikoordinasikan di dalam kelas yang khusus menangani kondisi *error* atau *exception*, yang secara diagram akan ditampilkan pada hirarki diagram kelas berikut :



Dari diagram pewarisan kelas diatas tampak bahwa kelas *throwable* merupakan kelas induk *(root/parentclass)* yang mewariskan sifat-sifatnya kepada kelas anak *(subclass)* seperti kelas *error* dan kelas *exception.*

Perlu diperhatikan bahwa kelas ***error*** merupakan kelas yang menangani kondisi program *error*, yang tidak dapat dipulihkan seperti misalnya kesalahan yang terjadi pada Java Virtual Machine (JVM) kehabisan memori, kesalahan karena tumpukan data yang meluap (seperti pada array), ketidakcocokan pustaka *(library)*, rekursi yang tidak berhenti, dan lain sebagainya. Sementara kelas ***exception*** adalah kelas yang menangani instruksi-instruksi *exception* agar dapat ditangkap dan ditangani oleh program, yakni dengan menciptakan objek-objek *exception* yang telah tersimpan di dalam library. Java menyediakan berbagai objek *exception* yang dikategorikan berdasarkan tipe *exception*-nya.

10.1.2 Jenis *exception* Java

Exception java secara umum terbagi dalam dua jenis, yaitu *RuntimeException* dan *IOException.*

a) RuntimeException

Yakni *exception* yang terjadi karena kesalahan pemrograman, dimana proses kompilasi tidak diprioritaskan. Dengan kata lain, *exception* terjadi pada saat *runtime*, buka pada saat kompilasi. Disebut juga dengan tipe *unchecked exception.* Adapun jenis *RuntimeException* terdiri dari :

1) Penggunaan API yang tidak benar - *IllegalArgumentException*
2) Akses pointer kosong (melewatkan inisialisasi variabel) - *NullPointerException*
3) Akses array di luar batas - *ArrayIndexOutOfBoundsException*
4) Membagi angka dengan 0 - *ArithmeticException*

Perlu diperhatikan, untuk jenis *NullPointerException* tidak akan terjadi jika penggunaan variabel telah sesuai, serta inisialisasi variabel telah dilakukan dengan benar. Begitu pun dengan *exception* jenis *ArrayIndexOutOfBoundsException* tidak akan terjadi jika inisialisasi atau input array tidak melewati batas kapasitas array.

b) IOException

IOException juga dikenal sebagai *checked exception.* Dimana *exception* ini diperiksa oleh kompiler pada waktu kompilasi dan *programmer* diminta untuk menangani pengecualian ini. Beberapa contoh *checked exception* adalah :

1) Ketika mencoba membuka file yang tidak ada hasil - *FileNotFoundException*
2) Ketika mencoba membaca melewati akhir file

10.2 Penanganan *Exception* Java

Diawal kita telah membahas tentang *exception*, yaitu suatu kejadian yang tidak normal (tidak terduga) selama pengujian program. Suatu *exception* dapat diselesaikan dengan penanganan/pengendalian *(exception handling)*. Dalam bahasa java, penanganan *exception* menggunakan kata kunci ***try***, ***catch***, dan ***finally***. Perlu dicatat bahwa kata kunci *finally* bersifat opsional. Dengan kata lain, kata kunci tersebut boleh saja tidak digunakan.

10.2.1 Struktur penanganan *exception*

Secara sederhana penanganan kesalahan *(exception hendling)* menggunakan struktur berikut :

```
try {
    // kode program
  } catch (tipe_exception e) {
    // blok catch
  }
```

Dimana :

- Blok *try{...}* adalah blok yang berisi instruksi-instruksi program yang akan dieksekusi
- Blok *catch{...}* merupakan blok yang menangkap, mengendalikan, dan menangani bilamana *exception* (kesalahan) terjadi pada blok *try*.

Supaya lebih mudah, mari kita perhatikan contoh program berikut, misal *exception* terjadi manakala kesalahan terjadi pada proses aritmatika (perhitungan matematika) pada program berikut ini :

```
class ExceptionBilangan{
 public void bagiBilangan(){
       try{
             int pembagiannol = 10/0;
             System.out.println("10 dibagi 0, adalah
"+pembagiannol);
       }catch(ArithmeticException e){
             System.out.println("Maaf, bilangan tidak dapat
dibagi 0");
       }
 }
}
public class Except{
 public static void main(String[] args) {
       ExceptionBilangan arithmatikCounter = new
```

```
     ExceptionBilangan();
            arithmatikCounter.bagiBilangan();
        }
    }
```

Jika di eksekusi, program akan menghasilkan keluaran sebagai berikut :

```
Maaf, bilangan tidak dapat dibagi 0
```

Dari keluaran program tersebut, dapat dilihat perintah yang dijalankan terdapat pada blok *catch*, ini karena secara matematis memang tidak ada bilangan yang dapat dibagi dengan nilai 0. Kompiler java menilai formula tersebut tidak dapat dijalankan karena mengandung *exception*, sehingga, *catch* menangkap formula tersebut untuk diselesaikan.

a) Blok *Catch* bertingkat

Ada kalanya blok *catch* memiliki tingkatan dimana penggunaan *catch* lebih dari satu. Ini disebut dengan blok *catch* bertingkat. Adapun strukturnya sama dengan struktur *try{...}catch{...}* tunggal, hanya saja pada *catch* bertingkat blok *catch* lebih dari satu blok. Untuk memudahkan pemahaman, kita terapkan saja pada program berikut ini :

```
      import java.util.*;
      public class Except{
            public static void main(String[] args) {
                  int bil, pembagi, hasil=0;
                  Scanner io = new Scanner(System.in);
                  try{
                        System.out.print("Input deviden = ");
                        bil=io.nextInt();

                        System.out.print("Input pembagi = ");
                        pembagi=io.nextInt();

                        hasil=bil/pembagi;
                        System.out.println("hasil baginya =
      "+hasil);
                  }catch(ArithmeticException e){
                        System.out.println("Exception "+
      e.toString());
                  }catch(InputMismatchException e) {
                        System.out.println("Exception "+
      e.toString());
                  }
            }
}
```

Sekarang kita akan implementasikan dalam beberapa percobaan berikut ini :

1) Percobaan petama

```
Input deviden = 16
Input pembagi = 5
hasil baginya = 3
```

2) Percobaan kedua

```
Input deviden = 15
Input pembagi = 0
Exception java.lang.ArithmeticException: / by zero
```

3) Percobaan ketiga

```
Input deviden = 15
Input pembagi = a
Exception java.util.InputMismatchException
```

Pada percobaan pertama, program berjalan normal karena input sesuai dengan tipe data dan bilangan masih relevan prinsip matematis yang diterima oleh kompilator. Pada percobaan kedua, meskipun input bilangan masih sesuai tipe, tetapi ini diluar prinsip matematis, yang dianggap sebagai *exception* oleh kompiler, sehingga diambil alih oleh blok *catch* pertama. Dan pada percobaan ketiga, input kedua tidak relevan dengan tipe, sehingga diambil alih oleh blok *catch* ketiga karena juga dianggap sebagai *exception.*

b) Menggunakan *keyword finally*

Kata kunci *finally* bersifat opsional, dengan kata lain, kata kuncil *finally* boleh tidak disertakan. Blok *finally* akan selalu dijalankan meskipun *exception* tidak terjadi. Hanya saja tahap blok *finally* mengacu kepada blok *try* dan ataupun blok *catch.* Tahapan pada *finally* adalah :

1) Jika *exception* terjadi, maka blok *finally* dijalankan setelah blok *try{ ..}catch{...}*
2) Jika *exception* tidak terjadi, maka blok *finally* dijalankan setelah blok *try{...}*

Sebagai contoh, berikut ini disajikan contoh program untuk melihat bagaimana fungsi blok *finally* terhadap *exception* yang terjadi, seperti pada kode berikut :

```
import java.io.*;
class DaftarArray {
      private int[] Arr = new int[10];
      public DaftarArray() {
        // Menyimpan data integer ke array Arr
        for (int i = 0; i < 10; i++) {
          Arr[i] = i;
        }
```

```
        }

        public void tulisArr() {
          PrintWriter out = null;
          try {
            System.out.println("Mencoba try ");
            // Membuat file baru : OutputFile.txt
            out = new PrintWriter(new
    FileWriter("OutputFile.txt"));
            // Mencetak nilai dari daftar Array Arr ke file
    baru
            for (int i = 0; i < 10; i++) {
              out.println("Elemen ke : " + i + " = " + Arr[i]);
            }
          } catch (IndexOutOfBoundsException e1) {
            System.out.println("IndexOutOfBoundsException => "
    + e1.getMessage());
          } catch (IOException e2) {
            System.out.println("IOException => " +
    e2.getMessage());
          } finally {
            // Mengecek apakah PriterWriter telah dibuka
            if (out != null) {
              System.out.println("Mengakhiri cetakan Array");
              out.close();
            } else {
              System.out.println("PrintWriter tidak terbuka");
            }
          }
        }
    public class ExceptCatchFinally {
        public static void main(String[] args) {
          DaftarArray array = new DaftarArray();
          array.tulisArr();
        }
    }
```

Jika program diatas dijalankan, program akan menampilkan keluaran seperti berikut ini :

```
Mencoba try
Mengakhiri cetakan Array
```

Mekanisme program diatas adalah :

1) Methode main di di kelas ExceptCatchFinally memanggil method `tulisArr()`, dimana method ini juga memanggil method FileWrite() untuk membuat file "OutputFile.txt"

2) *Exception* terjadi pada methode FileWrite(), tetapi karena method tersebut tidak memiliki *exception handler*, maka pada saat *runtime* exception blok *try* diabaikan dan diambil alih oleh blok catch *IOException*
3) *Exception handler IndexOutOfBoundsException* tidak sesuai dengan *IOException*, sehingga blok catch pertama dilewatkan dan dilempar ke blok *catch* yang cocok dibawahnya, yaitu blok *catch IOException.*
4) Blok *finally* merupakan blok yang terus akan dijalankan meskipun tidak terjadi *exception*, sehingga eksekusi berakhir dengan menampilkan pesan di dalam blok *finally*.

10.2.2 Menggunakan *keyword throw* dan *throws*

Pada umumnya penanganan jenis *RuntimeException* tidak perlu dilakukan karena terjadi karena kesalahan pemrograman. Namun pada *IOException* penanganan perlu dilakukan. Penanganan untuk pada *IOException* juga dapat dilakukan dengan menggunakan kata kunci *throw* dan *throws.*

a) Throw

Adapun penanganan dengan *throw* mengacu kepada sintaks berikut :

```
throw new kelas_exception("pesan error");
```

sekarang, perhatikan contoh program berikut :

```
public class Except{
      static void cekUsia(int usia) {
            if(usia>=35) {
                  throw new ArithmeticException("Maaf, usia Anda
    sudah tidak bisa mengikuti seleksi CPNS");
            }else {
                  System.out.println("Anda masih bisa mengikuti
    seleksi CPNS");
            }
      }
      public static void main(String[]args) {
            cekUsia(36);
      }
    }
```

Jika program tersebut dijalankan akan menampilkan keluaran sesuai dengan kondisi sebagai berikut :

1) Untuk usia >= 35, keluaran sebagai berikut

```
Exception in thread "main" java.lang.ArithmeticException: Maaf,
usia Anda sudah tidak bisa mengikuti seleksi CPNS
	at Except.cekUsia(Except.java:4)
	at Except.main(Except.java:10)
```

Tampak *exception* menampilkan keluaran yang sesuai, meskipun tetap menampilkan pesan kesalahan

2) Untuk usia <35, keluaran sebagai berikut :

```
Anda masih bisa mengikuti seleksi CPNS
```

b) Throws

Kata kunci *throws* umumnya ditulis bersamaan dengan deklarasi method. Adapun penggunaan kata kunci *throws* mengacu kepada sintaks berikut :

```
nilai_baliktipe nama_method() throws kelas_exception{
// kode program
}
```

Untuk melihat penggunaan *throws* berikut ini kita akan lihat dalam contoh program berikut ini :

```
public class ExceptThrows{
    int bagiBilangan(int x, int y) throws ArithmeticException{
            int hasil = x/y;
            return hasil;
    }

    public static void main(String[]args) {
            ExceptThrows bilangan = new ExceptThrows();
            try {
                    System.out.println(bilangan.bagiBilangan(15, 0));
            }catch(ArithmeticException e) {
                    System.out.println("Bilangan tidak bisa dibagi dengan
0");
            }
    }
            }
```

Jika program dijalankan, hasilnya seperti pada tampilan berikut ini :

```
Bilangan tidak bisa dibagi dengan 0
```

MODUL 11

PEMROGRAMAN GRAFIK (GUI)

11.1 Grafical User Interface (GUI)

Secara teknis perancangan aplikasi GUI dengan bahasa Java dengan NetBeans dapat diterapkan dalam dua metode umum. Metode pertama adalah dengan menggunakan metode *script* atau yang umum disebut dengan *hardcoding*. Metode tersebut membutuhkan pemahaman, baik konsep maupun praktis yang cukup baik. Sehingga tentunya, *programmer* memerlukan waktu yang lama untuk mempelajarinya secara fundamental. Begitupun, ukuran lama waktu yang dibutuhkan seorang programmer dapat merancang aplikasi gui dengan metode ini tidak menutup kemungkinan dapat ditempuh dengan cepat. Semua bergantung kepada jam terbang, dan banyaknya latihan dan fokus. Dalam modul ini, GUI hanya akan disampaikan melalui metode *drag* and *drop* (tarik-lepas) yang telah tersedia pada editor NetBeans.

