MODUL

Pemrograman Berorientasi Objek

Berbasis Problem Based Learning

Fitria Nur Hasanah, M.Pd
Cindy Taurusta, M.T

Java

Modul
Pemrograman Berorientasi Objek

Penulis:

Fitria Nur Hasanah

Cindy Taurusta

Diterbitkan oleh

UMSIDA PRESS

Jl. Mojopahit 666 B Sidoarjo

ISBN: 978-602-5914-58-4

Modul

Pemrograman Berorientasi Object

Penulis :

Fitria Nur Hasanah

Cindy Taurusta

ISBN : 978-602-5914-58-4

Editor :
Septi Budi Sartika
M. Tanzil Multazam

Copy Editor :
Fika Megawati

Design Sampul dan Tata Letak :
Mochamad Nashrullah

Penerbit :
UMSIDA Press

Redaksi :
Universitas Muhammadiyah Sidoarjo
Jl. Mojopahit No 666B
Sidoarjo, Jawa TImur

Cetakan pertama, Juli 2019

KATA PENGANTAR

Puji syukur dipanjatkan kehadirat Allah SWT yang telah memberikan kekuatan dan kemudahan dalam menyelesaikan penyusunan modul **Pemrograman Berorientasi Objek Berbasis Problem Base Learning**. Modul ini dilengkapi dengan kajian teori dan panduan praktik yang dapat dijadikan acuan dalam praktikum jaringan dan komunikasi data. Modul ini merupakan luaran dari penelitian hibah institusi. Ucapan terima kasih disampaikan kepada semua pihak yang telah memberikan bantuan dan dukungan terhadap proses penyusunan modul ini. Selanjutnya penulis berharap semoga modul ini bermanfaat khususnya bagi penulis sendiri dan umumnya bagi mahasiswa Prodi Pendidikan Teknologi Informasi (PTI) UMSIDA

.

Sidoarjo, Juli 2019

Penulis

DAFTAR ISI

1 INSTALASI DAN SETTING JAVA DI WINDOWS

TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

1. Melakukan instalasi dan setting Java Development Kit.
2. Menggunakan Jcreator sebagai editor pemrograman
3. Menjalankan (eksekusi) program Java sederhana

PERALATAN PRAKTIKUM

1. Personal Komputer
2. *Software Java Netbeans*

TEORI

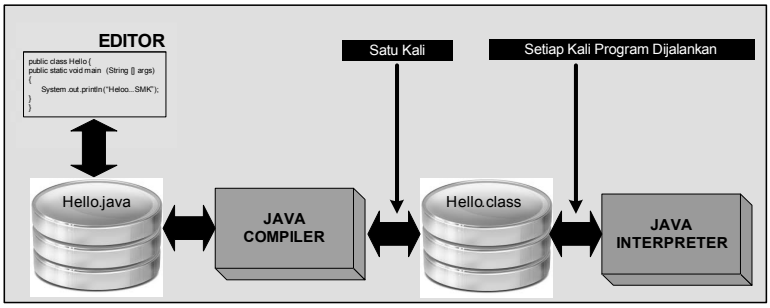
Penulisan code program JAVA dapat dilakukan dengan berbagai editor, dimulai dari yang paling sederhana yaitu notepad. Editor atau IDE (*Integrated Development Environment*) untuk Java, yang lainnya adalah:

a. NetBeans (*open source- Common Development and Distribution License* (CDDL))
b. NetBeans yang disponsori Sun Microsystems, yang dilengkapi dengan GUI Editor.
c. Eclipse JDT (*open source- Eclipse Public License*)
d. Eclipse dibuat dari kerja sama antara perusahaan-perusahaan anggota 'Eclipse
e. Oracle JDeveloper (free)
f. Xinox JCreator (ada versi berbayar maupun free)

Untuk pengembangan aplikasi berbasis Java, maka kegiatan pertama kali yang dilakukan adalah dengan menginstall software aplikasi java atau JSDK (*Java Software Development Kit* . JSDK atau JDK adalah sebuah aplikasi yang dibuat oleh perusahaan Sun Microsystems untuk membuat dan memodifikasi program java.

Fase-Fase pemrograman Java

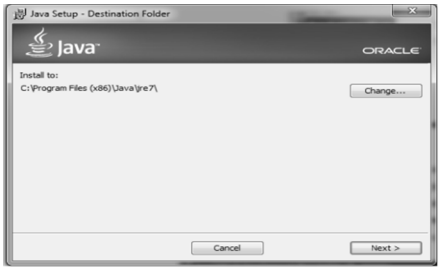
Gambar 1 menjelaskan aliran proses kompilasi dan eksekusi sebuah program Java.



Gambar Fase dari sebuah program Java

A. PRAKTIK INSTALASI

1. Java Standart Development Kit (SDK) tersedia untuk di download pada situs Web software Java Sun Microsystem pada :http://java.sun.com.
2. Open folder tempat file-file instalasi Java SDK
3. Jalankan setup program java (contoh: **jdk-7u5-nb-7_1_2-windows-ml**)
4. Muncul kotak dialog awal instalasi JDK



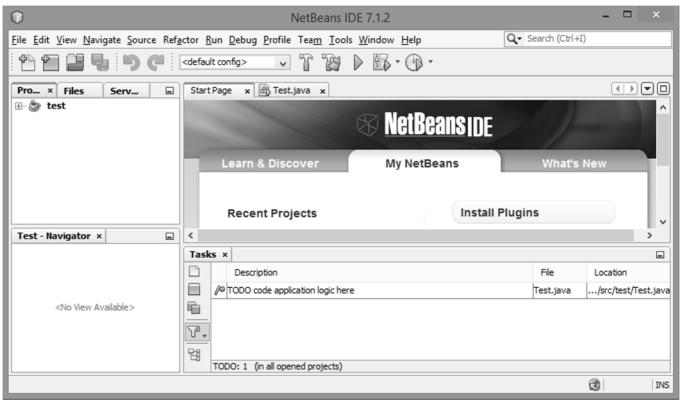
Gambar Kota dialog awal instalasi java

Tahapan-tahapan proses instalasi Java SDK dapat dilakukan dengan mudah dengan mengikuti petunjuk proses instalasi dengan menekan button next sampai pada tahap finish. Saat instalasi selesai, muncul kotak dialog yang memberitakan bahwa instalasi Java SDK lewatkan dengan mengklik tombol ***Finish***

Gambar Aplikasi java berhasil diinstal

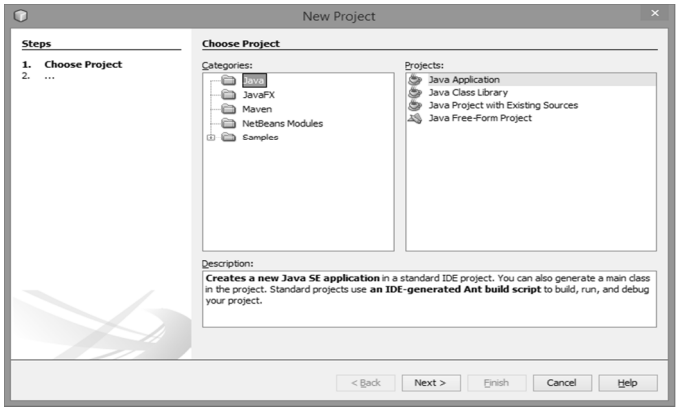
Tanda NetBeans sedang dalam proses membuka modul-modul yang diperlukan untuk

membuat aplikasi.

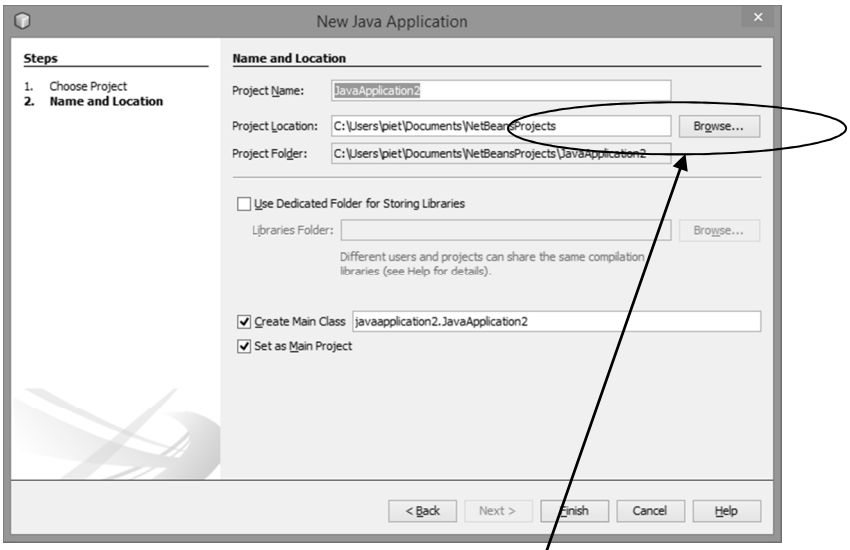


Gambar Aplikasi java dijalankan pertama kali

Untuk membuat program pertama kali buat file - new project, pastikan cursor berada pada folder (categories) java, pada menu project pilih java application. Klik next

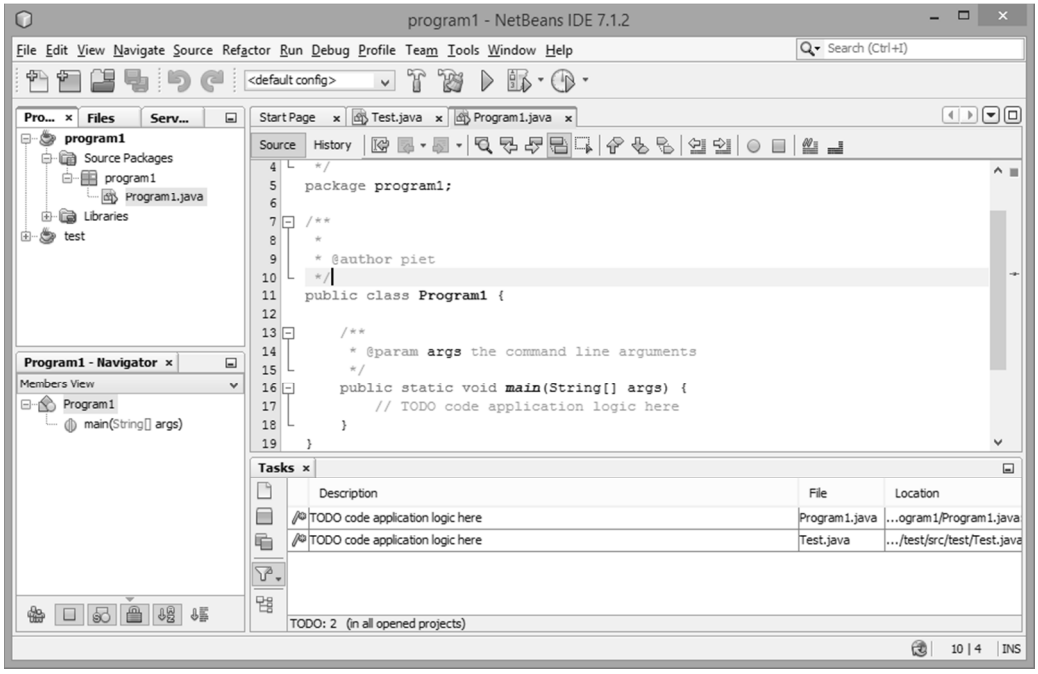


Gambar Tampilan project baru



Gambar Memberi nama dan menentukan lokasi penyimpanan

IDE NetBeans mengharuskan membuat *new Project* terlebih dahulu sebelum menulis *script* program java. Dengan cara klik File new Project , langkah berikutnya memilih aplikasi *Java Aplication*. File dengan *extension .java* dibuat untuk memulai menulis program java.pilih lokasi penyimpanan file project java pada menu project location.