Dalam perkuliahan teori, penggunaan aplikasi GUI dengan bahasa Java telah disampaikan yakni dapat menggunakan beberapa jenis *package* yaitu : (1) *AWT (Abstract Window Toolkit)*, yang merupakan GUI Toolkit pertama yang dibangun oleh *Microsoft*. Toolkit *AWT* memiliki kekurangan terutama, variasi tools yang dibutuhkan untuk dapat digunakan dalam perancangan aplikasi GUI; (2) Selain *AWT* IBM melalui Sun Microsystem, juga mengembangkan *package* yang diberi nama dengan *SWT (Standart Widget Toolkit)*; dan, (3) *Java Swing*. Fasilitas *Java Swing* cukup lengkap untuk membangun aplikasi GUI berbahasa Java karena sifatnya yang *cross platform*, yaitu dengan Java Swing aplikasi yang dirancang dapat berjalan di berbagai *platform* sistem operasi. Berbeda dengan *package AWT* yang dibangun untuk mendukung aplikasi GUI dari Java yang berjalan di sistem operasi Windows, sementara *SWT* yang dirancang untuk mendukung aplikasi java di sistem operasi Unix.

11.1.1 JAVA Swing

Kelebihan *Java Swing* yang menawarkan berbagai tools didalamnya (palette), memudahkan para *programmer*¸ terutama programmer pemula. Pada umumnya di dalam *package* Java Swing terdapat beberapa tools, seperti yang diuraikan pada tabel dibawah :

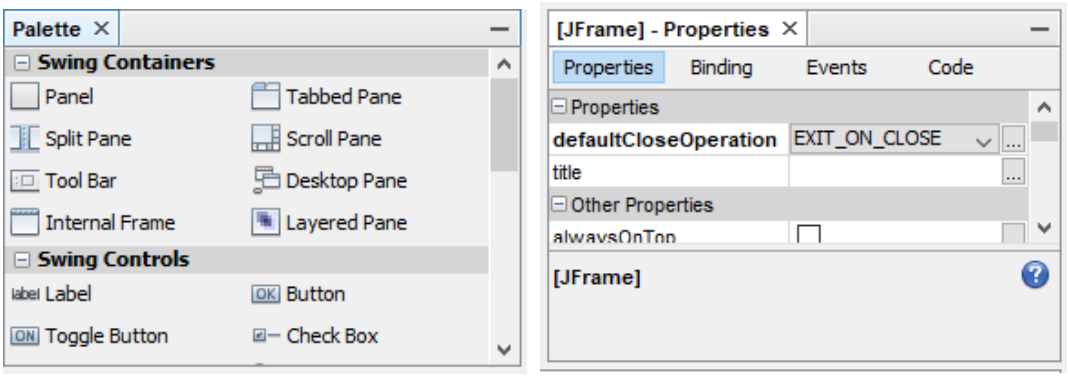
Tabel 10.1 Tabel Tools NetBeans

NO	NAMA TOOLS	FUNGSI UMUM
1	jLabel	Digunakan untuk meletakan *caption* (*labeling*) pada sebuah tools lain yang memiliki
2	jTtextField	Digunakan untuk kotak input dan keluaran
3	jPanel	Umumnya digunakan untuk mengkoleksi tools berdasarkan kategori.
4	jComboBox	Digunakan untuk pemilihan secara *full down option*
5	jRadioButton	Digunakan untuk pemilihan secara klik
6	jButton	Digunakan untuk memanggil *script* untuk melakukan proses tersentu

Tools yang disampaikan di tabel 7.1 belum lengkap, ada banyak *tools* lain yang tersimpan didalam *palette*, nantinya akan diuraikan dalam contoh-contoh program.

11.1.2 Palette dan Properties

Jika familiar dengan Microsoft Visual Studio, tentu familiar dengan Toolbox. Nah, *palette* pada NetBeans merupakan Toolbox yang menyediakan berbagai tools yang telah disinggung pada subtitel Java Swing. Secara visual, palette seperti pada gambar berikut :



(a) (b)

Gambar 10.1 (a) Palette, dan (b) Properties

Sementara *Properties* merupakan komponen yang digunakan untuk merubah karakteristis *tools* yang diambil dari *Toolbox*, seperti nama variabelnya, nama properti dari *tool* tersebut dan lain sebagainya.

11.1.3 Frame (Form)

Frame merupakan jendela (Form) perancangan aplikasi GUI yang disediakan pada Java Swing. Tools ini secara pemberkasan berada pada class Java Jframe Form. Adapun Frame (Form) ditampilkan pada gambar berikut :

Gambar 10.2 jFrame (Form)

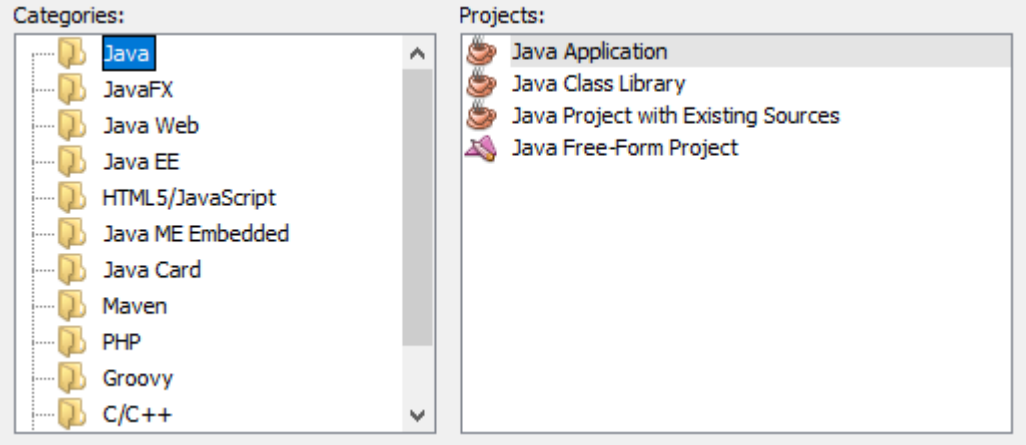
11.2 Menggunakan GUI

Dalam penggunaanya dengan NetBeans, Java GUI tidak serta merta mamberikan antarmuka Frame Rancangan GUInya seperti pada Microsoft Visual Studio. Jika aplikasi memiliki beberapa Frame yang terkait dalam satu aplikasi (aplikasi yang sama), dibutuhkan file utama yang berfungsi untuk menjalankan seluruh Form Melalui Jendela Utama (Jendela Menu). Untuk menggunakan GUI Java di NetBeans ada beberapa langkah yang akan diuraikan sebagai berikut :

11.2.1 Membuat Projek Baru

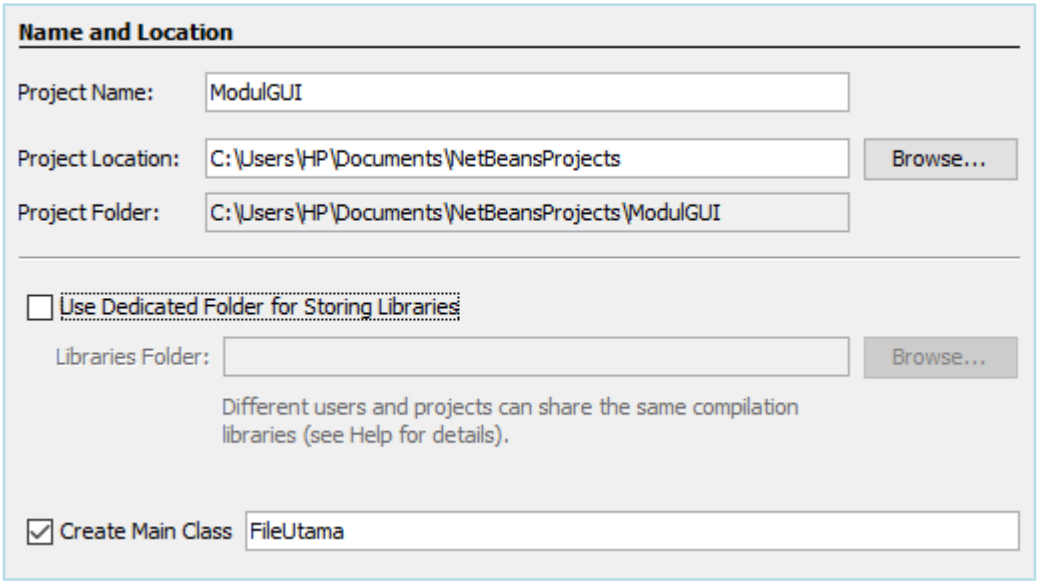
Untuk membuat projek baru, sama seperti ketika menjalankan aplikasi java *console*. Secara lengkap dapat mengikuti langkah-langkah berikut :

1) Klik Menu File - New Project, atau klik Icon New Project dibawah Menu File ( )
2) Langkah selanjutnya pada kolom Categories pilih **Java** dan sementara pada kolom **Project** pilih **Java Application.**



Gambar 10.3 Penentuan Nama Project

3) Klik **Next,** kemudian ubah kotak **Project Name** untuk memberi nama projectnya.



Gambar 10.4 Penentuan Nama Project

4) Untuk mengakhiri klik **Finish.** Jika berhasil akan menampilkan halaman pengkodean seperti berikut

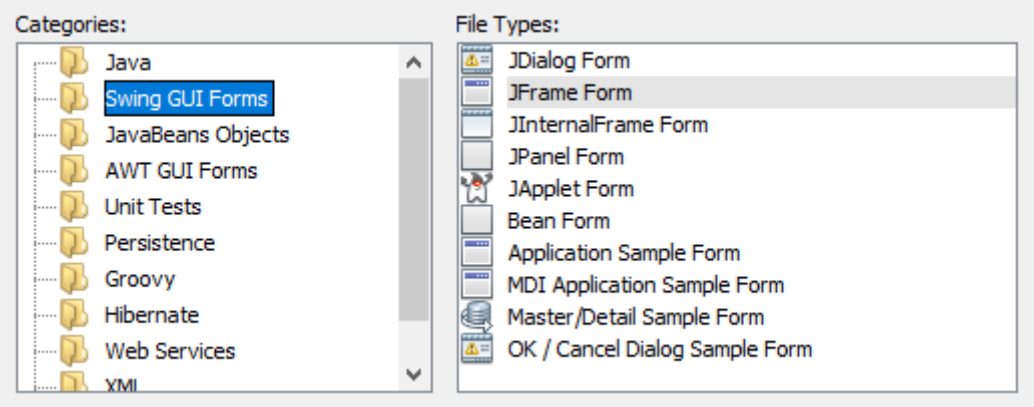
```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6
7  /**
8   *
9   * @author HP
10  */
11 public class FileUtama {
12     /**
13      * @param args the command line arguments
14      */
15     public static void main(String[] args) {
16         // TODO code application logic here
17     }
18 }
```

Gambar 10.5 Halaman Pengkodean Program Utama

11.2.2 Membuat Form Baru (jFrame)

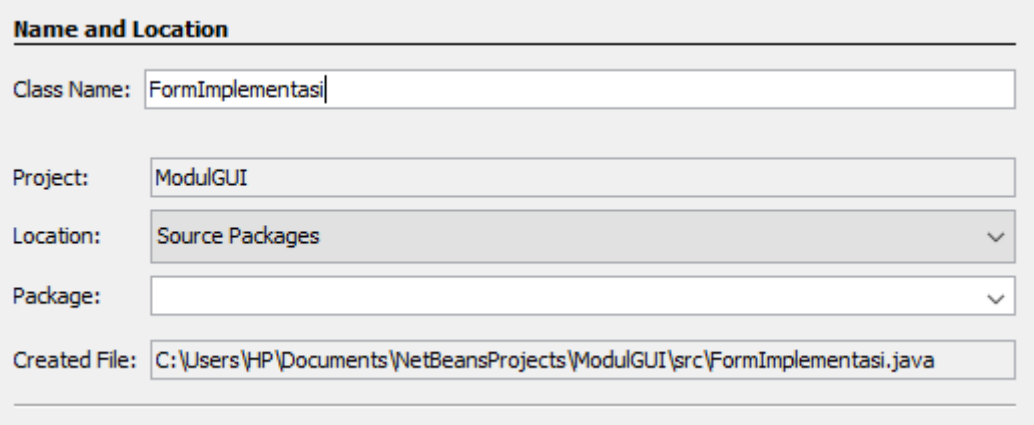
Ini adalah tahap pembuatan Form Baru, dimana Form merupakan antarmuka yang berfungsi untuk mengimplementasikan *tools* sebagai komponen pendukung dalam perancangan aplikasi berbasis GUI. Untuk membuat Form baru, ikuti langkah-langkah berikut ini :

1) Pastikan Kursor berada pada Project Utama yang telah dibuat. Dalam modul ini, Project diberinama ModulGUI.
2) Klik menu File - New File, atau klik icon New File yang berada dibawah menu File ( )