Gambar Lembar kerja java (netbeans)

2 PENGENALAN PEMROGRAMAN BERORIENTASI OBYEK

TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

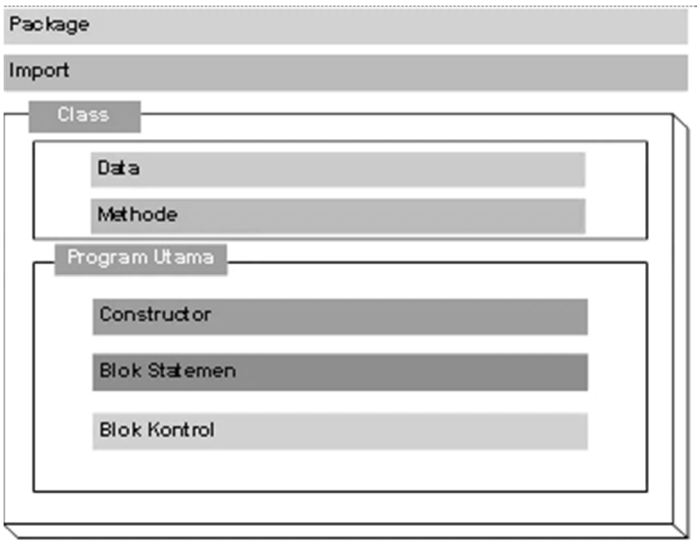
1. Mahasiswa mampu mengidentifikasi bagian dasar dari program java
2. Mahasiswa mampu membedakan tipe data dasar, tipe variabel
3. Mahasiswa mampu mengembangkan program java sederhana
4. Mahasiswa mampu mengalisa program java

PERALATAN PRAKTIKUM

1. Personal Komputer
2. *Software Java Netbeans*

TEORI

1) Bagan dasar program java



Gambar Bagian –bagian pemrograman Java

✓ **Package**

Perintah java yang digunakan untuk memberitahukan bahwa suatu class adalah anggota dari package, sedangkan nama Package dapat berupa susunan direktori tempat dimana file class disimpan atau nama folder.

✓ **Import**

Perintah import digunakan untuk memberitahukan kepada program untuk mengacu pada class-class yang terdapat pada package tersebut dan bukan menjalankan class-class tersebut**.**

- ✓ **Class**

  Merupakan bentuk logis yang menjadi landasan bangun seluruh bahasa pemograman berorientasi object. Class mendefinisikan bentuk dan perilaku object. Class merupakan contoh abstrak dari sebuah object yang telah terbentuk dari proses penyederhanaan. Kemudian contoh nyata atau perwujudan dari sebuah object dinamakan instance.

- ✓ **Data dan Methode**

  Data merupakan identitas yang berupa variabel yang menjelaskan properti dari class. Metoda adalah sekumpulan instruksi untuk menjalankan data yang diberi nama dan dapat dipanggil dari manapun di dalam program dengan menuliskan nama metoda tersebut.

- ✓ **Program utama**

  Salah satu metoda yang paling penting di dalam bahasa Java adalah metoda main. Metoda main harus dideklarasikan sendiri oleh programmer di dalam sebuah kelas. Kelas yang mempunyai metoda main disebut dengan kelas main (main class), Interpreter Java akan meminta metoda main saat program aplikasi dieksekusi.

```
package helloworld;

public class HelloWorld {

    //program utama
    public static void main(String[] args) {
     //menampilkan kalimat PTI Bisa
        System.out.println("PTI Bisa");
    }

      }
```

Keterangan :

Baris pertama kode:

public class HelloWorld

nama class : **Helloworld**.

Tiga baris selanjutnya menandakan adanya komentar Java. Komentar adalah sesuatu yang digunakan untuk mendokumentasikan setiap bagian dari kode yang ditulis. Komentar bukan merupakan bagian dari program itu sendiri, tetapi digunakan untuk tujuan dokumentasi.

```
/**
* Program Pertama
*/
```

Komentar dinyatakan dengan tanda "/*" dan "*/". Segala sesuatu yang ada diantara tanda tersebut diabaikan oleh compiler Java, dan mereka hanya dianggap sebagai komentar.

public static void main(String[] args) {

Mengindikasikan nama suatu method dalam class **Helloworld** yang bertindak sebagai **method utama.** Method utama adalah titik awal dari suatu program Java. Baris selanjutnya juga merupakan komentar,

//Menampilkan kalimat PTI Bisa!!!

Baris selanjutnya,

System.out.println("PTI Bisa!!!");

menampilkan teks “HelloWorld!” pada layar. Perintah **System.out.println(),** menampilkan teks yang diapit oleh tanda *doublepute* (“”) pada layar. Dua baris terakhir yang terdiri atas dua kurung kurawal digunakan untuk menutup method utama dan masing-masing class secara berurutan.

E. Tipe Data Primitif

Bahasa pemrograman Java mendefinisikan delapan tipe data primitif. Diantaranya adalah boolean (untuk bentuk logika), char (untuk bentuk tekstual), byte, short, int, long (integral), double and float (floating point).

1. ***logika - boolean***

   Tipe data boolean diwakili oleh dua pernyataan : true dan false. Sebagai contoh adalah, boolean result = true; Contoh yang ditunjukkan diatas, mendeklarasikan variabel yang dinamai **result** sebagai tipe data **boolean** dan memberinya nilai **true**.

2. ***teksual – char***

   Tipe data character (char), diwakili oleh karakter single Unicode. Tipe data ini harus memiliki ciri berada dalam tanda *single quotes*(’ ’). Sebagai contoh,

   ‘a’ //Huruf a
   ‘\t’ //A tab
   Untuk menampilkan karakter khusus seperti ' (single quotes) atau " (double quotes), menggunakan karakter escape \. Sebagai contoh,
   '\'' //untuk single quotes
   '\"' //untuk double quotes

   Mereka memiliki literal yang terdapat diantara tanda double quotes(“”). Sebagai contoh,
   String message=“Hello world!”

3. ***Integral –byte, short, int & long***

   Tipe data integral dalam Java menggunakan tiga bentuk- yaitu desimal,oktal atau heksa desimal. Contohnya,

   2 //nilai desimal 2
   077 //angka 0awal mengindikasikan nilai oktal
   0xBACC //karakter 0x mengindikasikan nilai heksadesimal

   Tipe-tipe integral memiliki default tipe data yaitu int. Anda dapat merubahnya ke bentuk long dengan menambahkan huruf l atau L

4. ***Floating Point –float dan Double***

Tipe Floating point memiliki double sebagai default tipe datanya. Floating-point literal termasuk salah satunya decimal point atau salah satu dari pilihan berikut ini:

E or e //(add exponential value) F or f //(float)

Contohnya :

3.14 //nilai floating-point sederhana (a double)
6.02E23 //A nilai floating-point yang besar

Tabel Tipe Data Primitif

Grup	Type Data	Size	Min Value	Max Value
Integral	byte	8 bits	-128	128
	short	16 bits	-32768	32768
	int	32 bits	-2147483648	2147483648
	long	64 bits	-9223372036854775808	9223372036854775808
Real	float	32 bits	± 1.40239846E-45	±3.40282347E+8
	double	64 bits	±4.94065645841246544E-324	±1.79769313486231570E+308
Karakter	char	16 bits	\u0000	\uFFFF
Boolean	boolean	n/a	true atau false	

A. Variabel

Variabel adalah item yang digunakan data untuk menyimpan pernyataan objek. Variabel memiliki tipe data dan nama. Tipe data menandakan tipe nilai yang dapat dibentuk oleh variabel itu sendiri. Nama variabel harus mengikuti aturan untuk identifier.

a. Deklarasi dan Inisialisasi Variabel

Untuk deklarasi variabel adalah sebagai berikut,

<data tipe><**name>** [=initial value];

Catatan: Nilainya berada diantara <> adalah nilai yang disyaratkan, sementara nilai dalam tanda [ ] bersifat optional. Berikut ini adalah contoh program yang mendeklarasikan dan menginisialisasi beberapa variabel,

Listing Program

```
short x;
int umur;
float gaji;
```

Inisialisasi variabel dapat dilakukan dengan memberikan nilai pada variabel yang telah dideklarasikan, contoh :

Listing Program

```
int x=21;
double d = 3.5;
```

b. ***Variabel Reference dan Variabel Primitif***

Pada program java, terdapat dua type variabel **variabel reference** dan **variabel primitif** .**Variabel primitif** adalah variabel dengan tipe data primitif. Mereka menyimpan data dalam lokasi memori yang sebenarnya dimana variabel tersebut berada. **Variabel Reference** adalah variabel yang menyimpan alamat dalam lokasi memori. Yang menunjuk ke lokasi memori dimana data sebenarnya berada. Sebagai contoh, apabila kita mempunyai dua variabel dengan tipe data int dan String.

Listing Program

```
int no = 10;
String nama = "PTI ";
```

PRAKTIKUM

1) Type data karakter : **Contoh1.Java**

```
public class Contoh1 {

    public static void main(String[] args) {
    char ch1 = 65;
    char ch2 = 'B';

    System.out.println("ch1 = " + ch1);
    System.out.println("ch2 = " + ch2);
    }

}
```

a) Jalankan source kode di atas, bagaimana hasilnya?

……

……

b) Lakukan analisis pada output program jika tanda petik koma (;) dihilangkan, apa yang terjadi?

……

……

2) Penerapan method

Buat project baru dengan nama **Barang** seperti di bawah ini :

```
1   package barang;
2
3   public class Barang {
4    //pendeklarasian variabel dan inisialisasi variabel
5       String merk = "Sepatu Adidas" ;
6       int jumlah = 100;
7
8        /**
9        * main program
10       */
11      public static void main(String[] args) {
12
13
14      }
15  }
```

a) Jalankan program, Jelaskan pendapat Anda tentang hasil program!

……

……

b) Tambahkan method dengan nama **tampilkanbarang,** seperti listing program di bawah ini

```
1   package barang;
2
3   public class Barang {
4    //pendeklarasian variabel dan inisialisasi variabel
5       String merk = "Sepatu Adidas" ;
6       int jumlah = 100;
7
8       //definisi method tampilkanbarang()
9       public void tampilkanbarang(){
10        //perintah untuk menampilkan/mencetak merk
11        System.out.println("merk :" +merk);
12
13        //perintah untuk menampilkan/mencetak jumlah
14        System.out.println("jumlah:" +jumlah);
15           }
16
17       //main program
18      public static void main(String[] args) {
19
20
21      }
22  }
```

c) Tambahkan listring program pada **main program**, seperti di bawah ini

```
1   package barang;
2
3   public class Barang {
4    //pendeklarasian variabel dan inisialisasi variabel
5       String merk = "Sepatu Adidas" ;
6       int jumlah = 100;
7
8       //definisi method tampilkanbarang()
9       public void tampilkanbarang(){
10        //perintah untuk menampilkan/mencetak merk
11        System.out.println("merk :" +merk);
12
13        //perintah untuk menampilkan/mencetak jumlah
14        System.out.println("jumlah:" +jumlah);
15           }
16
17       //main program
18      public static void main(String[] args) {
19          // instanstiasi objek Contoh1
20            Barang barang = new Barang();
21
22            //perintah untuk memanggil method tampilkancontoh1()
23            //perintah untuk menampilkan data Contoh1
24            barang.tampilkanbarang();
25      }
26  }
```

Lakukan analisis terhadap listing program:

a. Bagaimana output dari program tersebut?

……

……

……

b. Jelaskan fungsi dari method **tampilkanbarang**!

……

……

……

c. Jika pada **baris ke 6** type data diubah menjadi **double,** maka output dari program adalah?

……

Tuliskan alasan jawaban Anda!

……

……

d. Jelaskan fungsi dari baris ke **24!**

……

……

TUGAS

Buatlah listing program untuk **kelas mahasiswa** yang menampilkan method dari mahasiswa tersebut. Tunjukkan keyword, tipe data dasar, dan tipe variabel di dalamnya.

Dikusikan dengan anggota kelompok Anda. Baca dengan cermat dan teliti setiap soal dibawah ini. Kemudian berdasarkan uraian materi diatas tulislah jawabannya pada lembar jawaban test yang telah disediakan.