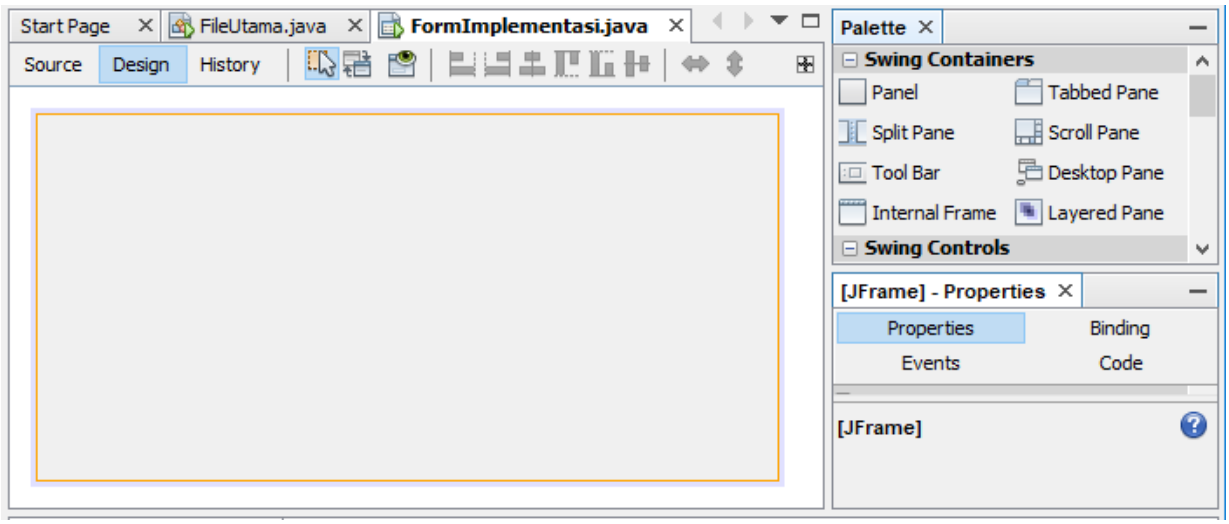
Gambar 10.6 Pemilihan Mode Form GUI

3) Kemudian pada kolom Categories pilih **Swing GUI Forms,** sementara pada kolom **File Types** pilih **Jframe Form,** klik **Next** untuk melanjutkan



Gambar 10.7 Menentukan Nama (*class*) Form

4) Langkah terakhir, ubah dahulu nama classnya pada kolom ***Class Name*** seperti pada gambar diatas, dan klik **Finish** untuk mengakhiri langkah pembuatan Form. Jika berhasil akan menampilkan antarmuka seperti pada gambar berikut :



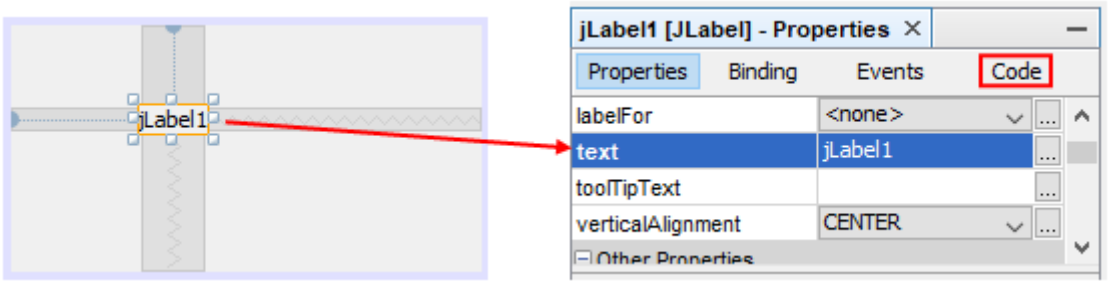
Gambar 10.8 Antarmuka Form Perancangan

11.3 Menggunakan *Tools*

Beberapa komponen tools telah diuraikan pada subtitel *JAVA Swing* diatas, selanjutnya pada bagian ini akan diuraikan penggunaan tools, properties dan penerapannya kedalam pengkodean program. Perlu diperhatikan, penggunaan GUI dan *tools* disini menerapkan prinsip *drag* dan *drop*. Adapun prinsip *drag* dan *drop* adalah klik tools, tahan, geser tools yang dipilih lalu lepaskan dijendela form perancangan. Untuk mempermudah penguasaan praktisnya, alangkah baiknya mengikut panduan praktis yang secara runut diuraikan sebagai berikut :

11.3.1 jLabel

jLabel merupakan tools yang umum digunakan untuk membuat pelabelan (penamaan). jLabel tidak dapat menerima input, tetapi dapat diterapkan sebagai jendala output (keluaran). Secara visual, jLabel mewakili suatu ***field*** atau judul kolom didalam tabel. Untuk menggunakannya, klik tools jLabel lalu geser dan lepas ke jendela form perancangan.

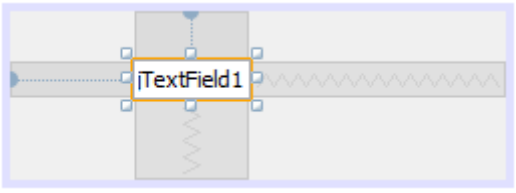


Gambar 10.9 Pengaturan jLabel

Untuk properti jLabel, umumnya cukup merubah text saja, namun jika ingin merubah latar belajar, ukuran font dan lain sebagainya dapat menggantinya pada jendela properties, yang dapat dieksplorasi pada tab ***Binding***, ***Events***, dan ***Code***. Sementara untuk nama variabel dapat diatur melalui tab ***Code***.

11.3.2 jTextField

jTextField merupakan tools yang berfungsi untuk kotak masukkan. Dalam penerapannya, tools ini membutuhkan perubahan properti pada komponen text, variabel name dan latarbelakangnya jika dibutuhkan.



Gambar 10.10 jTextField

1) Input Bilangan jTextField

Input yang dilakukan terhadap jTextField memiliki beberapa perbedaan terutama karena tipedatanya. Karena, pada prinsipnya input yang dilakukan melalui tools ini dianggap sebagai strings, sehingga perlu mengkonversinya ke dalam data tertentu sesuai tipedatanya. Adapun konversi input mengikuti struktur berikut :

```
Tipedata namavariabel = Interger.ParseInt(tools.getText());
```

a) Input Tipedata Integer (bilangan bulat)

Contoh :

```
int x = Integer.ParseInt(jTextField1.getText());
```

b) Input tipe data floating point (bilangan pecahan)

Contoh :

```
float x = Float.ParseFloat(jTextField1.getText());
Double x = Double.ParseDouble(jTextField1.getText());
```

c) Input tipe data String

```
Tipedata namavariabel = String.valueOf(tools.getText());
```

Contoh :

```
String teks = String.valueOf(lblKeterangan.getText());
```

2) Menampung Output

Keluaran yang diterapkan dari tools tertentu, dapat pula ditampilkan atau diterima oleh tools yang lainnya. Perancang dapat memilih tools yang paling tepat sesuai kebutuhan. Sifat tools hanya mampu menerima data bertipe String, sehingga keluaran harus dikonversi dahulu ke String agar dapat ditampilkan pada tools lain. Untuk mengkonversi data yang ingin ditampilkan (dikirim) ke tools lain, dapat mengikuti struktur berikut :

```
Tools.setText(String.valueOf(variabel));
```

a) Mengirim Output ke jTextField

```
Int c = a * b;
jTextField1.setText(String.valueOf(c));
```

b) Mengirim Output ke jLabel

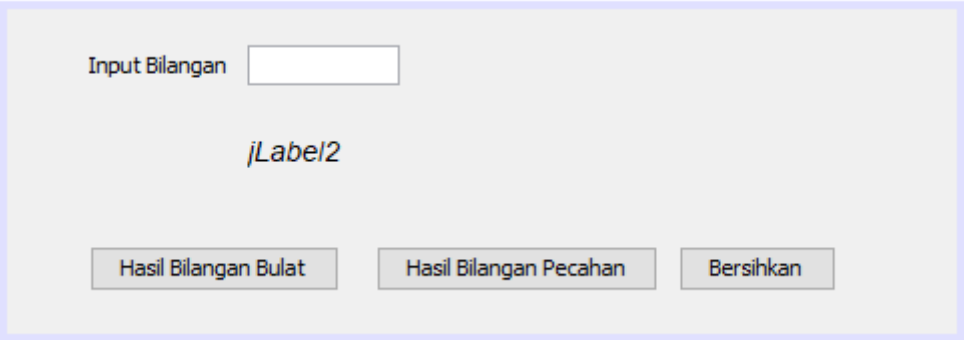
```
Double c = a * b;
jLabel.setText(String.valueOf(c));
```

Karena jTextField merupakan kotak masukan, maka input yang diterima dapat berupa input bilangan bulat, bilangan pecahan, maupun data karakter dan string. Misal sebuah form memiliki beberapa komponen dengan keterangan properti seperti pada tabel berikut :

Tabel 10.3 Tabel Penggunaan Tools

Tools	Text Properties / title	Nama Variabel	Font
jFrame	Kalkulator Bilangan	Frame1	(default)
jLabel1	Input Bilangan	jLabel1	(default)
jLabel2	jLabel2	lblHasil	Consolas, 24
jTextField1	(kosongkan)	txtBilangan	
jButton1	Hasil Bilangan Bulat	btnBulat	
jButton2	Hasil Bilangan Pecahan	btnPecahan	
jButton3	Bersih	btnBersih	

Dari daftar kebutuhan tools diatas, dapat dirancangkan antarmuka seperti pada gambar berikut :



Gambar 10.11 Contoh Program Hitung Bilangan

Rancangan antarmuka program diatas dapat kita tuliskan kode programnya. Ikut langkah-langkahnya :

1) Klik Ganda tombol Hasil Bilangan Bulat, lalu ketik program berikut

```
1.   private void btnBulatActionPerformed(java.awt.event.ActionEvent
     evt) {
2.           int n = Integer.parseInt(txtBilangan.getText());
3.           n %= 2 ;
4.           if (n==0) {
5.               lblHasil.setText(String.valueOf(n+ " adalah bilangan
     : "+"Genap"));
```

```
6.          } else {
7.              lblHasil.setText(String.valueOf(n+ " adalah bilangan
    : "+"Ganjil"));
8.          }
9.      }
```

2) Kemudian klik kembali tombol Hasil Bilangan Pecahan, lalu ketik kode program berikut

```
10. private void
    btnPecahanActionPerformed(java.awt.event.ActionEvent evt) {
11.         Double n = Double.parseDouble(txtBilangan.getText());
12.         n *= 3;
13.         lblHasil.setText(String.valueOf(n+ " adalah bilangan
    pecahan"));
14.     }
```

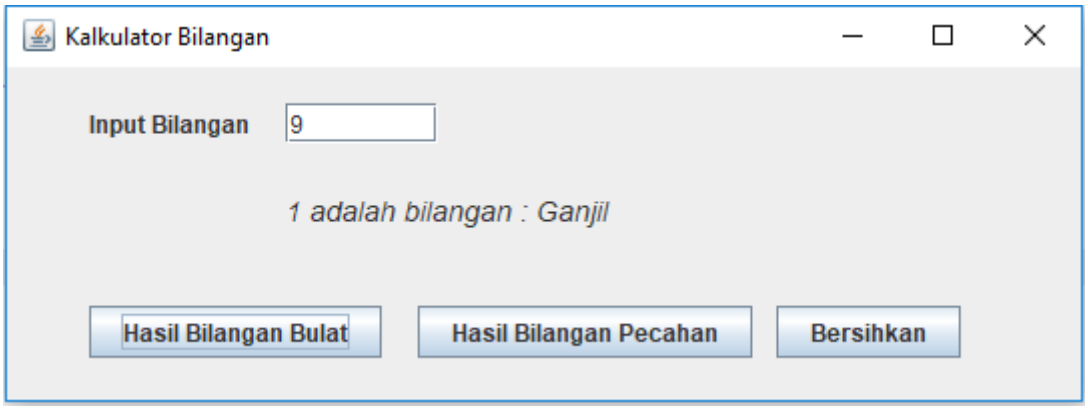
3) Kemudian klik tombol bersih untuk membersihkan kotak masukkan

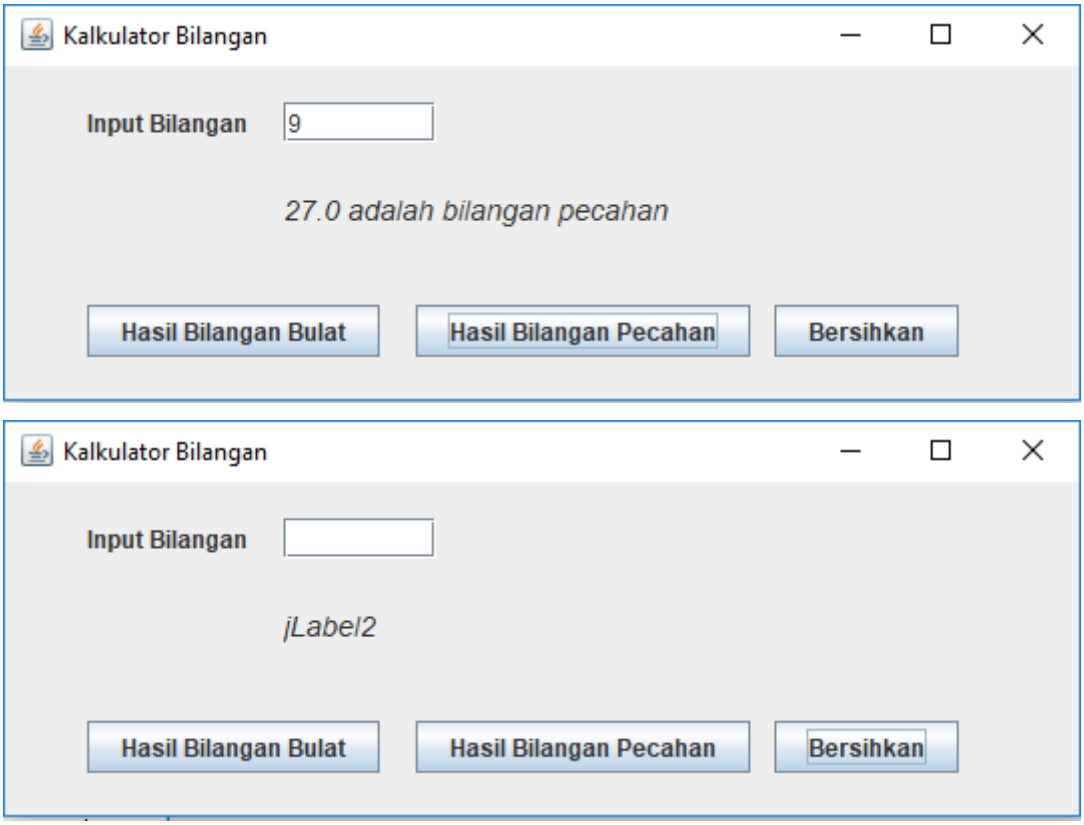
```
15. private void btnBersihActionPerformed(java.awt.event.ActionEvent
    evt) {
16.         txtBilangan.setText("");
17.         lblHasil.setText("jLabel2");
1.      }
```

4) Langkah terakhir, buka jendela pengkodean program di file Utama, lalu ketik kode program berikut

```
1.  public class FileUtama {
2.
3.      public static void main(String[] args) {
        // Memanggil class jFrame dari jendela pengkodean Program Utama
4.          TesTools tes=new TesTools();
5.          tes.setVisible(true);
6.      }
7.  }
```

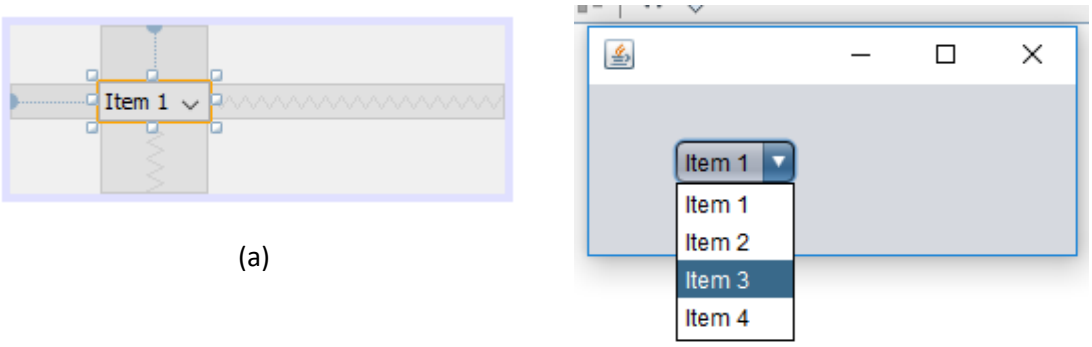
5) Jalankan program, hasil program akan seperti pada gambar berikut





11.3.3 jComboBox

Merupakan tools yang berfungsi untuk mendaftarkan pilihan secara menurun *(full down).* Tools ini biasanya membutuhkan pengaturan properti pada komponen nama variabel dan daftar item. Ada beberapa cara yang dapat dilakukan untuk mendaftarkan pilihan ke dalam jComboBox, yakni melalui pengaturan properties dan melalui pengkodean dengan memanfaatkan method .addItem(). Berikut akan disajikan dua cara saja yang termudah.



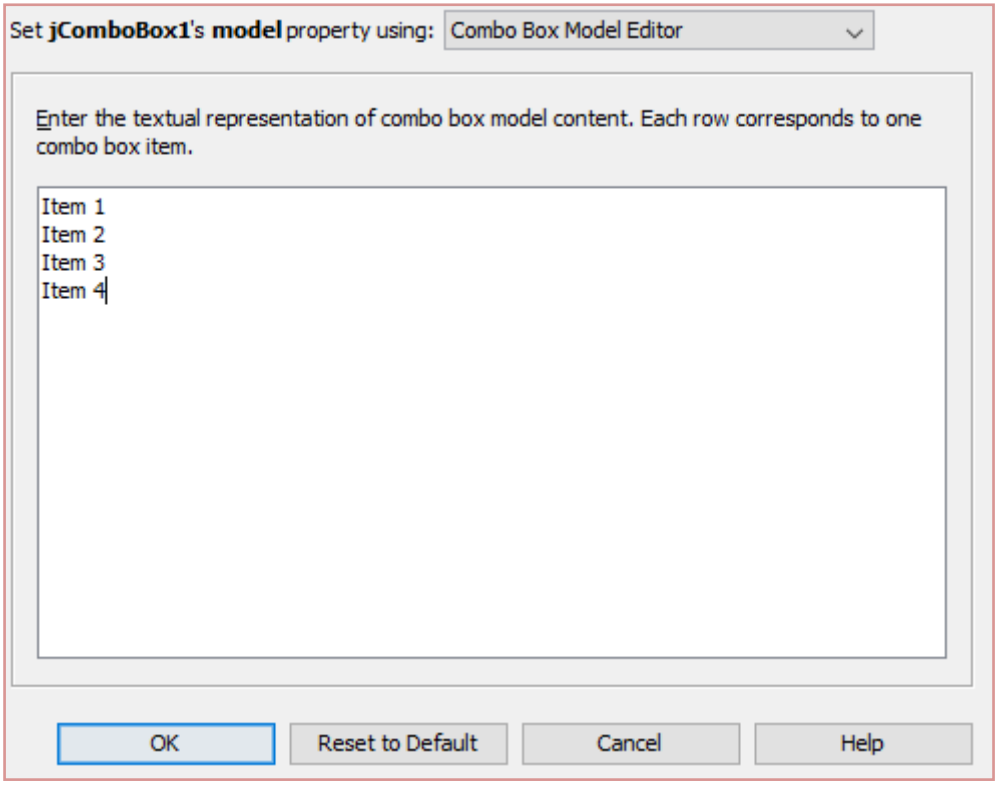
(a)

(b)

Gambar 10.13 (a) ComboBox sebelum dijalankan, (b) Setelah dijalankan

1) Mendaftarkan pilihan melalui properties

Untuk mendaftarkan pilihan jComboBox melalui properties, klik kanan **jComboBox** - pilih **properties** - pilih **model** - kemudian ganti pilihan Item 1, Item 2, Item 3, dan Item 4 yang ada didalam kotak yang tersedia dengan daftar pilihan yang diinginkan, seperti gambar berikut dibawah, selanjutnya tekan **OK, Close** :



Gambar 10.14 Menambahkan Daftar Pilihan jComboBox

2) Mendaftarkan Pilihan Melalui Pengkodean

Pendaftaran item pilihan jComboBox dapat dilakukan melalui kode program. Dengan model ini ada banyak method yang dapat dilakukan, namun pada modul ini akan disampaikan cara yang paling sederhana yaitu dengan method `.addItem();` Untuk mendaftarkan pilihan dengan methode .AddItem(), klik kanan jendela **Form** - pilih **Events** - **Window** kemudian klik **windowActivated,** langkah selanjutnya isikan daftar pilihan dengan struktur perintah :

```
namavariabelJComboBox.addItem(“daftar”);
```

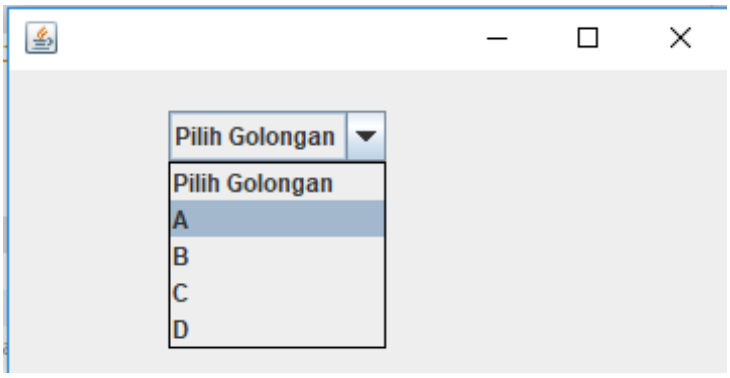
contoh :

```
cmGolongan.addItem(“A”);
```

Pengkodean jComboBox untuk mendaftarkan pilihan didalamnya, secara lengkap ditunjukkan pada contoh dibawah ini :

```
private void formWindowActivated(java.awt.event.WindowEvent evt) {
    cbGolongan.addItem("Pilih Golongan");
    cbGolongan.addItem("A");
    cbGolongan.addItem("B");
    cbGolongan.addItem("C");
    cbGolongan.addItem("D");
}
```

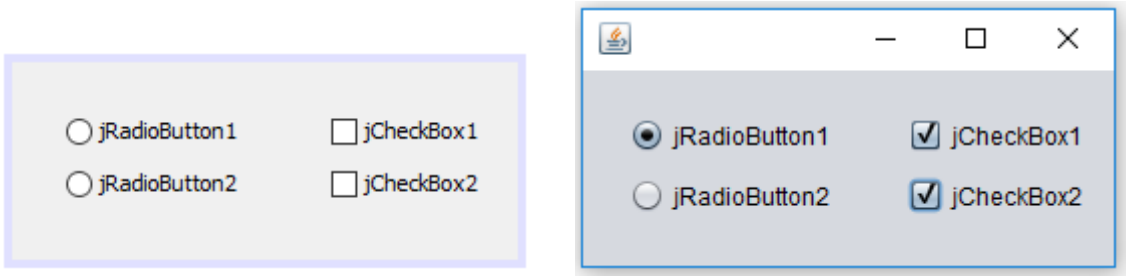
Gambar 10.15 Kode Program addItem() JComboBox



Gambar 10.16 Daftar Pilihan JcomboBox dengan method .addItem();

11.3.4 jRadioButton dan jCheckBox

jRadioButton umumnya digunakan untuk mendaftarkan pilihan yang hanya dapat menerima opsi tunggal, meskipun ini dapat digunakan juga untuk multipilihan, tetapi disarankan untuk opsi tunggal. Berbeda dengan jCheckBox, yang dapat dipilih secara multi opsi. Kedua tools ini umumnya membutuhkan perubahan properti pada komponen nama variabel dan textnya.

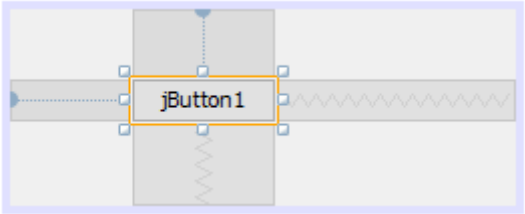


(a) (b)

Gambar 10.17 (a) Gambar sebelum dijalankan; dan, (b) Setelah dijalankan

11.3.5 jButton

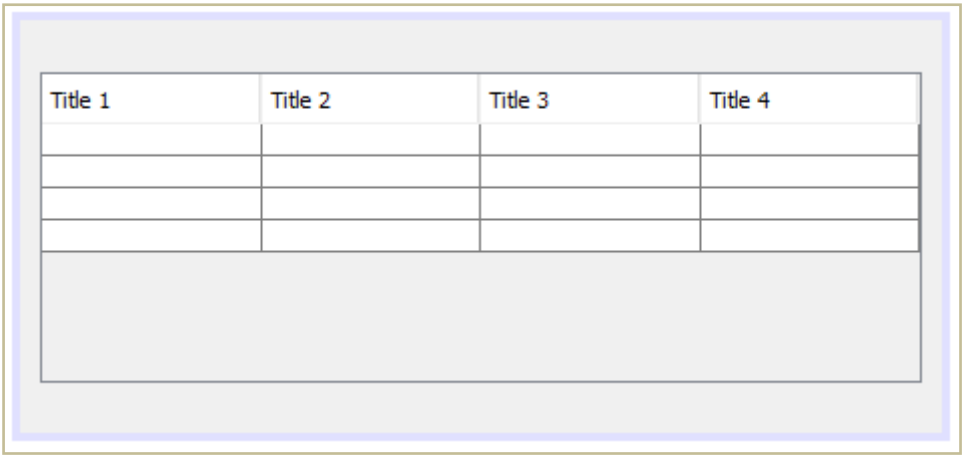
jButton merupakan tools yang berfungsi untuk memanggil perintah yang telah ditulis oleh programmer didalam tools ini. Pengaturan properti untuk tools ini cukup pada komponen text, dan nama variabelnya saja.



Gambar 10.17 Penggunaan jButton

11.3.6 jTable

Merupakan tools yang berfungsi untuk manampilkan data dari database aplikasi. Untuk pengaturan properti jTable cukup pada komponen nama variabel dan disain judul kolom (*field*)-nya saja. Namun ini bergantung kepada kebutuhan dan selera saja. Untuk mengatur kolom jTable, dapat dilakukan beberapa cara, dalam modul ini akan disajikan dalam dua cara, yakni melalui jendela properties dan pengkodean.