1. Coba amati sebuah obyek di lingkungan sekitar Anda. Selanjutnya tentukan package, class, data, dan method-nya.
2. Sebutkan keyword, tipe data dasar, dan tipe variabel dari sebuah obyek yang Anda amati.

c. Lembar Jawaban

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

3 Dasar dan Aturan PBO (Operator Logika)

TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

1. Mengidentifikasi operator dalam program Java.
2. Menyajikan dalam perbedaan antara syntax error dan runtime error.

PERALATAN PRAKTIKUM

1. Personal Komputer
2. *Software Java Netbeans*

TEORI

OPERATOR

Terdapat beberapa tipe operator pada java, yaitu operator aritmatika, operator relasi, operator logika, dan operator kondisi. Operator ini mengikuti bermacam-macam prioritas yang pasti sehingga compilernya akan tahu yang mana operator untuk dijalankan lebih dulu dalam kasus beberapa operator yang dipakai bersama-sama dalam satu pernyataan.

1. Operator Aritmatika

Berikut ini adalah dasar operator aritmatik yang dapat digunakan untuk membuat suatu program Java,

Tabel 3. Operator Aritmatika dan Fungsi-Fungsinya

Operator	*Penggunaan*	*Keterangan*
+	*op1+ op2*	*Menambahkan op1 dengan op2*
*	*op1*op2*	*Mengalikan op1 dengan op2*
/	*op1/op2*	*Membagi op1 dengan op2*
%	*op1%op2*	*Menghitung sisa dari pembagian op1 denganop2*
-	*op1-op2*	*Mengurangkan op2 dari op1*

2. Operator Increment dan Decrement

Dari sisi operator dasar aritmatika, Java juga terdiri atas operator *unary increment* (++) dan operator *unary decrement* (--). Operator increment dan decrement menambah dan mengurangi nilai yang tersimpan dalam bentuk variabel angka terhadap nilai 1.

Sebagai contoh, pernyataan,

count = count + 1
count++;

Tabel 4. Operator Increment dan Decrement

Operator	***Penggunaan***	***Keterangan***
++	*op++*	*Menambahkan nilai 1 pada op; mengevaluasi nilai op sebelum diincrementasi/ ditambahkan*
++	*++op*	*Menambahkan nilai 1 pada op; mengevaluasi nilai op setelah diincrementasi/ ditambahkan*
--	*op--*	*Mengurangkan nilai 1 pada op; mengevaluasi nilai op sebelum didecrementasi/ dikurangkan*
--	*--op*	*Mengurangkan nilai 1 pada op; mengevaluasi nilai op setelah didecrementasi/ dikurangkan*

Operator increment dan decrement dapat ditempatkan sebelum atau sesudah operand. Ketika digunakan sebelum operand, akan menyebabkan variabel diincrement atau didecrement dengan nilai1, dan kemudian nilai baru digunakan dalam pernyataan dimana dia ditambahkan. Ketika operator increment dan decrement ditempatkan setelah operand, nilai variabel yang lama akan digunakan lebih dulu dioperasikan lebih dulu terhadap pernyataan dimana dia ditambahkan. Sebagai contoh,

Listing Program 1	**Listing Program 2**

int i =10;
int j = 3;
int k = 0;
k = **++j** + i;
	int i = 10,
	int j = 3;
	int k = 0;
	k = j++ + i;

3. Operator Relasi

Operator Relasi membandingkan dua nilai dan menentukan keterhubungan diantara nilai- nilai tersebut. Hasil keluarannya berupa **nilai boolean** yaitu true atau false.

Tabel 5. Operator Relasi

Operator	***Penggunaan***	***Keterangan***
>	*op1>op2*	*op1 lebih besar dari op2*
>=	*op1>= op2*	*op1 lebih besar dari atau sama dengan op2*
<	*op1<op2*	*op1 kurang dari op2*
<=	*op1<= op2*	*op1kurang dari atau sama dengan op2*
==	*op1== op2*	*op1 sama dengan op2*
!=	*op1!= op2*	*op1 tidak sama dengan op2*

4. *Operator logika*

Operator logika memiliki satu atau lebih operand Boolean yang menghasilkan nilai boolean. Terdapat enam operator logika yaitu : && (logika AND), & (Boolean logika AND), || (logika OR), | (Boolean logika inclusive OR), ^ (Boolean logika exclusive OR), dan ! (logika NOT). Pernyataan dasar untuk operasi logika adalah x1 op x2, dimana x1,x2 dapat menjadi pernyataan boolean. Variabel atau konstanta, dan op adalah salah satu dari operator &&, &, ||, | atau ^. Tabel kebenaran yang akan ditunjukkan selanjutnya, merupakan kesimpulan dari hasil dari setiap operasi untuk semua kombinasi yang mungkin dari x1 dan x2.

5. *Operator Kondisi (?:)*

Operator kondisi **?:** adalah operator ternary. Berarti bahwa operator ini membawa tiga argumen yang membentuk suatu ekspresi bersyarat. Struktur pernyataan yang menggunakan operator kondisi adalah, **exp1?exp2:exp3**

Dimana nilai exp1 adalah suatu pernyataan Boolean yang memiliki hasil yang salah satunya harus berupa nilai true atau false. Jika exp1 bernilai true, exp2 merupakan hasil operasi. Jika bernilai false, kemudian exp3 merupakan hasil operasinya. Berikut ini adalah flowchart yang menggambarkan bagaimana operator ?: bekerja,

ERROR PADA JAVA

1. *Syntax Error*

Syntax Error biasanya terjadi karena kesalahan penulisan. Mungkin kekurangan sebuah perintah di Java atau lupa untuk menulis tanda titik koma pada akhir pernyataan. Java mencoba untuk megisolasi error tersebut dengan cara menunjukkan baris dari kode dan terlebih dahulu karakter yang salah dalam baris tersebut.

Kesalahan umum lainnya adalah dalam kapitalisasi, ejaan, penggunaan dari karakter khusus yang tidak benar, dan penghilangan dari pemberian tanda baca yang sebenarnya. *Syntax error*, merupakan jenis error yang paling mudah dideteksi. *Syntax error* yang terjadi pada kode program disebabkan kode yang diketik tidak sesuai dengan aturan/tata cara penulisan yang dimiliki oleh bahasa pemrograman yang digunakan. Contoh syntax error :

a. Pada OOP, baris kode harus selalu diakhiri dengan (;). Jika tanda ini tidak disertakan maka akan terjadi kesalahan, karena tidak sesuai dengan aturan pengkodean pada OOP.

b. Kesalahan penulisan keyword (perintah baku yang telah disediakan oleh bahasa pemrograman).

2. Runtime Error

Sebuah program yang berhasil dikompile belum tentu berhasil dijalankan. Inilah yang dinamakan ***Runtime error***, kesalahan ini tidak akan ditampilkan sampai kita menjalankan program tersebut. Hal ini bisa saja terjadi misalnya dikarenakan struktur yang dibuat programmer tidak jelas atau mungkin tidak logis. seperti pembagian dengan nilai nol (*division by zero error*) atau aplikasi mencoba mengakses network drive yang tidak terhubung. Sintaks benar, logika benar, tetapi pada saat eksekusi sesuatu terjadi yang akan menginterupsi proses. Tipe *error* ini biasanya terjadi saat *event* atau *syntax error* yang tidak terdefinisi menyebabkan terjadinya situasi yang membuat program *crash* atau *hang* tanpa diduga. Untuk mengantisipasi run-time error programmer harus menulis kode program untuk menangani *exception* sehingga aplikasi tidak akan di *halt*. *Exception* merupakan class khusus yang digunakan untuk mengkomunikasikan status error antara berbagai bagian aplikasi.

3. Logical Error

Logical error (kesalahan logika) terjadi pada saat aplikasi di *compile* dan dieksekusi dengan benar tetapi hasilnya tidak sesuai dengan yang diharapkan. Kesalahan ini merupakan kesalahan logika pemrograman. *Logical Error* adalah jenis kesalahan yang paling sulit ditemukan karena tidak ada pesan kesalahan (tidak ada indikasi dimana *error* terjadi). Contoh sederhana adalah terjadi kesalahan hasil perhitungan.

PRAKTIKUM

1) <u>Operator Aritmatika</u>

Buat project baru dengan nama **operator.java,** tuliskan program berikut ini dan simpan.

```
public static void main(String[] args) {
    // TODO code application logic here

System.out.println("Operasi aritmetika " + "pada tipe integer");
int a = 2 + 1
int b = a - 1
int c = a * b
int d = c / 3;
System.out.println("Nilai a: " + a)
System.out.println("Nilai b: " + b)
System.out.println("Nilai c: " + c)
System.out.println("Nilai d: " + d);
System.out.println();
```

```
System.out.println("Operasi aritmetika " + "pada tipe floating-point");
double fa = 2 + 1
double fb = fa - 1
double fc = fa * fb
double fd = fc / 3;
System.out.println("Nilai fa: " + fa);
System.out.println("Nilai fb: " + fb);
System.out.println("Nilai fc: " + fc);
System.out.println("Nilai fd: " + fd);
}
```

Lakukan kompilasi pada file tersebut dan amati hasilnya?

……

……

Kenapa terjadi error pada saat kompilasi? Termasuk jenis error apakah yang terjadi pada program tersebut?

……

……

Lakukan modivikasi untuk memperbaiki kesalahan diatas sehingga program tersebut dapat berjalan dengan baik.

……

……

……

……

2) <u>Operator Decremen dan Incremen</u>

Buat project baru dengan nama **decrement.java,** tuliskan program berikut ini dan simpan.

```
public static void main(String[] args) {
    // TODO code application logic here

System.out.println("Operasi aritmetika " + "pada tipe integer");
int a=5;
System.out.print("Pre-decrement");
System.out.print("a\t: " + a);
System.out.print("--a\t: " + --a);
System.out.println("a\t: " + a);

int b=5;
System.out.println("\nPost-decrement");
System.out.println("b\t: " + b);
System.out.println("b--\t: " + b--);
   }
```

Lakukan kompilasi pada file tersebut dan amati hasilnya?

……

……

Jelaskan perbedaan antara print dengan println?

……

……

3) Contoh modulus

Buat project baru dengan nama **modulus.java,** tuliskan program berikut ini dan simpan.

```
public static void main(String[] args) {
    // TODO code application logic here

int a=11, b=4;
int c = a % b;

double da = 13.75;
double dc = da % b;

System.out.println("Sisa bagi " + a + "/" + b + " adalah " + c);
System.out.println("Sisa bagi " + da + "/" + b + " adalah " + dc);
   }
```

Lakukan kompilasi pada file tersebut dan amati hasilnya?

……

……

……

4) Operator logika

Buat project baru dengan nama **logika.java,** tuliskan program berikut ini dan simpan.

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

System.out.println("Operasi AND");
System.out.println("true && true   = " + (true && true));
System.out.println("true && false  = " + (true && false));
System.out.println("false && true  = " + (false && true));
System.out.println("false && false = " + (false && false));

System.out.println("\nOperasi OR");
System.out.println("true || true   = " + (true || true));
System.out.println("true || false  = " + (true || false));
System.out.println("false || true  = " + (false || true));
System.out.println("false || false = " + (false || false));

System.out.println("\nOperasi XOR");
System.out.println("true ^ true    = " + (true ^ true));
System.out.println("true ^ false   = " + (true ^ false));
System.out.println("false ^ true   = " + (false ^ true));
System.out.println("false ^ false  = " + (false ^ false));

System.out.println("\nOperasi NOT");
System.out.println("!true          = " + (!true));
System.out.println("!false         = " + (!false));
   }
```

Lakukan kompilasi pada file tersebut dan amati hasilnya?

……

……

……

……

5) Operator Relasi

Buat project baru dengan nama **logika.java,** tuliskan program berikut ini dan simpan

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

int a=5, b=10;

System.out.println("a == b bernilai " + (a == b));
System.out.println("a != b bernilai " + (a != b));
System.out.println("a > b bernilai " + (a > b));
System.out.println("a < b bernilai " + (a < b));
System.out.println("a >= b bernilai " + (a >= b));
System.out.println("a <= b bernilai " + (a <= b));
   }
```

Lakukan kompilasi pada file tersebut dan amati hasilnya?

……

……

……

……

TUGAS

1. Buatlah program untuk menghitung suatu harga barang yang bernilai Rp. 200.000 dengan diskon 15%.
2. Buatlah program untuk menghitung volume balok dengan ukuran panjang = 4 cm, lebar 3 cm, dan tinggi 5 cm.