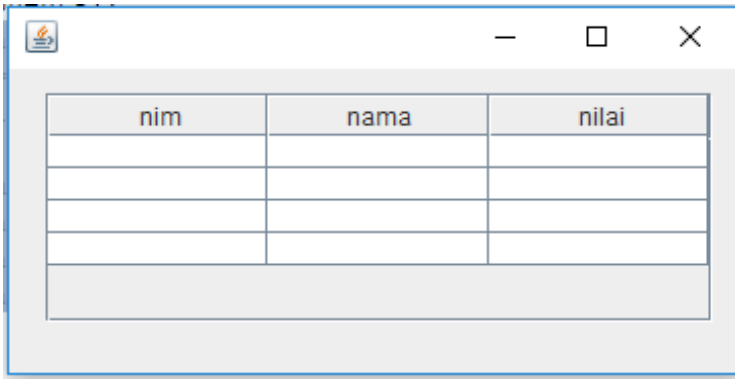


Gambar 10.18 Penggunaan jTable

1) Mengubah judul kolom jTable melalui properties

Untuk mengubah nama judul kolom (*field)* melalui properties dapat dilakukan dengan cara klik kanan **jTable** - pilih **properties** - pilih **model** - selanjutnya ganti Title 1, Title 2, Title 3, dan Title 4 dengan cara klik ganda pada Title yang akan diubah, misal diganti dengan **nim, nama, nilai**. Posisi judul kolom juga dapat dipindahkan posisinya dengan Move Up dan

Mode Down, serta dapat juga menghapusnya dengan menekan tombol delete. Selanjutnya klik **Ok,** dan **Close** untuk mengakhiri.



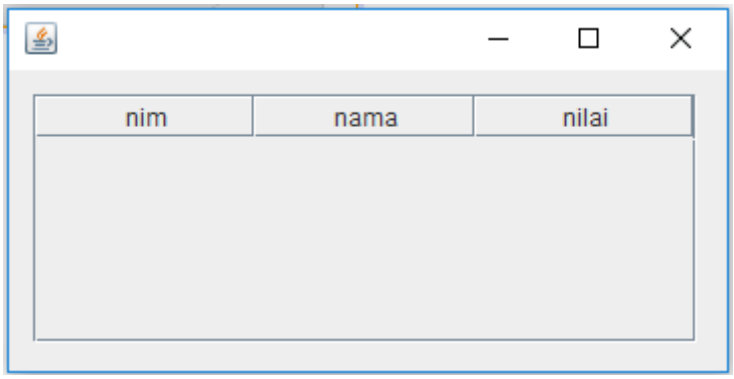
Gambar 10.19 Pengaturan Kolom jTable melalui properties

2) Mengubah Judul Kolom dengan pengkodean program

Cara ini mungkin lebih repot, tetapi jika terus berlatih justeru cara ini menjadi lebih mudah. Ada banyak method yang dapat diterapkan, namun pada kesempatan ini disajikan cara yang paling mudah. Yaitu dengan menggunakan package ***import javax.swing.table.DefaultTableModel;*** dan method .addColumn(); method addColumn() dapat diletakkan didalam konstruktor class. Misal class bernama Table, maka kode dapat ditulis didalam konstruktor **public Table(){ }** seperti pada contoh berikut :

```
1  import javax.swing.table.DefaultTableModel;
2  public class Table extends javax.swing.JFrame {
3      private static final long serialVersionUID = 1L;
4      private final DefaultTableModel model;
5
6      public Table() {
7          initComponents();
8          setLocationRelativeTo(null);
9          model = new DefaultTableModel();
10         jTable1.setModel(model);
11
12         //Membuat nama kolom
13         model.addColumn("nim");
14         model.addColumn("nama");
15         model.addColumn("nilai");
16     }
```

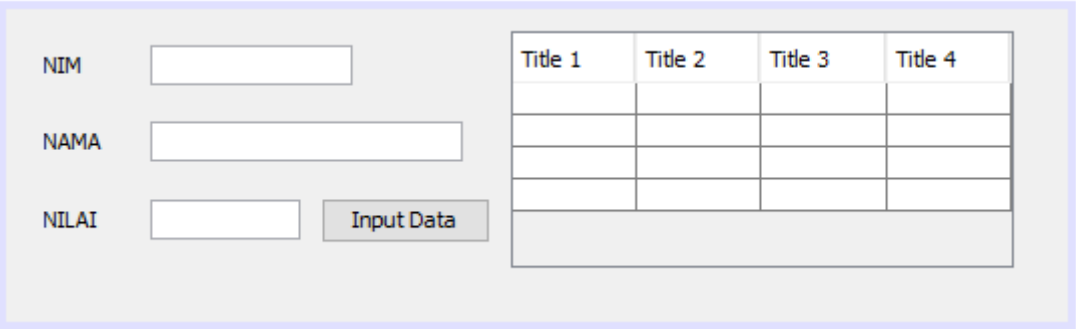
Gambar 10.20 Kode program pengaturan kolom jTable



Gambar 10.21 Tabel dengan pengaturan kode program

3) Menampilkan Data ke jTable

Kadangkala data yang diinput perlu ditampilkan di tabel agar mudah dimengerti. Untuk menampilkan data tersebut ke tabel, bergantung kepada dimana letak data. Data yang terletak di tabel-database tentu berbeda dengan menampilkan data yang berasal dari input jTextField. Pada kali ini akan diuraikan cara menampilkan data yang berasal dari jTextField. Misal rancangan data mengacu kepada tabel diatas, maka rancangan antarmuka form input dapat dibentuk seperti berikut :



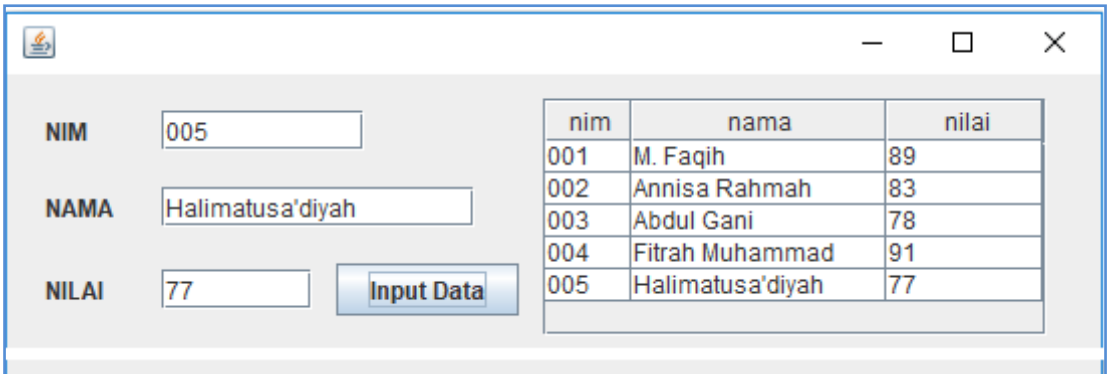
Gambar 10.22 Form Input

Selanjutnya untuk menampilkan data yang diinput melalui jTextField (NIM, NAMA, dan NILAI) dapat dilakukan melalui tombol **Input Data**, perhatikan kode program berikut :

```
private void btnInputActionPerformed(java.awt.event.ActionEvent evt) {
    jTable1.getModel();
    String data[] ={txtnim.getText(),txtnama.getText(),txtnilai.getText()};
    model.addRow(data);
}
```

Gambar 10.23 Kode program Input Data

Jika dijalankan, aplikasi tersebut akan seperti pada gambar berikut :



Gambar 10.24 Keluaran Program jTable

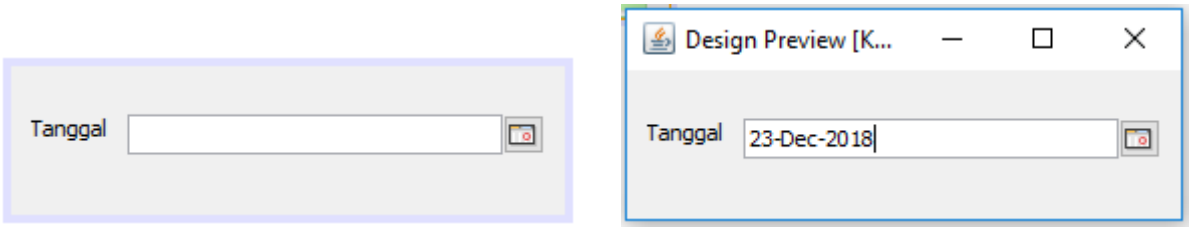
11.3.7 jCalender

jCalender merupakan komponen tools yang juga tersedia di Java, namun untuk menggunakannya di edito NetBeans, perlu sedikit konfigurasi karena kadangkala jCalender tidak ikut terinstal bersama dengan editornya. Untuk itu kita perlu mengunduhnya terlebih dahulu. Perlu diingat juga, selain jCalender, Java NetBeans menyediakan plugin lain diluar jCalender, yaitu DateChooser dan jdatepicker. Dalam modul ini akan disajikan jCalender untuk menambahkan data waktu (penanggalan). Untuk menambahkan plugin jcalender-1.4.jar ikuti panduan berikut :

1) Unduh plugin **jcalender-1.4.jar** dan ekstraksi
2) Tambahkan ke palette dengan cara **Tools - Palette - Swing/AWT Components**, kemudian klik **New Category..** ketik **Date,** tekan **Ok.**
3) Pilih **Date,** kemudian klik tombol **Add from JAR..,** cari file jcalender-1.4.jar yang telah diekstraksi sebelumnya
4) Pada kolom **Available Components,** pilih semua componen yang tersedia di kolom, pilih **Next,** pilih folder **Date** yang telah dibuat sebelumnya, selanjutnya klik Finish untuk mengakhiri.
5) Jika berhasil, perhatikan di jendela Palette, kategori Date dengan komponen jcalender akan ditampilkan disana. Selanjutnya plugin jcalender siap untuk digunakan

1) Menggunakan jCalender

Klik jcalender seperti komponen lain, kemudian atur format tanggal sesuai kebutuhan, variabel name bisa saja diubah, tetapi ini tidak terlalu penting.



(a) (b)

Gambar 10.25 (a) Sebelum diload, (b) Setelah diload

Selain tools-tools diatas, *Swing GUI Form* masih memiliki banyak komponen tools lain yang tersedia didalamnya, dan penerapannya, secara umum juga sama dengan komponen tools-komponen tools yang telah disajikan diatas.

11.4 Latihan

Setelah membaca dan memahami teori yang disampaikan diatas, selanjutnya pada bagian ini akan disajikan contoh latihan program dengan menggunakan komponen tools-komponen tools diatas. Harap menerapkannya agar target kompetensi matakuliah yang diharapkan dapat terpenuhi dengan baik.

11.4.6 Mendisain Form Aplikasi

Untuk memudahkan praktisnya, lakukan langkah-langkah berikut :

1) Buatlah Project baru dengan nama LatihanGUI, klik menu File - New Project..
2) Tambahkan File baru (jFrame), klik menu File - New
3) Pada kolom Categories pilih **Swing GUI Forms**, pada File Type pilih **jFrame Form**
4) Klik **Next,** pada kotak **Class Name**, ganti **NewJFrame** dengan **KlinikBersalin,** selanjutnya klik **Finish** untuk mengakhiri
5) Langkah selanjutnya, rancanglah Form berdasarkan data komponen tools berikut, dan ubah properti sesuai data pada tabel berikut

TOOLS	TEXT PROPERTIES/TITLE	VARIEBLE NAME	FONT / KETERANGAN LAIN
jFrame (Form)			
jLabel1	KLINIK BERSALIN SAHABAT SEHATI	jLabel1	Time New Roman, 18 Bold
jLabel2	Nama Pasien	jLabel2	Default
jLabel3	Kelas	jLabel3	Default
jLabel4	Inap	jLabel4	Default
jLabel5	Layanan	jLabel5	Default

jLabel6	Biaya Persalinan	jLabel6	Default
jLabel7	Masuk	jLabel7	Default
jLabel8	Keluar	jLabel8	Default
jLabel9	Lama Inap	jLabel9	Default
jLabel10	Total Pembayaran	jLabel10	Default
jTextField1	(kosong)	txtnmpasien	-
jTextField2	(kosong)	txtbiayaInap	-
jTextField3	(kosong)	txtbiayaLayanan	-
jTextField4	(kosong)	txtbiayaSalin	-
jTextField5	(kosong)	txtlamaInap	-
jTextField6	(kosong)	txttotalBiaya	-
jComboBox1	-	cbKelas	Kelas = {A, B, C};
jRadioButton1	Dokter	rbDokter	-
jRadioButton2	Bidan	rbBidan	-
jDataChooser1	-	dcMasuk	-
jDataChooser2	-	dbKeluar	-
jButton1	Hitung	btnHitung	-
jButton2	Bersikan Kotak	btnBersih	-
jButton3	Keluar	btnKeluar	-

Rancangan form Aplikasi akan seperti pada gambar form dibawah ini. Disain form bolehs aja berbeda dari contoh, akan tetapi komponennya, disarankan sama dengan gambar supaya lebih mudah dalam pengkodean programnya.

6) **Mendeklarasikan variabel**

Masuk jendela editor, tambahkan variabel berikut dibawah nama **class** public

```
1.  int lminap;
2.  double inap, layanan, persalinan, totalbyr, totalinap,
    totallayanan;
```

7) **Mendaftarkan Pilihan Kelas Layanan**

Klik kanan **Form** - pilih **events** - **Window** - pilih **windowActivated**, selanjutnya ketik kode program berikut untuk menambahkan daftar pilihan Kelas

```
1.  private void formWindowActivated(java.awt.event.WindowEvent evt)
    {
2.          cbKelas.setSelectedItem("Pilih Kelas");
3.          cbKelas.addItem("Pilih Kelas");
4.          cbKelas.addItem("A");
5.          cbKelas.addItem("B");
6.          cbKelas.addItem("C");
7.      }
```

8) **Menyeleksi Pilihan Kelas Layanan**

Klik kanan **cbKelas** - pilih **Events** - **Action** - **actionPerformed**, kemudian ketik kode program berikut

```
1.  private  void  cbKelasActionPerformed(java.awt.event.ActionEvent
    evt) {
2.          if(cbKelas.getSelectedItem().equals("Pilih Kelas")){
3.              txtbiayaInap.setText("");
4.              txtbiayaLayanan.setText("");
5.          }
6.          else if(cbKelas.getSelectedItem().equals("A")){
7.             inap=300000;
8.              layanan=100000;
9.          }
10.         else if(cbKelas.getSelectedItem().equals("B")){
11.             inap=200000;
12.             layanan=70000;
13.         }
14.         else {
15.             inap=100000;
16.             layanan=50000;
17.         }
18.
19.         txtbiayaInap.setText(String.valueOf(inap));
20.         txtbiayaLayanan.setText(String.valueOf(layanan));
21.     }
```

9) **Menentukan Biaya Layanan Persalinan**

Klik kanan **rbDokter** - pilih **Events** - **Mouse** - **mouseClicked**, selanjutnya tambahkan kode program berikut

```
1. private  void  rbDokterMouseClicked(java.awt.event.MouseEvent  evt)
   {
2.     if(rbDokter.isSelected()) {
3.         rbBidan.setSelected(false);
4.         persalinan=1000000;
5.     }
```

```
6.          txtbiayaSalin.setText(String.valueOf(persalinan));
7.      }
```

Lakukan hal yang sama dengan **rbBidan,** lalu tambahkan kode program berikut

```
1. private void rbBidanMouseClicked(java.awt.event.MouseEvent evt) {
2.         if(rbBidan.isSelected()) {
3.          rbDokter.setSelected(false);
4.          persalinan=750000;
5.      }
6.          txtbiayaSalin.setText(String.valueOf(persalinan));
7.      }
```

10) **Menentukan Lama Inap dan Pelayanan**

Tambahkan terlebih dahulu package import java.text.SimpleDateFormat; diatas nama class public. Lalu, klik kanan **txtlamaInap** - pilih **Events** - **Mouse** - **mouseClicked,** selanjutnya tambahkan kode program berikut

```
1.  private  void  txtlamaInapMouseClicked(java.awt.event.MouseEvent
    evt) {
2.      try{
3.              String tglmasuk, tglkeluar;
4.              SimpleDateFormat ft = new SimpleDateFormat("dd-MMM-
    yyyy");
5.
6.              tglmasuk = ft.format(dcMasuk.getDate());
7.              int tm = Integer.parseInt(tglmasuk.substring(0,2));
8.              tglkeluar = ft.format(dcKeluar.getDate());
9.              int                          tk                        =
    Integer.parseInt(tglkeluar.substring(0,2));
10.             lminap = tk-tm;
11.             txtlamaInap.setText(String.valueOf(lminap));
12.         }catch(NumberFormatException ex){
13.             System.out.println(""+ex.getMessage());
14.         }
15.         totalinap=lminap*inap;
16.         totallayanan=lminap*layanan;
17.     }
```

11) **Menghitung total biaya persalinan**

Untuk menghitung total biaya persalinan, klik kanan **btnHitung** - pilih **Events** - **Action** - **actionPerformed**, atau klik ganda btnHitung, selanjutnya tambahkan kode program berikut

```
1. private  void  btnHitungActionPerformed(java.awt.event.ActionEvent
   evt) {
2.          totalbyr=persalinan+totallayanan+totalinap;
3.          txttotalBiaya.setText(String.valueOf(totalbyr));
```

```
4.      }
```

12) **Membersihkan Kotak input**

Klik ganda btnBersih, lalu tambahkan kode program berikut

```
1. private void btnBersihActionPerformed(java.awt.event.ActionEvent
   evt) {
2.         txtnmpasien.setText("");
3.         cbKelas.setEnabled(false);
4.         rbDokter.setEnabled(false);
5.         rbBidan.setEnabled(false);
6.         txtbiayaInap.setText("");
7.         txtbiayaLayanan.setText("");
8.         txtbiayaSalin.setText("");
9.         dcMasuk.setEnabled(false);
10.           dcKeluar.setEnabled(false);
11.           txtlamaInap.setText("");
12.           txttotalBiaya.setText("");
13.    }
```

13) **Menutup Aplikasi**

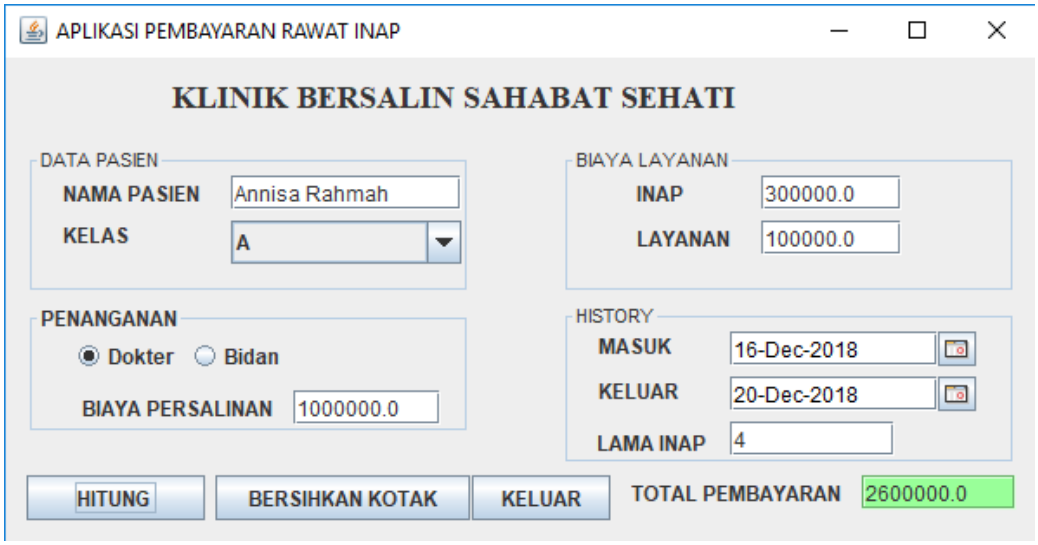
Untuk menutup aplikasi, klik ganda btnKeluar, lalu tambahkan kode program berikut

```
1.  private void
    btnKeluarActionPerformed(java.awt.event.ActionEvent evt) {
2.          this.dispose();
3.      }
```

14) **Menguji Program**

Dengan mengacu kepada rancangan program diatas, jalankan program Run Project (F6), selanjutnya ikuti langkah berikut :

1) Isi Nama pasien
2) Pilih Kelas Pelayanan, maka biaya inap dan layanan akan muncul secara otomatis
3) Pilih penanganan persalinan (Dokter atau Bidan), maka biaya persalinan akan tampil
4) Atur tanggal masuk dan keluar klinik
5) Klik kotak Lama Inap, maka jumlah hari menginap dan pelayanan akan tampil
6) Selanjutnya tekan tombol Hitung untuk mendapatkan total biaya yang harus dibayarkan pasien
7) Keluaran program akan seperti pada gambar berikut



8) Tekan tombol Bersihkan Kotak untuk menginput data baru, dan tombol Keluar untuk menutup aplikasi

15) **Kode Program Lengkap**

```
1.   import java.text.SimpleDateFormat;
2.   // Class public
3.   public class KlinikBersalin extends javax.swing.JFrame {
4.       int lminap;
5.       double inap, layanan, persalinan, totalbyr, totalinap,
     totallayanan;
6.       public KlinikBersalin() {
7.           initComponents();
8.       }
9.
10.  // Mendaftarkan pilihan kelas
11.   private void formWindowActivated(java.awt.event.WindowEvent evt)
     {
12.          cbKelas.setSelectedItem("Pilih Kelas");
13.          cbKelas.addItem("Pilih Kelas");
14.          cbKelas.addItem("A");
15.          cbKelas.addItem("B");
16.          cbKelas.addItem("C");
17.      }
18.  // Membersihkan kotak input
19.      private                                                  void
     btnBersihActionPerformed(java.awt.event.ActionEvent evt) {
20.          txtnmpasien.setText("");
21.          cbKelas.setEnabled(false);
22.          rbDokter.setEnabled(false);
23.          rbBidan.setEnabled(false);
24.          txtbiayaInap.setText("");
25.          txtbiayaLayanan.setText("");
26.          txtbiayaSalin.setText("");
27.          dcMasuk.setEnabled(false);
28.          dcKeluar.setEnabled(false);
29.          txtlamaInap.setText("");
```

```
30.          txttotalBiaya.setText("");
31.      }
32.
33.  // Menentukan harga berdasarkan kelas pelayanan
34.      private                                                    void
     cbKelasActionPerformed(java.awt.event.ActionEvent evt) {
35.          if(cbKelas.getSelectedItem().equals("Pilih Kelas")){
36.              txtbiayaInap.setText("");
37.              txtbiayaLayanan.setText("");
38.          }
39.          else if(cbKelas.getSelectedItem().equals("A")){
40.             inap=300000;
41.              layanan=100000;
42.          }
43.          else if(cbKelas.getSelectedItem().equals("B")){
44.              inap=200000;
45.              layanan=70000;
46.          }
47.          else {
48.              inap=100000;
49.              layanan=50000;
50.          }
51.
52.          txtbiayaInap.setText(String.valueOf(inap));
53.          txtbiayaLayanan.setText(String.valueOf(layanan));
54.      }
55.
56.  // Menentukan biaya berdasarkan penanganan Bidan
57.   private  void  rbBidanMouseClicked(java.awt.event.MouseEvent  evt)
     {
58.         if(rbBidan.isSelected()) {
59.          rbDokter.setSelected(false);
60.          persalinan=750000;
61.      }
62.          txtbiayaSalin.setText(String.valueOf(persalinan));
63.      }
64.
65.  // Menentukan biaya berdasarkan penanganan dokter
66.      private   void   rbDokterMouseClicked(java.awt.event.MouseEvent
     evt) {
67.      if(rbDokter.isSelected()) {
68.          rbBidan.setSelected(false);
69.          persalinan=1000000;
70.      }
71.          txtbiayaSalin.setText(String.valueOf(persalinan));
72.      }
73.
74.  // Menutup aplikasi
75.      private                                                    void
     btnKeluarActionPerformed(java.awt.event.ActionEvent evt) {
76.          this.dispose();
77.      }
78.
79.  // Menentukan lama (jumlah hari) pelayanan klinik
```

```
80.   private   void   txtlamaInapMouseClicked(java.awt.event.MouseEvent
     evt) {
81.      try{
82.              String tglmasuk, tglkeluar;
83.              SimpleDateFormat  ft  =  new  SimpleDateFormat("dd-MMM-
     yyyy");
84.
85.              tglmasuk = ft.format(dcMasuk.getDate());
86.              int tm = Integer.parseInt(tglmasuk.substring(0,2));
87.              tglkeluar = ft.format(dcKeluar.getDate());
88.              int tk = Integer.parseInt(tglkeluar.substring(0,2));
89.              lminap = tk-tm;
90.              txtlamaInap.setText(String.valueOf(lminap));
91.          }catch(NumberFormatException ex){
92.              System.out.println(""+ex.getMessage());
93.          }
94.          totalinap=lminap*inap;
95.          totallayanan=lminap*layanan;
96.      }
97.
98.  // Menentukan total biaya pelayanan
99.    private                                                    void
     btnHitungActionPerformed(java.awt.event.ActionEvent evt) {
100.         totalbyr=persalinan+totallayanan+totalinap;
101.         txttotalBiaya.setText(String.valueOf(totalbyr));
102.     }
103.
104. // Method static Form Aplikasi (kode bersifat otomatis/tidak
105. // membuatuhkan pengeditan oleh perancang program)
106.  public static void main(String args[]) {
107.         java.awt.EventQueue.invokeLater(() -> {
108.             new KlinikBersalin().setVisible(true);
109.         });
110.     }
111. }
```

MODUL 12

KONEKSI

12.1 Aplikasi

Pada akhirnya, pembelajaran pemrograman komputer adalah bagaimana merancang aplikasi yang interaktif, yang mudah dioperasikan oleh pengguna akhir. Aplikasi yang mudah digunakan tentu merujuk kepada aplikasi yang menggunakan antarmuka *(interface)* yang menarik, serta berbasis pada grafis *(grafical user interface : GUI).* Aplikasi berbasis grafis, sejak tahun 2000an, semakin hari semakin menarik bukan hanya dari sisi tampilan saja, tetapi juga dari *feature* dan teknologi yang terdapat pada aplikasi tersebut.