Lembar Kerja

……

……

……

……

……

4 Dasar dan Aturan PBO (Kondisi)

TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

1. Memahami struktur kontrol pemilihan (if, else, switch)
2. Menggunakan struktur kontrol pemilihan (if, else, switch) yang digunakan untuk memilih blok kode yang akan dieksekusi

PERALATAN PRAKTIKUM

1. Personal Komputer
2. *Software Java Netbeans*

TEORI

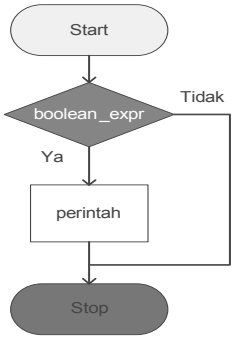
Struktur kontrol pemilihan adalah pernyataan dari Java yang mengijinkan user untuk memilih dan mengeksekusi blok kode spesifik dan mengabaikan blok kode yang lain.

1. Statement if

Pernyataan *if* akan menentukan sebuah pernyataan (atau blok kode) yang akan eksekusi jika dan hanya jika persyaratan bernilai benar *(true).* Bentuk dari pernyataan if,

Sintaks Perintah If	Sintaks Perintah If
`if(boolean_expression) statement;`	`if(boolean_expression) { statement1; statement2; }`

boolean_expression adalah sebuah penyataan logika *(true/false)* atau variabel bertipe *boolean*.



Gambar 4.1 Flowchart Statement if

2. *Statement if-else*

Pernyataan *if-else* digunakan apabila kita ingin mengeksekusi beberapa pernyataan dengan kondisi *true* dan pernyataan yang lain dengan kondisi *false*. Bentuk statement if-else,

Sintaks Perintah If

```
If  (boolean_expression)
statement;
```

Berikut ini contoh code **statement if-else,**

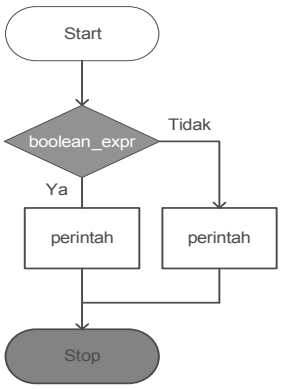
Listing Program

```
Int grade=68;
If (grade>60)
System.out.println("Congratulations!");
else
System.out.println("Sorry you failed");
```

Atau

Listing Program

```
intgrade=68;
if(grade>60)
{
System.out.println(“Congratulations!”);
System.out.println(“You passed!”);
}
Else  {
System.out.println("Sorry you failed");
}
```



Gambar 4.1 Flowchart Statement If-Else

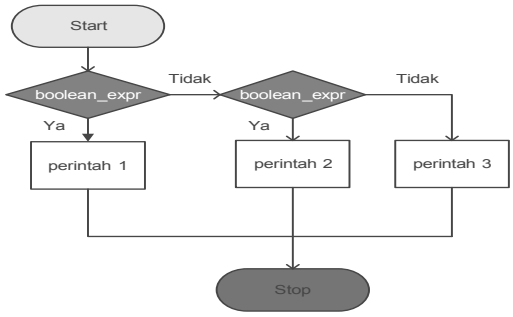
3. *Statement if-else-if*

Pernyataan pada bagian kondisi *else* dari blok *if-else* dapat menjadi struktur *if-else* yang lain. Kondisi struktur seperti ini mengijinkan kita untuk membuat seleksi persyaratan yang lebih kompleks. Bentuk statement *if-else if*.

Sintaks perintah If else If

```
if(boolean_expression1)
statement1;
else if(boolean_expression2)
statement2;
else
statement3;
```

else bersifat opsional dan dapat dihilangkan. Pada contoh yang ditampilkan diatas, jika boolean_expression1 bernilai true, maka program akan mengeksekusi statement1 dan melewati pernyataan yang lain. Jika boolean_expression2 bernilai true, maka program akan mengeksekusi statement2 dan melewati statement2.



Gambar 4.2. Flowchart Statement If-Else-If

Berikut ini contoh code statement if-else-if

Listing Program

```
Int grade=68;
if(grade>90)  {
System.out.println(" Excellent good!");
}
Else if(grade>60)
{
System.out.println("Very good!");
}
else{
System.out.println("Sorry you failed");}
```

4. Statement switch

Cara lain untuk membuat cabang adalah dengan menggunakan kata kunci ***switch***. *Switch* mengkonstruksikan cabang untuk beberapa kondisi dari nilai. Bentuk statement switch adalah sebagai berikut:

Sintaks Perintah Switch

```
switch(switch_expression)
{
Case case_selector1:
statement1;
statement2;
case case_selector2:
statement1;
statement2;
break;
default:
}
statement1;
statement2;
break;
```

PRAKTIKUM

Praktikum 1 (Struktur if satu Kondisi)

1) ***Ketikkan listing program di bawah ini:***

```
1   package struktur;
2
3   public class Struktur {
4
5       public static void main(String[] args) {
6           int grade = 68;
7            if( grade > 60 )
8            {
9                System.out.println("Congratulations!");
10           }
11      }
12      }
```

2) Bagaimana output dari program tersebut?

……

……

……

3) Jika nilai pada variable grade diubah menjadi 52, bagaimana hasilnya?

……

Lakukan analisis terhadap output program!

……

……

Praktikum 2 (Struktur If dua Kondisi)

Ketikkan listing program di bawah ini:

```
1   package struktur;
2
3   public class Struktur {
4
5       public static void main(String[] args) {
6           int a=1, b=10;
7
8       if (a < 5) {
9         System.out.println(a + " lebih kecil dari 5");
10      } else {  // (a >= 5)
11          System.out.println(a + " lebih besar dari 5");
12      }
13      }
14  }
```

Jelaskan output dari program di atas!

……

……

Praktikum 3 (Struktur tiga Kondisi)

Ketikkan listing program di bawah ini

```
1   package struktur;
2
3   public class Struktur {
4
5       public static void main(String[] args) {
6           int nilai = 86;
7
8       if (nilai > 85) {
9         System.out.println("Nilai akhir adalah A" );
10      }
11      else if (nilai>80) {
12         System.out.println("Nilai akhir adalah B");
13      }
14      else {
15        System.out.println("Nilai akhir adalah C");
16      }
17      }
18      }
```

Output dari coding program di atas adalah?

……

……

Berikan penjelasan tentang alur program tersebut!

……
……
……
……
……
……
……

*Jika variable nilai di ubah menjadi **nilai= 75,** Bagaimana hasilnya? Mengapa demikian?*

……
……
……
……

Lakukan modifikasi coding program sehingga syarat untuk mendapatkan nilai berubah menjadi : jika **nilai > 85** maka akan mendapat nilai **A**;
jika **nilai > 80** maka akan mendapat **nilai B-;**
jika **nilai > 75** maka akan mendapat nilai **B**;
jika **nilai > 70** maka akan mendapat **nilai C**;
jika **nilai > 65** maka akan mendapat **nilai C-;**
selain itu mendapatkan **nilai D**
Tuliskan listing program pada lembar kerja:

……
……
……
……
……
……
……
……
……
……
……
……
……

Praktikum 4. Switch Case

System akan menampilkan sesuai pilihan yang diinputkan, contoh listing programnya

```
1  package struktur;
2
3  public class Struktur {
4
5      public static void main(String[] args) {
6         int noHari = 7;
7
8      switch (noHari) {
9        case 1:
10         System.out.println("Hari ke-" + noHari + " adalah Minggu");
11         break;
12       case 2:
13         System.out.println("Hari ke-" + noHari + " adalah Senin");
14         break;
15       case 3:
16         System.out.println("Hari ke-" + noHari + " adalah Selasa");
17         break;
18       case 4:
19         System.out.println("Hari ke-" + noHari + " adalah Rabu");
20         break;
21       case 5:
22         System.out.println("Hari ke-" + noHari + " adalah Kamis");
23         break;
24       case 6:
25         System.out.println("Hari ke-" + noHari + " adalah Jum\'at");
26         break;
27       case 7:
28         System.out.println("Hari ke-" + noHari + " adalah Sabtu");
29         break;
30       default:
31         System.out.println("Tidak ada hari ke-" + noHari);
32     }
33     }
34 }
```

1) *Bagaimana hasil output coding program di atas ?*

   ……
   ……

2) *Jika nilai (input) variable **noHari** adalah selain **angka 1 - 7** bagaimana hasil output program? Berikan penjelasan dari analisis program tersebut?*

   ……
   ……
   ……
   ……
   ……

TUGAS

Tugas 1

Buatlah suatu program yang berfungsi mengecek suatu nilai. Jika nilai lebih dari 75 cetak “Lulus”, jika nilai kurang dari 75 cetak “Gagal”.

Tugas 2

Buatlah program pendaftaran mahasiswa baru di universitas yang memilih jurusan di universitas tersebut. Terdapat 5 program studi PTI, PGSD, PAUD, PBI, dan IPA. Jika pendaftar memilih salah satu jurusan, cetak nama jurusan yang dipilih dengan menggunakan **If** dan **switch case.**

Tuliskan listing program pada lembar kerja berikut

5 Dasar dan Aturan PBO (Perulangan)

TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

1. Memahami struktur kontrol pengulangan (while, do-while, for)
2. Menggunakan struktur kontrol pengulangan (while, do-while, for) untuk menjalankan blok tertentu pada program beberapa kali

PERALATAN PRAKTIKUM

1. Personal Komputer
2. *Software Java Netbeans*

TEORI

Struktur kontrol pengulangan adalah berupa pernyataan dari Java yang mengijinkan kita untuk mengeksekusi blok code berulang-ulang sesuai dengan jumlah tertentu yang diinginkan. Ada tiga macam jenis dari struktur kontrol pengulangan yaitu while, do- while, dan for-loops.

1. while loop

Pernyataan whileloop adalah pernyataan atau blok pernyataan yang diulang-ulang sampai mencapai kondisi yang cocok. Bentuk pernyataan while,

```
while(boolean_expression){ statement1; statement2;}
```

Pernyataan di dalam whileloop akan di eksekusi berulang-ulang selama kondisi boolean_expression bernilai benar (true).

Contoh pada kode dibawah ini.

```
int i=4; //inisialisasi awal
    while(i>0)
    {
    System.out.print(i);
    i--;
    }
```

Contoh di atas akan mencetak angka 4321 pada layar. Perlu dicatat jika bagian i--; dihilangkan, akan menghasilkan pengulangan yang terus menerus (infiniteloop). Sehingga, ketika menggunakan whileloop atau bentuk pengulangan yang lain, pastikan Anda memberikan pernyataan yang membuat pengulangan berhenti pada suatu kondisi.

Berikut ini adalah beberapa contoh while loop,

```
intx=0;
    while(x<10)
    {
    System.out.println(x);
    x++;
    }
```

2. *do-whileloop*

Do-while loop mirip dengan while-loop. Pernyataan di dalam do-whileloop akan dieksekusi beberapa kali selama kondisi bernilai benar (true). Perbedaan antara while dan do-whileloop adalah dimana pernyataan di dalam **do-while loop** akan dieksekusi sedikitnya satu kali.

Sintaks do-while loop

```
do{
    statement1;
    statement2;
    ...
    }while(boolean_expression);
```

Pernyataan di dalam do-while loop akan dieksekusi pertama kali, dan akan dievaluasi kondisi dari boolean_expression. Jika nilai pada boolean_expression tersebut bernilai true, pernyataan di dalam do-whileloop akan dieksekusi lagi. Contoh do-while loop:

```
    intx=0;
    do
    {
    System.out.println(x);
    x++;
    }while(x<10);
```

Contoh ini akan memberikan output 0123456789 pada layar.

3. *For loop*

Pernyataan forloop memiliki kondisi hampir mirip seperti struktur pengulangan sebelumnya yaitu melakukan pengulangan untuk mengeksekusi kode yang sama sebanyak jumlah yang telah ditentukan. Bentuk dari forloop,

```
for(InitializationExpression; LoopCondition;StepExpression)
{
statement1;
statement2;
... }
```

Berikut ini adalah contoh dari for loop,

```
int i;
for(i=0;i<10;i++)
{
System.out.print(i);
}
```

Pada contoh ini, pernyataan i = 0 merupakan inisialisasi dari variabel. Selanjutnya, kondisi < 10 diperiksa. Jika kondisi bernilai true, pernyataan didalam forloop dieksekusi. Kemudian, ekspresii ++ dieksekusi, lalu akan kembali pada bagian pemeriksaan terhadap kondisii < 10 lagi. Kondisi ini akan dilakukan berulang-ulang sampai mencapai nilai yang salah (false).

PRAKTIKUM

Praktikum 1 (For)

1) Ketikkan listing program di bawah ini

```
1  package perulangan;
2
3  public class Perulangan {
4      public static void main(String[] args) {
5          for (int i=0; i<10; i++) {
6            System.out.println("Java");
7        }
8        }
9  }
```

2) Beri penjelasan tentang hasil dari coding program

……
……
……
……
……

3) Modifikasi coding program pada point 1, sehingga hasil output coding program menghasilkan angka 1 sampai dengan 10, tuliskan listing program pada lembar yang tersedia!

```
run:
1 2 3 4 5 6 7 8 9 10 BUILD SUCCESSFUL (total time: 1 second)
```

……
……
……
……

……
……
……

4) Dengan menggunakan **for,** modifikasi program untuk membuat perulangan decrement (--) dengan menampilan angka 8 sampai dengan 4. Tuliskan listing programnya

```
run:
8 7 6 5 4 BUILD SUCCESSFUL (total time: 0 seconds)
```

……
……
……
……
……
……

Praktikum 2 (While)

1) Ketikkan listing program berikut

```
1   package perulangan;
2
3   public class Perulangan {
4       public static void main(String[] args) {
5        int i=0;
6       while (i<10) {
7         System.out.println(i);
8         i++;
9           }
10    }
11  }
```

2) Bagaimana hasilnya ? jelaskan output yang dihasilkan!

……
……

3) Modifikasi listing program dengan menggunakan while untuk menampilkan :

```
run:
8 7 6 5 4 BUILD SUCCESSFUL (total time: 0 seconds)
```

Tulis listing programnya!

……
……
……
……
……
……

4) Modifikasi listing program dengan menggunakan while untuk menampilkan:

```
run:
1 + 2 + 3 + 4 + 5 = 15
BUILD SUCCESSFUL (total time: 2 seconds)
```

Tuliskan listing programnya

……
……
……
……
……

Praktikum 3 (While)

Jalankan listing program di bawah ini:

```
int i=4;
while(i>0)
{
System.out.println(i);
i--;

}
```

1) Bagaimana hasilnya ?

……
……

2) Lakukan modifikasi program dengan menghapus **i-- ,** bagaimana hasilnya? Mengapa demikian? Lakukan analisis terhadap program !

……
……
……
……

Praktikum 4 (Do … While)

Jika ingin membuat perulangan angka 1 samapi dengan 4 dengan menggunakan do while

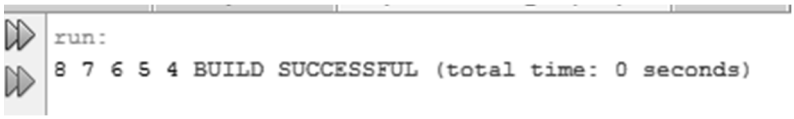
1) Ketikkan listing program berikut

```
1   package perulangan;
2
3   public class Perulangan {
4       public static void main(String[] args) {
5   int i=1;
6       do {
7         System.out.println(i);
8         i++;
9       } while (i < 5);
10    }
11  }
```

2) Bagaimana hasilnya ?

……
……

3) Modifikasi listing program dengan menggunakan do… while untuk menampilkan :



Tulis listing programnya!

……
……
……
……
……
……
……

TUGAS

1. Buatlah program untuk mencetak angka **genap** mulai dari 1 – 20.
2. Tulislah suatu program yang membaca sepuluh bilangan integer dan kemudian menampilan nilai terkecil dan terbesar dari sepuluh integer tersebut.

Tuliskan listing program pada lembar kerja berikut

……
……
……
……
……
……
……
……
……
……
……
……
……
……

6 Dasar dan Aturan PBO (Array)

TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

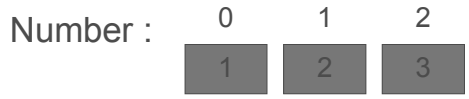
1. Memahami struktur kontrol array
2. Menggunakan struktur kontrol array satu dimensi dan multidimensi

PERALATAN PRAKTIKUM

1. Personal Komputer
2. *Software Java Netbeans*

TEORI

Pada bahasa pemrograman Java maupun dibahasa pemrograman yang lain, terdapat sebuah kemampuan untuk menggunakan satu variable yang dapat menyimpan beberapa data dan memanipulasinya dengan lebih efektif. Tipe variabel inilah yang disebut sebagai array.



Contoh dari Integer Array

Array adalah suatu wadah bentukan yang menyediakan penyimpanan sejumlah item yang bertipe sama. Array digunakan untuk mengelompokkan informasi yang berhubungan. Dalam Java, item dalam array selalu dinomori dari nol hingga nilai maksimum tertentu, yang nilainya ditentukan pada saat array tersebut dibuat.

Pendeklarasian Array

Array harus dideklarasikan seperti layaknya sebuah variabel. Pada saat mendeklarasikan array, Anda harus membuat sebuah daftar dari tipe data, yang diikuti oleh sepasang tanda kurung [], lalu diikuti oleh nama identifier-nya. Sebagai contoh,

```
int[]ages;
```

Atau Anda dapat menempatkan sepasang tanda kurung [] sesudah nama *identifier*. Sebagai

contoh,

```
Int ages[];
```

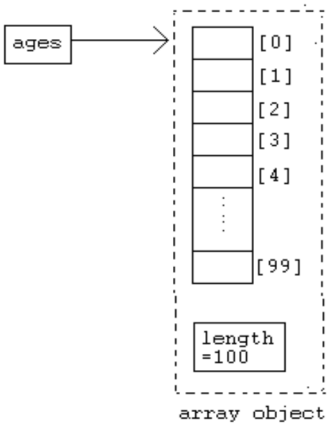
Setelah pendeklarasian array, kita harus membuat array dan menentukan berapa panjangnya dengan sebuah konstruktor. Proses ini di Java disebut sebagai *instantiation* (istilah dalam Java yang berarti membuat). Sebagai catatan bahwa ukuran dari array tidak dapat diubah setelah anda menginisialisasinya. Sebagai contoh,

Deklarasi Array

```
Int ages[];    //deklarasi
ages=new int[100];    //instantiate obyek
```

Atau bisa juga ditulis dengan,

```
//deklarasi dan instantiate obyek
Int ages[]=new int[100];
```



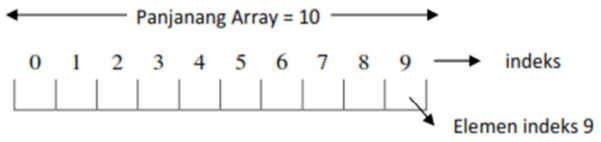
Pada contoh di samping, pendeklarasian tersebut akan memberitahukan kepada compiler Java, bahwa identifier ages akan digunakan sebagai nama array yang berisi data bertipe integer, dan dilanjutkan dengan membuat atau meng-*instantiate* sebuah array baru yang terdiri dari 100 elemen. Selain menggunakan sebuah pernyataan *new* untuk meng-instantiate array, Anda juga dapat mmendeklarasikan, membangun, kemudian memberikan sebuah nilai pada array sekaligus dalam sebuah pernyataan.

Contoh Listing Program

```
//Membuat sebuah array yang terdiri dari penginisialisasian
//4 variabel double bagi value{100,90,80,75}
double[]grades={100,90,80,75};
//Membuat sebuah array String dengan identifier days.Array
//ini terdiri dari 7 elemen.
String days[]={“Mon”,“Tue”,“Wed”,“Thu”,“Fri”,“Sat”,“Sun”};
```

b. Pengaksesan sebuah elemen array

Indeks adalah sebuah angka yang menyatakan *urutan* sebuah elemen pada suatu variabel array. Nomor indeks variabel array selalu dimulai dari 0 (nol), sehingga nomor indeks bagi elemen terakhir adalah sebesar (N-1), dimana N adalah jumlah total elemen. Untuk mengakses setiap elemen dalam variabel array cukup memanggil nomor indeksnya.



Gambar Elemen Array

Secara umum, deklarasi *array* dapat ditulis sebagai berikut :

```
type namavariabel[ ] ;
```

Atau

```
type[ ] namavariabel ;
```

Keterangan :

- type adalah mendeklarasikan tipe basis dari *array*. Dimana tipe basis ini menentukan data bagi masing-masing elemen yang membentuk *array*.
- namavariabel merupakan inisialisasi dari variabel yang akan didefenisikan.
- Untuk pembatas array maka diperlukan kurung siku [ ].

Sintaks Elemen Array

```
ages[0]=10;    //memberikan nilai 10 kepada elemen pertama array
System.out.print(ages[99]); //mencetak elemen array yang terakhir
```

Perlu diperhatikan bahwa sekali array dideklarasikan dan dikonstruksi, nilai yang disimpan dalam setiap anggota array akan diinisialisasi sebagai nol. Oleh karena itu, apabila Anda menggunakan tipe data seperti String, array tidak akan diinisalisasi menjadi string kosong. Untuk itu Anda tetap harus membuat String array secara eksplisit. Berikut ini adalah contoh kode untuk mencetak seluruh elemen di dalam array. Dalam contoh ini digunakanlah pernyataan *forloop*, sehingga kode kita menjadi lebih pendek.

```
Public class Array Sample
{
Public static void main (String[]args)
{
int[]ages=new int[100];
for(int i=0;i<100;i++)
{
System.out.print(ages[i]);
}
}
}
```

c. Array 1 Dimensi

Array 1 dimensi merupakan deklarasi array dengan satu variabel yang bertipe serupa. Untuk lebih memahami marilah Kita lihat contoh berikut:

```
class bulan{
        public static void main(String args[]){
//Deklarasi Variabel Array
int haribulan[];
//Penciptaan array index 12
haribulan = new int[12];
haribulan[0]  = 31;
haribulan[1]  = 29;
haribulan[2]  = 31;
haribulan[3]  = 30;
haribulan[4]  = 31;
haribulan[5]  = 30;
haribulan[6]  = 31;
haribulan[7]  = 31;
haribulan[8]  = 30;
haribulan[9]  = 31;
haribulan[10] = 30;
haribulan[11] = 31;

//Menampilkan Output
System.out.println("\nSeptember mempunyai "+haribulan[8]+" hari");
System.out.println("Februari mempunyai "+haribulan[1]+" hari\n");
}
}
```

Dari program di atas dapat Kita lihat bahwa arrray satu dimensi memiliki 12 index dengan masing- masing element diinisialisasikan satu persatu. Cara ini sering Kita jumpai dalam pemograman profesional. Dengan cara ini, array baru terbentuk ketika menggunakan *operator new()*. Di samping itu penciptaan atau inisialisasi nilai element pada array dapat juga langsung diberikan pada saat dideklarasikan. Proses ini hampir sama dengan inisialisasi variabel sederhana. Dimana nilai element array dikelompokkan ke dalam kurung kurawal dan dipisahkan dengan tanda koma. Hal ini akan secara otomatis menciptakan nilai element array sehingga *operator new()* tidak digunakan lagi. Untuk lebih memahami mari Kita lihat Contoh berikut

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

//Deklarasi Variabel Array dan sekligus pemberian nilai elemen

int haribulan[] = {31,29,31,30,31,30,31,31,30,31,30,31};