Jika kita merujuk kepada defenisi grafis itu, yakni berbentuk garis, huruf dan simbol, maka tidak salah jika aplikasi berbasis grafis pada suatu komputer juga demikian. Dan faktanya memang demikian. Aplikasi berbasis grafis saat ini sudah merebak. Jika kita amati, hampir semua teknologi komputer sudah menggunakan aplikasi-aplikasi berbasis garfis, baik perangkat itu adalah komputer, maupun perangkat bergerak (*mobile).*

Aplikasi-aplikasi grafis sengaja diciptakan untuk mempermudah, terutama bagi pengguna akhir suatu aplikasi. Karena, tidak semua pengguna mampu merancang aplikasi, sementara suatu aplikasi dikatakan lengkap apabila aplikasi tersebut mendukung fasilitas penyimpanan data. Disinilah aplikasi berbasis grafis memerankan manfaat yang besar. Dengan antarmuka grafis, proses penyortiran data ke dalam gundang data (basisdata) suatu aplikasi menjadi sangat mudah. Perancangan aplikasi yang mendukung basisdata, mengharuskan perancang memahami mode perancangan antarmuka sekaligus perancangan struktur basisdata, karena keduanya merupakan komponen yang berbeda yang dikombinasikan bersama. Pada tahap ini, kita akan mencoba mengintegrasikan kedua komponen tersebut dalam suatu contoh sederhana. Sebelum ke permasalahan inti, kita akan mengulas kembali teori konsep basisdata dengan struktur bahasa SQL *(Structure Query Language).*

12.2 Basisdata

Jika merujuk kepada suku kata ‘basisdata’ itu, kita mendapati kata ‘basis’ memiliki arti asas, dasar, dan dapat juga dikatakan dengan pangkalan. Arti kata itu mengacu kepada sifat dasar suatu informasi. Dimana, informasi berasal dari suatu kumpulan data yang terpisah-pisah diolah sedemikian rupa sehingga memiliki makna yang jelas, dan dapat dikembangkan di tahap selanjutnya. Pangkalan data ini pada dasarnya terpisah-pisah yang dibungkus dalam

suatu kolom-kolom yang disebut dengan tabel. Tabel inilah yang terkait satusama lainnya sehingga membuat kumpulan data terhubung dalam satu gudang, yang disebut basisdata.

12.2.1 *Tools* Basisdata

Dalam suatu aplikasi, perancangan basisdata adalah percangan struktur, yakni perancangan struktur tabel-tabel yang dibutuhkan untuk mendukung terbentuknya suatu informasi. Namun, pada panduan ini, kita akan mencoba menerapkan satu tabel saja. Setelah menyelesaikan ini, mahasiswa diharapkan mampu untuk mengembangkan ke basisdata yang lebih kompleks.

Seperti yang rencanakan diawal, yaitu bagaimana supaya antarmuka aplikasi dapat digunakan untuk menyortir (simpan) data ke dalam basisdata, maka kita perlu terlebih dahulu mendisain struktur basisdata. Perancangan basisdata ini, bergantung kepada editor perancangan basisdata yang digunakan. Pada panduan ini kita menggunakan editor yang dimiliki oleh MySql yang terintegrasi pada aplikasi server XAMPP.

12.2.2 Structure Query Language

Structure Query Language (SQL) merupakan bahasa atau perintah-perintah yang digunakan untuk merancang struktur basisdata. SQL merupakan bahasa standar yang umum digunakan pada hampir semua *tools* basisdata yang menggunakan prinsip Database Basisdata Management System (DBMS). Ada tiga jenis struktur query SQL yang umum digunakan, yang dikelompokkan berdasarkan kategorinya, yaitu Data Definition Language (DDL), Data Manipulation Language (DML), dan Data Control Language (DCL).

a) Data Definition Language

   Data Definition Language (DDL) merupakan query yang berfungsi untuk mendefenisikan suatu basis data. Defenisi yang dimaksud disini adalah penamaan basisdata, penamaan tabel, ukuran data, dan lain sebagainya. DDL memiliki beberapa perintah query yaitu CREATE, DROP, dan ALTER.

b) Data Manipulation Language

   Data Manipulation Language (DML) merupakan query yang berfungsi untuk memanipulasi data. Kata manipulasi merujuk kepada fungsi-fungsi yang berkaitan dengan menambahkan data, merubah data, dan menghapus data di dalam tabel. Untuk mengimplementasi fungsinya sebagai manipulasi data, DML memiliki beberapa perintah query seperti INSERT, DELETE, SELECT, UPDATE dan lain sebagainya.

c) Data Control Languge

Merupakan query yang berfungsi untuk memberikan akses kepada user ke suatu basisdata dan tabel. Diantara perintah yang spesifik dengan DCL adalah GRAND dan REVOKE.

12.2.3 Perancangan Basisdata

Perancangan basisdata umumnya dimulai dari analisa kebutuhan data. Analisa kebutuhan data dilakukan untuk menentukan kapasitasa basisdata, koleksi data dalam tabel, serta ukuran data disetiap tabelnya. Perancangan basisdata dimulai dari mendefinisikan basisdata, selanjutnya struktur tabel-tabel yang digunakan. Untuk merancang basisdata, menggunakan query-query berikut ini :

a) Mecinptakan basisdata, dengan perintah *query*

```
CREATE DATABASE namaBasisdata;
```

b) Membuat tabel, dengan perintah query berikut

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
        ....
);
```

Untuk menambahkan primary key pada tabel, parameter primary key dapat ditambahkan, dengan perintah berikut :

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype, primary key(column1)
        ....
);
```

12.2.4 Memanipulasi Basisdata

Seperti yang disampaikan diawal, bahwa suatu aplikasi yang lengkap adalah adanya dukungan penyimpanan data. Data dapat ditambahkan melalui antarmuka aplikasi melalui jendela formulir. Formulir aplikasi tidak serta merta dapat digunakan untuk menambahkan data, dibutuhkan beberapa tahap yakni tahap instalasi dan konfigurasi.

Yang dimaksud instalasi disini adalah instalasi *tools* basisdata seperti Ms. Access, MySql, Navicat, dan lain sebagainya. *Tools-tools* tersebut harus dikonfigurasi melalui *tools* pendukung berbasis *Application Program Interface (API)* serta konfigurasi yang bersifat *module* yang umumnya berbentuk kode perintah yang ditambahkan di antarmuka aplikasi. Konfigurasi formulir aplikasi dan *tools* basisdata mengacu kepada konfigurasi berupa kode perintah, yakni perintah menyimpan, mencari, mengubah dan menghapus. Konfigurasi formulir aplikasi menggunakan query SQL yang diantaranya sebagai berikut :

a) Menambahkan data, dengan perintah :

```
INSERT INTO table_name (column1, column2, column3,
...) VALUES (value1, value2, value3, ...);
```

b) Mencari data, dengan perintah sebagai berikut :

Mencari data berdasarkan kolom dapat menggunakan perintah berikut :

```
SELECT column1, column2, ...
FROM table_name;
```

Atau dengan perintah berikut ini apabila ingin mencari data berdasarkan nama tabel yang digunakan, yaikur :

```
SELECT * FROM table_name;
```

c) Merubah data, dengan perintah query sebagai berikut :

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

d) Menghapus data

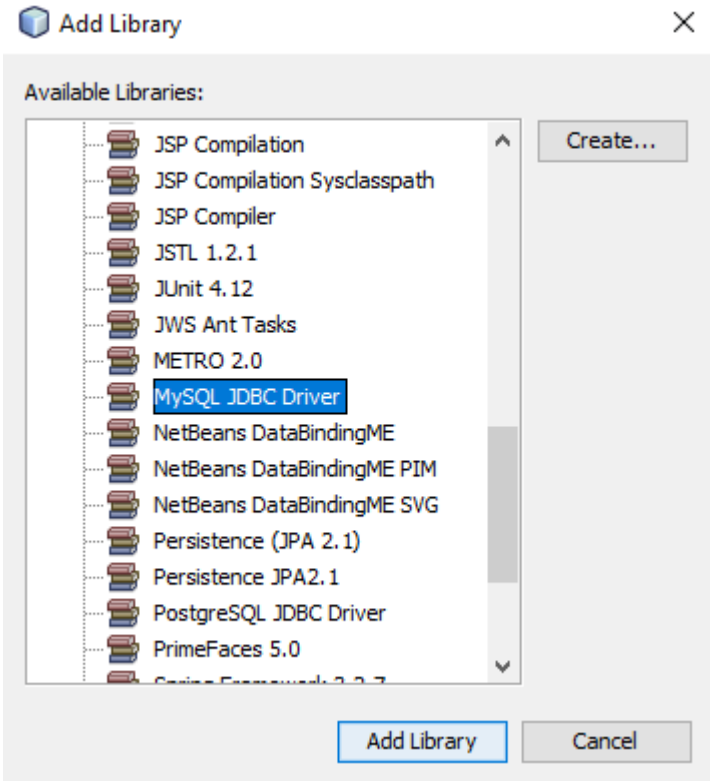
```
DELETE FROM table_name WHERE condition;
```

12.3 Konfigurasi Database

Konfigurasi database merupakan tahap menyisipkan atau manambahkan *package library class* yang terintegrasi di dalam file API. File API berbentuk file yang terdiri dari kumpulan *class-class* yang dapat digunakan secara bebas oleh pengguna, dalam hal ini adalah perancang aplikasi. File API tersebut dibungkus di dalam file yang memiliki format .jar, yaitu mysql-connector-java-5.1.24-bin.jar.

Tahap konfigurasi dilakukan di *folder library* di dalam file project java. Secara rinci konfigurasi database akan ditunjukkan tahap-demi tahap berikut ini :

1) Install driver mysql-connector-5.1.24 terlebih dahulu
2) Konfigurasikan *package library* dengan cara :
   - Klik kanan folder ***libraries*** di dalam project java yang sedang dikerjakan
   - Pilih *add library*
   - Pilih MySQL JDBC Driver, seperti pada gambar berikut, selanjutnya tekan *add library* :



12.4 Koneksi Basisdata

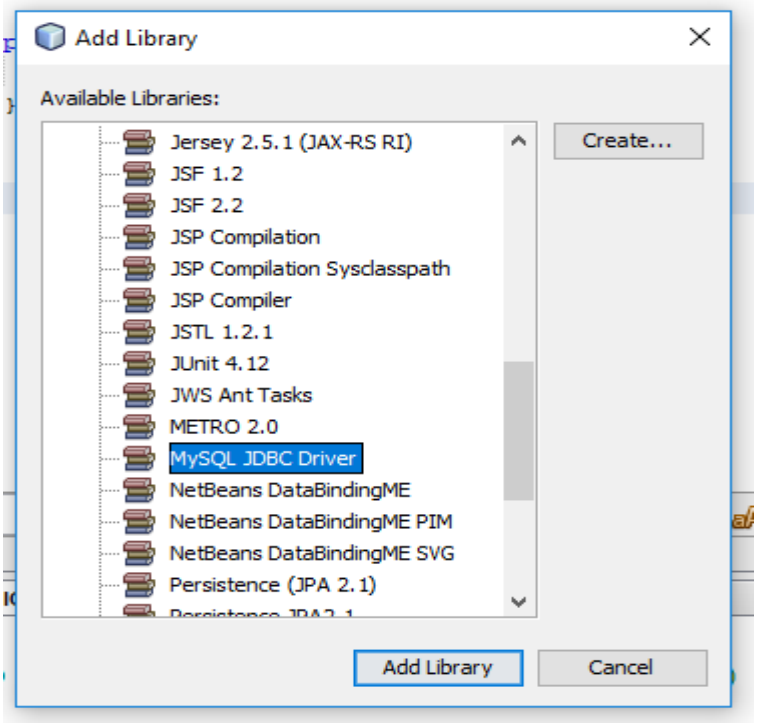
Koneksi basisdata merupakan hubungan satu-kesatuan antara antarmuka aplikasi dengan *tools* basisdata yang digunakan sebagai gudang data, sesuai nama basisdata yang digunakan. Koneksi basisdata berperan sebagai perantara, yang mengijinkan penguna memanipulasi data melalui formulir aplikasi. Koneksi basisdata, telah disinggung sedikit diawal, dimana koneksi dilakukan pada tahap konfigurasi. Kali ini akan dibahas konfigurasi antarmuka aplikasi dengan *tools* basisdata.

Koneksi basisdata menggunakan beberapa *package library* sesuai dengan metode koneksi yang digunakan. Pada bahasa Java, metode yang umum adalah menggunakan metode Java Database Connectivity (JDBC). Dimana jdbc merupakan antarmuka API, yang

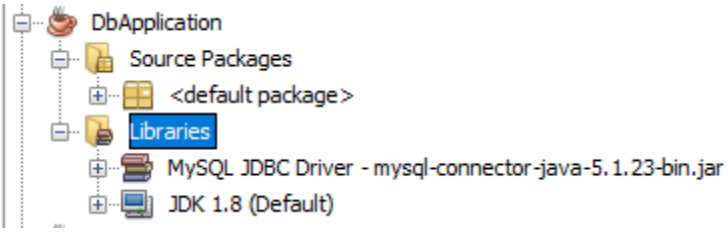
terpasang pada driver connector mysql-java. Dalam metode JDBC ini akan digunakan beberapa *library class* yang diantaranya import java.sql.Connection, import java.sql.Statement, import java.sql.DriverManager, dan *package library class* lainnya, yang di-import di dalam aplikasi pada saat pengkodean program. *Package* tersebut dilampirkan di baris awal kode program.