//Menampilkan Output
System.out.println("\nSeptember mempunyai "+haribulan[8]+" hari");
System.out.println("Februari mempunyai "+haribulan[1]+" hari\n");
}
}
```

d. Array Multidimensi

Array multidimensi diimplementasikan sebagai array yang terletak di dalam array, index array yang terdapat di dalam array. Pada beberapa kondisi diperlukan penulisan variabel

array yang menggunakan nomor indeks dua bilangan, misalnya pada aplikasi matrik. Data pada suatu matrik diketahui berdasarkan nilai baris dan kolomnya. Baris adalah sebuah bilangan dan kolom adalah sebuah bilangan juga. Pada dasarnya array 2 dimensi hampir sama dengan 1 dimensi. Hanya saja pada 2 dimensi terdapt indeks array di dalam array yang pertama. Di mana pada kelompok kurung siku yang pertama menyatakan elemen baris dan yang kedua menyatakan elemen kolom. Adapun bentuk umum array 2 dimensi sebagai berikut

type namavariabel[ ][ ];	*Atau*	*type[ ][ ] namavariabel ;*

Sebagai contoh,

Sintaks Array Multidimensi

```
Elemen512x128dariintegerarray
int[][]twoD = newint[512][128];
char[][][]threeD = new char[8][16][24];   //karakter array 8x16x24

//String array4 baris x 2 kolom
String[][]dogs= { {"terry","brown"}, {"Kristin","white"}, {"toby","gray"},
{"fido","black"}};
```

Untuk mengakses sebuah elemen di dalam array multidimensi, sama saja dengan mengakses array satu dimensi.

Sintaks Array Multidimensi

```
System.out.print(dogs[0][0]);
```

Array 1 Dimensi

Jika akan menampilan jumlah hari pada bulan tertentu dengan menggunakan array. Ketikkan listing program berikut:

```
1   package array;
2
3   public class Array {
4
5       public static void main(String[] args) {
6           // mendeklarasikan variabel bertipe array dengan tipe int
7       int [] jumlahHari;
8
9       // menentukan jumlah elemen array
10      jumlahHari = new int[12];
11
```

```
12          // mengisikan nilai dari setiap elemen array yang ada
13          jumlahHari[0] = 31;
14          jumlahHari[1] = 28;
15          jumlahHari[2] = 31;
16          jumlahHari[3] = 30;
17          jumlahHari[4] = 31;
18          jumlahHari[5] = 30;
19          jumlahHari[6] = 31;
20          jumlahHari[7] = 31;
21          jumlahHari[8] = 30;
22          jumlahHari[9] = 31;
23          jumlahHari[10] = 30;
24          jumlahHari[11] = 31;
25
26          // menampilkan salah satu elemen array
27          System.out.println("Bulan Maret memiliki " + jumlahHari[2] + " hari.");
28          }
29      }
```

1) Bagaimana hasilnya ? jelaskan output yang dihasilkan!

   .....
   .....
   .....
   .....

2) Jika pada baris ke **27** index pada variable jumlahHari diganti **[11] maka** output program yang dihasilkan adalah? Jelaskan alas an anda terkait output tersebut!

   .....
   .....
   .....
   .....

Array Multi Dimensi

Array multi dimensi 2 baris 2 kolom untuk menampilkan nama pemilik tas dan warna tas yang digunakan. Ketikkan listing program berikut:

```
1   package array2;
2
3   public class Array2 {
4
5       public static void main(String[] args) {
6          //Elemen512x128dariintegerarray
7                  int[][]twoD=new int[512][128];
8
9           //karakter array 8x16x24
10          char[][][]threeD=new char[8][16][24];
11
12          //String array4 baris x 2 kolom
13          String[][]bags= { {"terry","brown"}, {"Kristin","white"},{"toby","gray"},{"fido","black"} };
14
15            System.out.print(bags[0][1]);
16      }
17  }
```

1) Bagaimana hasilnya ? jelaskan output yang dihasilkan!

……
……
……

2) Jika ingin menampilkan warna tas yang dimiliki oleh toby, maka tuliskan listing programnya!

……
……
……
……
……
……

TUGAS

1. Buatlah program dengan array untuk mencetak angka genap mulai dari 1 – 20.
2. BUAT AGENDA HARIAN dengan menggunakan array, untuk menampilkan:
   Hari senin saya mengerjakan PR matematika
   Hari selasa saya mengantar ibu ke salon
   Hari rabu saya mencuci motor
   Hari sabtu saya mencuci baju

Tuliskan listing program pada lembar kerja berikut

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

7 Class dan Object

TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

1. Memahami perbedaan class dan obyek
2. Menyajikan pembuatan class

PERALATAN PRAKTIKUM

1. Personal Komputer
2. *Software Java Netbeans*

TEORI

Perbedaan *Class* dan Obyek

Pada dunia perangkat lunak**, sebuah obyek adalah** sebuah komponen perangkat lunak yang stukturnya mirip dengan obyek pada dunia nyata. Setiap obyek dibangun dari sekumpulan data (atribut) yang disebut **variable** untuk menjabarkan karakteristik khusus dari obyek, dan juga terdiri dari **sekumpulan method** yang menjabarkan tingkah laku dari obyek. Bisa dikatakan bahwa **obyek adalah** sebuah perangkatlunak yang berisi sekumpulan variable dan method yg berhubungan.Variabel dan method dalam obyek Java secara formal diketahui sebagai **variabel instance** dan **method instance**.

Class adalah sturktur dasar dari OOP. ***Class* terdiri dari dua tipe** dari anggota dimana disebut dengan *field* (attribut/properti) dan method. ***Field*** merupakan tipe data yang didefinisikan oleh *class*, sementara ***method*** merupakan operasi. Sebuah obyek adalah sebuah *instance* (keturunan) dari *class*.

✓ ***Instansiasi Class***

Untuk membuat sebuah obyek atau sebuah *instance* pada sebuah class. Kita menggunakan operator **new**. Sebagai contoh, jika anda ingin membuat *instance* dari *class string*, kita menggunakan kode berikut: String str2=new String(“Hello world!”);

Ini juga sama dengan, String str2= "Hello";

✓ ***Variabel Class dan Variabel Method***

Selain dari variabel *instance*, kita juga memungkinkan untuk mendefinisikan variabel dari *class*, yang nantinya variabel ini dimiliki oleh *class*. Ini berarti variabel ini dapat memiliki nilai yang sama untuk semua obyek pada *class* yang sama. Mereka juga disebut *static member variables*.

1) Pembuatan *Class*

Sebelum menulis *class* Anda, pertama pertimbangkan dimana Anda akan menggunakan *class* dan bagaimana *class* tersebut akan digunakan. Pertimbangkan pula nama yang tepat dan tuliskan seluruh informasi atau *property* yang ingin Anda isi pada *class*. Jangan **sampai terlupa untuk menuliskan secara urut *method* yang akan** Anda gunakan dalam *class*.

Dalam pendefinisian *class*, dituliskan:

Sintaks Pembuatan Class

```
<modifier>class<name>
{
<attributeDeclaration>*
<constructorDeclaration>*
<methodDeclaration>*
}
```

Dimana :

<modifier> adalah sebuah *access modifier*, yang dapat dikombinasikan denganti pe*modifier* lain. Pada bagian ini, kita akan membuat sebuah *class* yang berisi *record* dari mahasiswa. Jika kita telah mengidentifikasikan tujuan dari pembuatan *class*, maka dapat dilakukan pemberian nama yang sesuai. Nama yang tepat pada *class* ini adalah StudentRecord.

Untuk mendefinisikan *class*, kita tuliskan:

```
Public class StudentRecord
{
//area penulisan kode selanjutnya
}
```

dimana,

Public	-	*Class* ini dapat di akses dari luar *package //modifier*
Class	-	*Keyword* yang digunakan untuk pembuatan *Class* dalam Java
StudentRecord	-	*Identifier yang menjelaskan class*

1) Deklarasi Atribut

Dalam pendeklarasian atribut,kita tuliskan:

modifier><type><name>[=<default_value>];

Langkah selanjutnya adalah mengurutkan atribut yang akan diisikan pada *class.* Untuk setiap informasi, urutkan juga tipe data yang yang tepat untuk digunakan.

✓ *Instance Variable*

Jika kita telah menuliskan seluruh atribut yang akan diisikan pada *class*, selanjut nya kita akan menuliskannya pada kode.Jika kita menginginkan bahwa atribut–atribut tersebut adalah unik untuk setiap *object* (dalam hal ini untuk setiap mahasiswa), maka kita harus mendeklarasikannya sebagai *instance variable*

Sintaks Deklarasi Atribut

```
Public class StudentRecord
{
Private String name;
Private String address;
Private int age;
Private double mathGrade; private double englishGrade;
private double scienceGrade; private double average;
}
```

Private disini menjelaskan bahwa variabel tersebut hanya dapat diakses oleh *class* itu sendiri. *Object* lain tidak dapat menggunakan variabel tersebut secara langsung. Kita akan membahas tentang kemampuan akses pada pembahasan selanjutnya.

✓ *Class Variable atau StaticVariables*

Kita dapat mendeklarasikan *class variable* atau variabel yang dimiliki *class* sepenuhnya. Nilai pada variabel ini sama pada semua *object* di *class* yang sama. Anggaplah kita menginginkan jumlah dari mahasiswa yang dimiliki dari seluruh *class*, kita dapat mendeklarasikan satu *staticvariable* yang akan menampung nilai tersebut. Kita beri nama variabel tersebut dengan nama studentCount.

Berikut penulisan *staticvariable:*

Sintaks Class Variable

```
Public class StudentRecord
{  //area deklarasi instance variables
Private static int student Count;
//area penulisan kode selanjutnya
}
```

Kita gunakan *keyword : 'static'* untuk mendeklarasikan bahwa variabel tersebut adalah *static*. Maka keseluruhan kode yang dibuat terlihat sebagai berikut:

```
Public class StudentRecord
{
  private String name;
  private int age;
  Private double  average;
  //area penulisan kode selanjutnya
}
```

PRAKTIKUM

Latihan Class : Membuat class kotak (Simpan dengan nama **class**)

```
1   package pkgclass;
2
3   class Kotak{ //buat sendiri
4   double panjang;
5     double lebar;
6     double tinggi;
7   }
8
9   public class Class {
10      public static void main(String[] args) {
11          // TODO code application logic here
12      double volume1, volume2;
13
14      Kotak k1 = new Kotak();  // mendeklarasikan objek k1
15      Kotak k2 = new Kotak();  // mendeklarasikan objek k2
16
17      // Mengisikan nilai ke dalam objek k1
18      k1.panjang = 4;
19      k1.lebar = 3;
20      k1.tinggi = 2;
21
22      // Mengisikan nilai ke dalam objek k2
23      k2.panjang = 6;
24      k2.lebar = 5;
25      k2.tinggi = 4;
26
27          // Menghitung isi/volume dari objek k1
28      volume1 = k1.panjang * k1.tinggi * k1.lebar;
29
30     // Menghitung isi/volume dari objek k2
31     volume2 = k2.panjang * k2.tinggi * k2.lebar;
32
33     // Menampilkan nilai volume k1 dan k2 ke layar monitor
34     System.out.println("Volume k1 = " + volume1);
35     System.out.println("Volume k2 = " + volume2);
36   }
37     }
```

1) Bagaimana hasilnya ? jelaskan output yang dihasilkan!

……

……

……

Latihan 2. Mendefinisikan Method,

Ketikkan listing program di bawa ini dan simpan dengan nama **DemoMethod**

```
class Kotak {
  double panjang;
  double lebar;
  double tinggi;


  // Mendefinisikan method void (tidak mengembalikan nilai)
  void cetakVolume() {
    System.out.println("Volume kotak = " +   (panjang * lebar * tinggi));
  }
}

class DemoMethod {
  public static void main(String[] args) {
    Kotak k1, k2, k3;

    // instansiasi objek
    k1 = new Kotak();
    k2 = new Kotak();
    k3 = new Kotak();

    // mengisi data untuk objek k1
    k1.panjang = 4;
    k1.lebar = 3;
    k1.tinggi = 2;

    // mengisi data untuk objek k2
    k2.panjang = 6;
    k2.lebar = 5;
    k2.tinggi = 4;

    // mengisi data untuk objek k3
    k3.panjang = 8;
    k3.lebar = 7;
    k3.tinggi = 6;

    // memanggil method cetakVolume() untuk masing-masing objek
    k1.cetakVolume();
    k2.cetakVolume();
    k3.cetakVolume();
  }
```

Bagaimana hasilnya?

……
……

TUGAS

1. Buatlah listing program dengan kelas Mobil dan obyeknya Pajero. (capture listing program dan hasil)
2. Buatlah listing program untuk menampilkan behavior macan tutul (obyek) dari kelas binatang buas. (. (capture listing program dan hasil)

Tuliskan listing program pada lembar kerja berikut

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

8 Encapsulasi

A. TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

1. Memahami konsep enkapsulasi
2. Menerapkan konsep enkapsulasi dalam *class*

B. PERALATAN PRAKTIKUM

1. Personal Komputer
2. *Software Java Netbeans*

C. TEORI

1) Enkapsulasi dan *modifier*

Enkapsulasi merupakan teknik yang membuat variabel/*field class* menjadi bersifat *private* dan menyediakan akses ke variabel/*field* melalui *public method*. Jika *field* di deklarasikan sebagai *private*, maka *field* ini tidak bisa diakses oleh siapapun diluar *class*, dengan demikian *field* disembunyikan di dalam *class*.

Manfaat utama teknik enkapsulasi adalah kita mampu memodifikasi kode tanpa merusak kode yang telah digunakan pada *class* lain. Enkapsulasi memiliki manfaat sebagai berikut:

- ✓ Modularitas

  Source code dari sebuah *class* dapat dikelola secara independen dari *source code class* yang lain. Perubahan internal pada sebuah *class* tidak akan berpengaruh bagi *class* yang menggunakannya.
- ✓ *Information Hiding*

  Penyembunyian informasi yang tidak perlu diketahui objek lain.

jika Anda ingin beberapa atribut hanya dapat diubah hanya dengan *method* tertentu, tentu Anda ingin menyembunyikannya dari obyek lain pada *class*. Di Java, implementasi tersebut disebut dengan *access modifiers*.

2) Penerapan enkapsulasi dalam *class*

Kita dapat menyembunyikan information dari suatu *class* sehingga anggota-anggota

class tersebut tidak dapat diakses dari luar. Adapun caranya adalah cukup dengan memberikan akses *control private* ketika mendeklarasikan suatu atribut atau *method*. Contoh:

private int nip;

Encapsulation (Enkapsulasi) adalah suatu cara untuk menyembunyikan implementasi detail dari suatu class. Enkapsulasi mempunyai dua hal mendasar, yaitu:

- ✓ *information hiding*
- ✓ menyediakan suatu perantara (*method*) untuk pengaksesan data

Contoh:

```
public class Siswa {
private int nip;
public void setNip(int n) {
nip=n; } }
```

Contructor (konstruktor) adalah suatu *method* yang pertama kali dijalankan pada saat pembuatan suatu obyek. Konstruktor mempunyai ciri yaitu:

- ✓ mempunyai nama yang sama dengan nama class,
- ✓ tidak mempunyai return type (seperti void, int, double, dan lain-lain).

Contoh:

```
public class Siswa {
private int nrp; private String nama;

public Siswa(int n, String m) {
nrp=n;    nama=m;
}
}
```

Suatu *class* dapat mempunyai lebih dari 1 konstruktor dengan syarat daftar parameternya tidak boleh ada yang sama.

Contoh:

```
public class Siswa {
private int nrp;
private String nama;
public Siswa(String m) {
nrp=0;
nama="";
}
public Siswa(int n, String m) {
nrp=n;    nama=m;
}  }
```

Terdapat 4 macam *access modifiers* di JAVA, yaitu : *public*, *private*, *protected* dan *default*. 3 tipe akses pertama tertulis secara ekplisit pada kode untuk mengindikasikan tipe akses,

sedangkan yang keempat yang merupakan tipe default, tidak diperlukan penulisan *keyword* atas tipe.

No	Modifier	Class	Paket Sama		Paket Berbeda	
			Extend	Instan	Extend	Instan
1	Public	√	√	√	√	√
2	Protected	√	√	√	√	
3	Default	√	√	√		
4	Private	√				

PRAKTIKUM

✓ ***Public***

Dapat dilihat pada table diatas bahwa keyword Public dapat diakses didalam class itu sendiri, dapat diakses dengan menggunakan metode *extend* dan instan pada paket yang sama, serta dapat diakses dengan metode *extend* maupun instan dalam paket yang berbeda. Artinya hak akses *public* dapat diakses oleh sembarang object manapun dan dimanapun posisinya serta dengan apapun caranya. Data maupun *method* yang bersifat *public* dapat diakses oleh semua bagian didalam program. Untuk mendeklarasikan suatu data atau method dengan tingkat akses *public*, gunakan kata kunci *public*.

Berikut contoh program sederhana : buat file baru dengan nama **encapsulasi**

Listing Program Praktikum 1

```
1   package encapsulasi1;
2
3    class atas
4   {
5   public int a;
6   protected int b;
7   private int c;
8   }
9
10  public class Encapsulasi1 {
11
12  public static void main(String[] args) {
13          // TODO code application logic here
14      atas objek = new atas();
15      objek.a=2;
16      objek.b=3;
17
18  System.out.println("nilai a:" + objek.a);
19  System.out.println("nilai a:" + objek.b);
20
21      }
22  }
```

Bagaimana hasil dari listing program tersebut?

……
……

program diatas terdiri dari dua kelas yaitu kelas sekunder yang berisi variabel a, b dan c dengan tingkat akses yang berbeda, dan kelas primer yang berisi objek untuk melakukan *instance* pada kelas turunan, objek pada kelas primer hanya dapat mengisi nilai pada variabel a dan b karena kedua variabel tersebut memiliki tingkat akses *public* dan *protected*, karena variabel c memiliki tingkat akses *private* maka obyek pada kelas primer tidak bisa mengisi variabel tersebut.

- ✓ ***Protected***

  Suatu data maupun *method* yang dideklarasikan dengan tingkat akses *protected* dapat diakses oleh kelas yang memilkinya dan juga oleh kelas–kelas yang masih memiliki oleh hubungan turunan. Sebagai contoh, apabila data x dalam kelas A dideklarasikan sebagai *protected*, maka kelas B (yang merupakan turunan dari kelas A) diizinkan untuk mengakses data x. Namun apabila terdapat kelas lain, misalnya C (yang bukan merupakan turunan dari kelas A maupun B), tetap tidak dapat mengakses data – data yang dideklarasikan dengan tingkat akses *protected*. Untuk mendeklarasikan suatu data atau *method* dengan tingkat akses *protected*, gunakan kata kunci *ptotected*.

- ✓ ***Private***

  Dengan mendeklarasikan data dan method menggunakan tingkat akses private, maka data dan *method* tersebut hanya dapat diakses oleh kelas yang memilikinya saja. Ini berarti data dan *method* tersebut tidak boleh diakses atau digunakan oleh kelas-kelas lain yang terdapat didalam program. Untuk mendeklarasikan suatu data atau *method* dengan tingkat akses *private*, gunakan kata kunci *private*.

```
public class Siswa
{
  private String nama; //akses dasar terhadap variabel
private String getNama()    //akses dasar terhadap metode
{
        return name;
    }
}
```

Pada contoh diatas, variabel nama dan *method* getNama() hanya dapat diakses oleh *method internal class* tersebut.

Berikut contoh program sederhana : buat file baru dengan nama **encapsulasi**

Listing Program Praktikum 2

```
1   package encapsulasi1;
2
3    class atas
4   {
5   public int a;
6   protected int b;
7   private int c;
8   }
9
10  public class Encapsulasi1 {
11
12  public static void main(String[] args) {
13          // TODO code application logic here
14      atas objek = new atas();
15      objek.a=2;
16      objek.b=3;
17
18  System.out.println("nilai a:" + objek.a);
19  System.out.println("nilai a:" + objek.b);
20
21      }
22  }
```

Jalankan listing program tersebut, bagaimana hasilnya?

……

……

Pada baris ke 17, tambahkan script program untuk memanggil "c" **objek.c =5;**

Bagaimana hasilnya? Mengapa demikian? **Jelaskan!**

……

……

……

……

……

- ✓ ***Default***

Untuk hak akses *default*, sebenarnya hanya ditujukan untuk *class* yang ada dalam satu paket, atau istilahnya hak akses yang berlaku untuk satu folder saja (tidak berlaku untuk *class* yang tidak satu folder/package).

```
public class Siswa{
String nama;       //akses dasar terhadap variabel
String getName(){        //akses dasar terhadap method
   return nama;
  }   }
```

Pada contoh diatas, variabel nama dan *method* getNama() hanya dapat diakses oleh *method internal class* tersebut.

b. Praktikum 3

Modifikasi listing program pada praktikum 2, tambahkan variable alamat dengan type data String.

```
package encapsulasi1;

 class atas
{
public int a;
protected int b;
private int c;
private String alamat;

}
```

Tambahkan acessor method dan mutator method

```
package encapsulasi1;

 class atas
{
public int a;
protected int b;
private String alamat;

public String getAlamat(){
    return alamat;

}

public void setAlamat (String tempString){
alamat =tempString;
}
}
```

Panggil di program utama

```
public class Encapsulasi1 {
public static void main(String[] args) {
        // TODO code application logic here
    atas objek = new atas();
    objek.a=2;
    objek.b=3;
    objek.setAlamat("Malang");

System.out.println("nilai a:" + objek.a);
System.out.println("nilai a:" + objek.b);
System.out.println("Alamat:" + objek.getAlamat());

    }
}
```

Bagaimana cara mengakses atribut tersebut?

……
……
……

Apa fungsi acessor method dan mutator method?

……
……
……

Bagaimana hasilnya? Simpulkan!

……
……
……

TUGAS

1. Apa yang anda pahami terkait *keyword private*, *protected*, *public*?
2. Apa yang terjadi jika anda membuat sebuah *property* atau *method* menjadi *private*, *protected*, *public* ?

Buatlah program untuk menghitung gaji bersih dari seorang pegawai, pajak ppn sebesar 10% dari gaji kotor.

Tuliskan jawaban dan listing program pada lembar kerja berikut

……

9 INHERRITANCE (PEWARISAN)

TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

1. Memahami konsep pewarisan
2. Menciptakan superclass dan subclass
3. Memahami penggunaan kata kunci super
4. Menerapkan penggunaan kata kunci cuper dalam inheritas
5. Memahami konsep overloading dan overriding

PERALATAN PRAKTIKUM

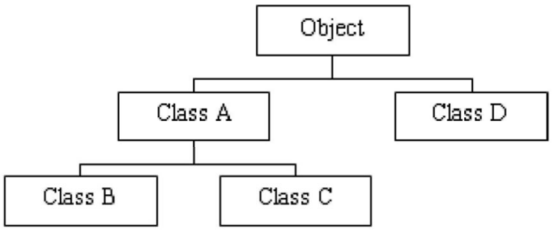
1. Personal Komputer
2. *Software Java Netbeans*

TEORI

Konsep Inheritas

Dengan *inheritance,* sebuah class dapat mempunyai class turunan.Suatu class yang mempunyai class turunan dinamakan parent class atau base class. Sedangkan class turunan itu sendiri seringkali disebut subclass atau child class. Suatu subclass dapat mewarisi apa-apa yang dipunyai oleh parent class. Karena suatu subclass dapat mewarisi apa apa yang dipunyai oleh parent class-nya, maka member dari suatu subclass adalah terdiri dari apa-apa yang ia punyai dan juga apa-apa yang ia warisi dari class parent-nya.

Dalam Java, semua class, termasuk class yang membangun Java API, adalah subclasses dari superclass Object. Contoh hirarki class diperlihatkan di bawah ini. Beberapa class di atas class utama dalam hirarki class dikenal sebagai superclass. Sementara beberapa class di bawah class pokok dalam hirarki class dikenal sebagai sub class dari class tersebut.



Class hierarchy in Java.

Pewarisan adalah keuntungan besar dalam pemrograman berbasis object karena suatu sifat atau method didefinisikan dalam ***superclass*****,** sifat ini secara otomatis diwariskan dari semua ***subclasses***. Jadi, Anda dapat menuliskan kode method hanya sekali dan mereka dapat digunakan oleh semua subclass. Subclass hanya perlu mengimplementasikan perbedaannya sendiri dan induknya.

Suatu class yang mempunyai class turunan dinamakan parent class atau base class. Sedangkan class turunan itu sendiri sering kali disebut subclass atau child class. Suatu subclass dapat Mewarisi siapa-apa yang dipunyai oleh parent class-nya, sehingga member dari suatu subclass adalah terdiri dari apa-apa yang ia punyai dan juga apa-apa yang diawarisi dari classparent-nya. Kesimpulannya, boleh dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (**extend**) parentclass-nya.

Mendefinisikan Superclass dan Subclass

Untuk memperoleh suatu class, kita menggunakan kata kunci **extend** yang digunakan untuk melakukan proses penurunan terhadap suatu kelas. Bentuk umum dari penggunaan kata kunci tersebut adalah sebagai berikut:

```
Class nama_subclass extends nama_superclass{
//badan class
}
```

- ✓ Kapan menerapkan inheritas?

  Kita baru perlu menerapkan inheritance pada saat kita jumpai ada suatu class yang dapat diperluas dari class lain.

  Misal terdapat class Pegawai public class Pegawai
  {public String nama;public double gaji;}

  Misal terdapat class Manager public class Manajer
  {public String nama;public double gaji;public String departemen;}

  Dari 2 buah class diatas, kita lihat class Manajer mempunyai data member yang identik sama dengan class Pegawai, hanya saja ada tambahan data member departemen.Sebenarnya yang terjadi disana adalah class Manajer merupakan perluasan dari class Pegawai dengan tambahan data member departemen.

  Disini perlu memakai konsep inheritance, sehingga class Manajer dapat kita tuliskan seperti berikut :

*public class Manajer **extends** Pegawai {public String departemen;}*

- ✓ Keuntungan inheritas
  - **Subclass** menyediakan state/behaviour yang spesifik yang membedakannya dengan **superclass**, hal ini akan memungkinkan programmer Java untuk menggunakan ulang source code dari superclass yang telah ada.
  - Programmer Java dapat mendefinisikan superclass khusus yang bersifat generik, yang disebut abstract class, untuk mendefinisikan class dengan behaviour dan state secara umum.
- ✓ Deklarasi inheritas

Di dalam Java untuk mendeklarasikan suatu class sebagai sub class dilakukan dengan cara menambahkan kata kunci **extends** setelah deklarasi nama class, kemudian diikuti dengan nama parentclass-nya. Kata kunci extends tersebut memberitahu compiler Java bahwa kita ingin melakukan perluasan class.

Berikut adalah contoh deklarasi inheritance:

```
public class B extends A{
.....
}
```

Contoh di atas memberitahukan compiler Java bahwa kita ingin meng-extend class A ke class B. Dengan kata lain, class B adalah subclass (class turunan) dari class A, sedangkan class A adalah parent class dari class B.

Java hanya memperkenankan adanya single inheritance. Konsep single inheritance hanya memperbolehkan suatu sublass mempunyai satu parent class. Dengan konsep single inheritance ini, masalah pewarisan akan dapat diamati dengan mudah.

Dalam konsep dasar inheritance dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parentclass-nya. Contoh:

```
Public class Pegawai{
Public String nama;
Public double gaji;
        }
Public class Manajer extends Pegawai{
Public String departemen;
        }
```

Pada saat class Manajer menurunkan atau memperluas (extend) class Pegawai, maka ia mewarisi data member yang dipunyai oleh class Pegawai. Dengan demikian, class

Manajer mempunyai data member yang diwarisi oleh Pegawai (nama, gaji), ditambah dengan data member yang ia punyai (departemen).

1. **Kata Kunci Super**

   Constructor yang terdapat pada kelas induk dapat dipanggil dari kelas turunannya menggunakan kata kunci ***super***. Bentuk umum pemanggilannya adalah :

   Super (daftar-parameter);

   Daftar-parameter adalah daftar parameter yang didefinisikan pada constructor kelas induk.

   Ada beberapa hal yang harus diingat ketika menggunakan pemanggil constuktor super:

   a. Pemanggil super() **harus dijadikan pernyataan pertama dalam** constructor.
   b. Pemanggil super() hanya dapat digunakan dalam definisi constructor.
   c. Termasuk constructor this() dan pemanggil super() **tidak boleh terjadi dalam** constructor **yang sama**.

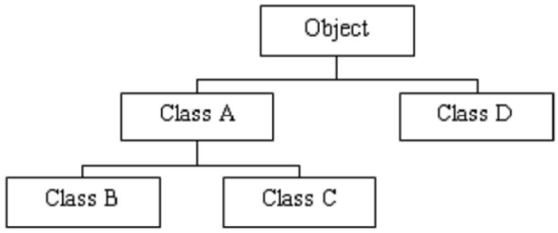
   Pemakaian lain dari super adalah untuk menunjuk anggota dari superclass (seperti reference **this**). Sebagai contoh,

```
public Student()
{
super.name = “somename”;
super.address = “some address”;
}
```

Untuk lebih memperjelas pembahasan, berikut contoh program yang menunjukkan penggunaan kata kunci *super* dalam pemanggilan *constructor* kelas induk

PRAKTIKUM

Praktikum 1. Perhatikuan ilustrasi berikut : class B turunan dari class A



```
Source  History
1  package turunan;
2  class A {
3    private int a;
4
5    public void setA(int nilai) {
6      a = nilai;
7    }
8
9    public int getA() {
10     return a;
11   }
12 }
13
14 class B extends A {  // membuat kelas turunan (subclass) dari kelas A
15   private int b;
16
17   public void setB(int nilai) {
18     b = nilai;
19   }
20
21   public int getB() {
22     return b;
23   }
24 }
25
26 public class Turunan {
27
28     /**
29      * @param args the command line arguments
30      */
31     public static void main(String[] args) {
32         // TODO code application logic here
33         B obj = new B();      // melakukan instansiasi terhadap kelas B
34
35     obj.setA(100);     // mengeset nilai objek dari kelas B
36     obj.setB(200);
37
38     // mendapatkan nilai yang terdapat dalam objek dari kelas B
39     System.out.println("Nilai a : " + obj.getA());
40     System.out.println("Nilai b : " + obj.getB());
41
42     }
43 }
```

Bagaimana hasilnya?

……

……

……

Lengkapi listing program dengan menambahkan class C merupakan turunan dari kelas A. tulis listing programnya!

……

……

……

……

.....

.....

.....

Praktikum 2. Class Balok turunan dari class persegipanjang

```
Source | History
 1  package pewarisan;
 2  class Persegipanjang {
 3    protected double panjang;
 4    protected double lebar;
 5
 6     Persegipanjang() {   //default constructor
 7      panjang = lebar = 0;
 8    }
 9
10    Persegipanjang(int p, int l) {
11      panjang = p;
12      lebar = l;
13       }
14
15    public double hitungLuas() {
16      return (panjang * lebar );
17    }
18  }
19
20  class Balok extends Persegipanjang {
21    private double tinggi;
22
23    Balok(int p, int l, int t) {
24
25      super(p, l);  // memanggil constructor kelas Kotak
26      tinggi = t;
27    }
28
29    public double getTinggi() {
30      return tinggi;
31    }
32  }
33
34  public class Pewarisan {
35      public static void main(String[] args) {
36          Balok k=new Balok(6,5,4);
37          System.out.println("Luas Persegi Panjang : " + k.hitungLuas());
38        System.out.println("tinggi balok  : " + k.getTinggi());
39        System.out.println("Volume balok : " + (k.hitungLuas()*k.getTinggi()));
40
41          // TODO code application logic here
42      }
43  }
```

Bagaimana hasilnya?

.....

.....

.....

.....

.....

2. konsep overloading dan overriding

1) Metode Overloading

Overloading adalah suatu keadaan dimana beberapa method sekaligus dapat mempunyai nama yang sama, akan tetapi mempunyai fungsionalitas yang berbeda. Overloading ini dapat terjadi pada class yang sama atau pada suatu parent class dan subclass-nya. Overloading mempunyai ciri-ciri sebagai berikut:

- ✓ Nama method harus sama
- ✓ Daftar parameter harus berbeda
- ✓ Return type boleh sama, juga boleh berbeda

Contoh penggunaan overloading dilihat di bawah ini:

```
Gambar(intt1)              →     1parametertitik,untuk menggambartitik
Gambar(intt1, int t2)      →     2parametertitik,untukmenggambargaris
Gambar(intt1, int t2, int  →     3parametertitik,untukmenggambarsegitiga
t3)
Gambar(intt1, int t2, int  →     3parametertitik,untukmenggambarsegiempat
t3, int t4)
```

2) Overriding Method

Overriding adalah suatu keadaan dimana method pada subclass menolak method pada parent class-nya. Overriding mempunyai ciri-ciri sebagai berikut:

- ✓ Nama method harus sama
- ✓ Daftar parameter harus sama
- ✓ Return type harus sama

Untuk beberapa pertimbangan, terkadang class asal perlu mempunyai implementasi berbeda dari method yang khusus dari *superclass* tersebut. Oleh karena itulah, method overriding digunakan. *Subclass* dapat mengesampingkan method yang didefinisikan dalam *superclass* dengan menyediakan implementasi baru dari method tersebut. Misalnya kita mempunyai implementasi berikut untuk method getName dalam superclass Person,

```
public class Person
{
public String getName()
{
System.out.println("Parent: getName");
return name;
}
}
```

Untuk override, method getName dalam subclass Student, kita tulis,

```
public class Student extends Person
{
public String getName()
{
System.out.println("Student: getName");
return name;
}
}
```

Jadi,ketika kita meminta method getName dari object class Student, method overridde akan dipanggil, keluarannya akan menjadi, *Student: getName*

3) Method final dan classfinal

Dalam Java, juga memungkinkan untuk mendeklarasikan class-class yang tidak lama menjadi subclass. Class ini dinamakan **class final**. Untuk mendeklarasikan class untuk menjadi final kita hanya menambahkan kata kunci **final** dalam deklarasi class. Sebagai contoh, jika kita ingin class Person untuk dideklarasikan final, kita tulis,

```
public final class Person
{
//area kode
}
```

Beberapa class dalam Java API dideklarasikan secara final untuk memastikan sifatnya tidak dapat di-*override*. Contoh-contoh dari class ini adalah Integer , Double , dan String. Ini memungkinkan dalam Java membuat method yang tidak dapat di-*override.* Method ini dapat kita panggil **method final**. Untuk mendeklarasikan method untuk menjadi final,kita tambahkan kata kunci final kedalam deklarasi method. Contohnya, jika kita ingin method getName dalam class Person untuk dideklarasikan final, kita tulis,

```
public final String getName(){
return name;
}
```

Method static juga secara otomatis final.Ini artinya Anda tidak dapat membuatnya override.

TUGAS

1. Buatlah program untuk menampilkan luas segitiga dengan superclass bangun_datar dan sub class Segitiga. Gunakan prinsip overriding dan atau overloading.
2. Buatlah program untuk menampilkan luas permukaan dan volume tabung. Gunakan parent-class Luas Lingkaran (method: jari-jari).

. Tuliskan listing program pada lembar kerja berikut

10 Polimorphism

TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

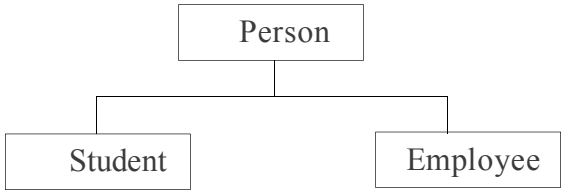
1. Memahami konsep polimorfisme
2. Menyajikan overloading dan overriding dalam class

PERALATAN PRAKTIKUM

1. Personal Komputer
2. *Software Java Netbeans*

TEORI

Class induk Person dan subclass Student dari contoh sebelumnya, kita tambahkan subclass lain dari Person yaitu Employee. Di bawah ini adalah hierarkinya,



Gambar Hirarki dari class induk Person

Dalam Java, kita dapat membuat referensi yang merupakan tipe dari superclass ke sebuah object dari subclass tersebut. Sebagai contohnya,

```
        public static main( String[] args )
        {
        Person      ref;
        Student     studentObject = new Student();
        Employee employeeObject = new Employee();

        ref