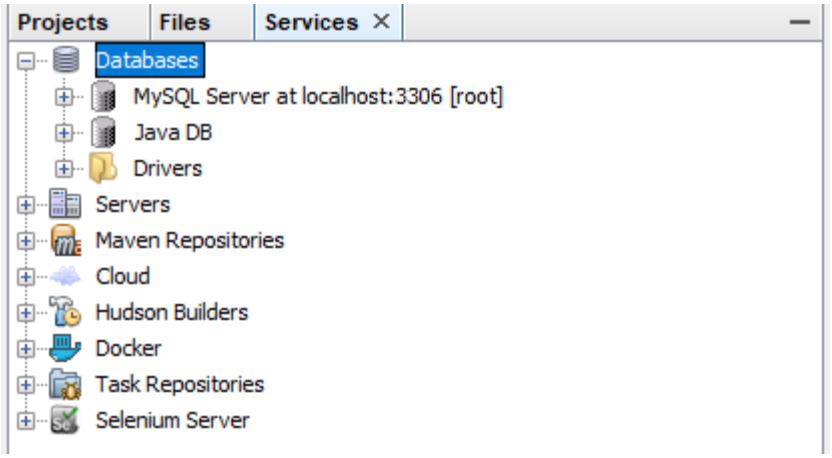
Ada banyak model atau algoritma untuk mengoneksikan antarmuka aplikasi dengan *tools* basisdata, kita mudah mendapatkannya di internet. Diantara model koneksi basisdata yang tersebar bebas di internet, berikut ini akan disajikan model algoritma koneksi basisdata yang paling mudah dipahami. Adapun langkah-langkahnya sebagai berikut :

1) Konfigurasikan mysql jdbc driver dengan cara, klik kanan library pada sisi kiri, lalu pilih *Add Library*

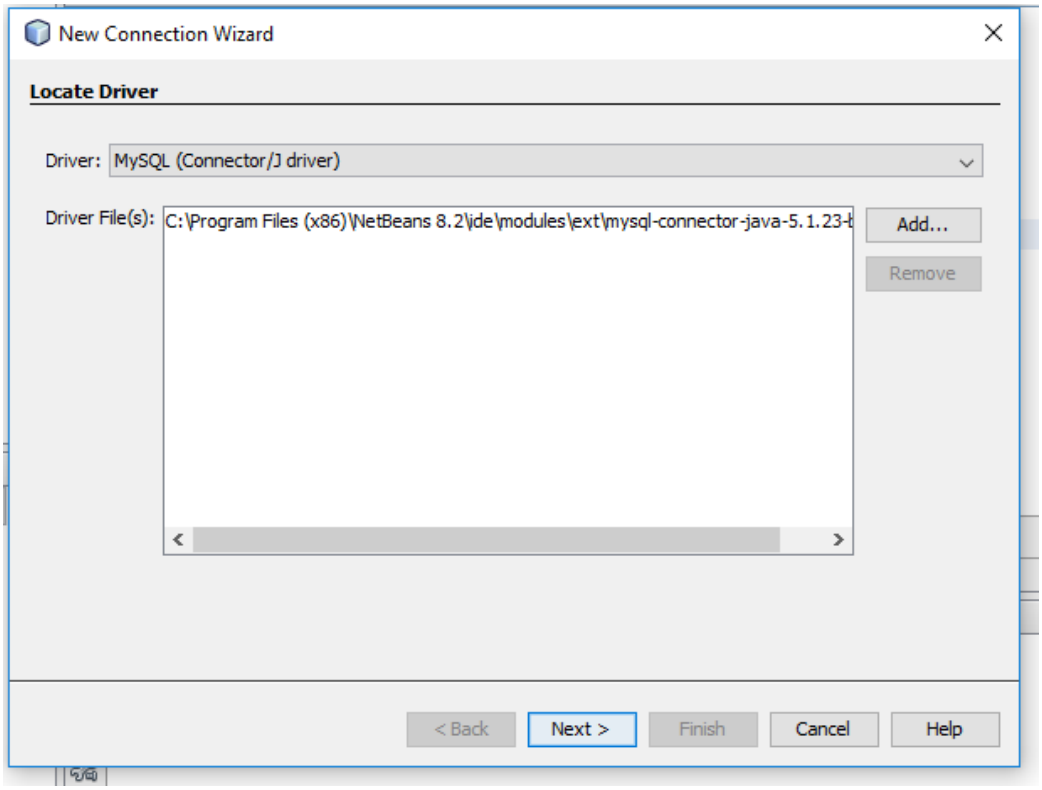


Jika berhasil maka driver mysql jdbc akan terdaftar ke dalam aplikasi yang sedang digunakan seperti pada gambar dibawah berikut ini :

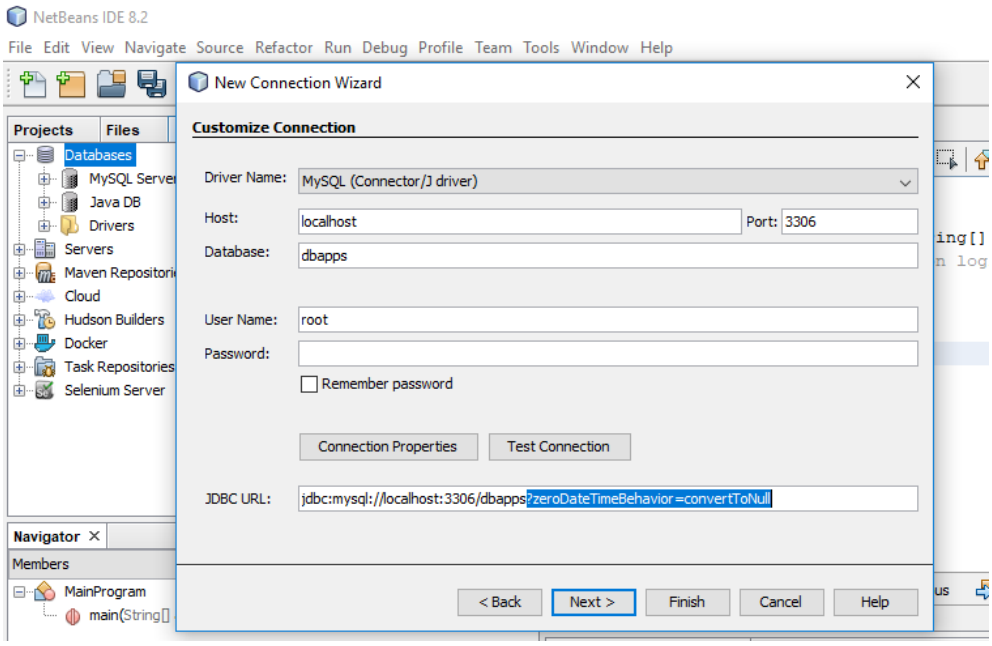


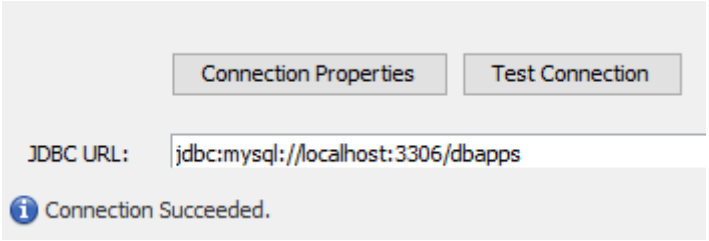


2) Jika driver mysql jdbc telah dikenali, selanjutnya kita dapat membuat database melalui netbean seperti pada langkah-langkah yang disajikan pada gambar-gambar berikut

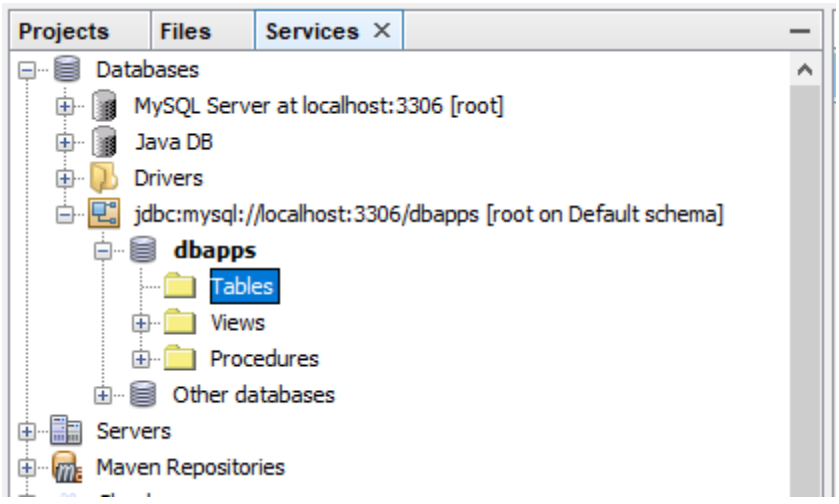


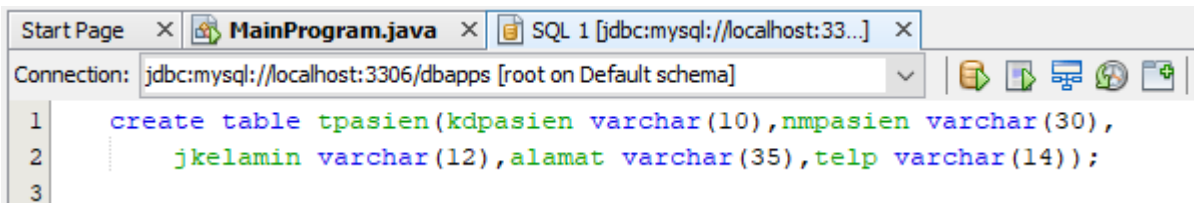
Ikuti langkah demi langsung sesuai pada panduan gambar.

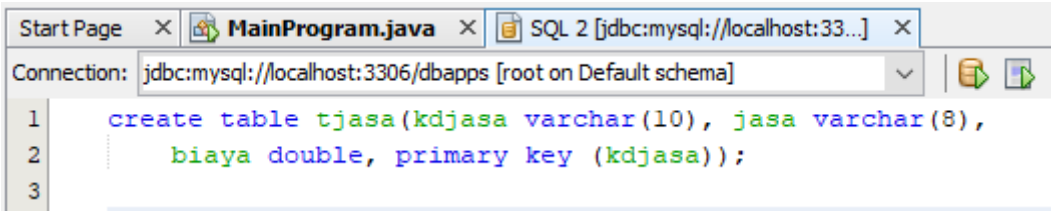


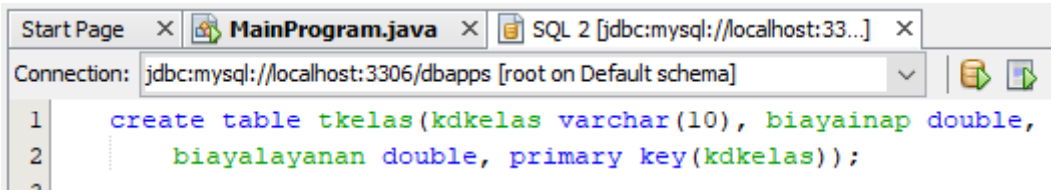


3) Untuk mengakhiri, teknan test connection lalu pilih next dan finish. Langsung selanjutnya buatlah database seperti pada langkah-langkah dibawah berikut :






Start Page
MainProgram.java
SQL 2 [jdbc:mysql://localhost:33...]
Connection: jdbc:mysql://localhost:3306/dbapps [root on Default schema]
1 create table tjasa(kdjasa varchar(10), jasa varchar(8),
2 biaya double, primary key (kdjasa));
3


Start Page
MainProgram.java
SQL 2 [jdbc:mysql://localhost:33...]
Connection: jdbc:mysql://localhost:3306/dbapps [root on Default schema]
1 create table tkelas(kdkelas varchar(10), biayainap double,
2 biayalayanan double, primary key(kdkelas));

DAFTAR PUSTAKA

[1] TechMetrix, 1999, *Java Application Servers Report,* TechMetrix Research, Burlington

[2] url : https://www.tutorialspoint.com/java/java_basic_syntax.htm, diakses pada 16 September 2019

[3] url : https://www.tiobe.com/tiobe-index/, diakses pada 16 Sepetember 2019

[4] P. Niemeyer, J. Knudsen, 2005, *Learning Java 3rd Edition*, O'Reilly Media, Inc., USA

[5] url : http://www.tutorialspoint.com/java/java_tutorial.pdf, diakses pada 23 September 2019, 13:34:52

[7] Jacquie Barker, 2005, *Beginning Java Objects From Concepts to Code, Second Edition,* Apress, USA

[8] Kishori Sharan, 2007, Beginning Java 8 Fundamentals, Apress, USA

[9] url : https://www.programtopia.net/cplusplus/docs/nested-loops, siakses pada Minggu, 6 Oktober 2019, 6:09:44

[10] url : https://www.sanfoundry.com/java-program-accept-array-elements-find-sum/, diakses pada Sabtu, 12 Oktober 2019, 21:09:44

[11] Robert Cecil Martin, 2002, *UML for Java Programmers,* Prentice-Hall, Inc., New Jersey

[12] https://beginnersbook.com/2013/04/difference-between-throw-and-throws-in-java/

[13] https://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws

(14] https://beginnersbook.com/2013/04/difference-between-throw-and-throws-in-java/

[15] https://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws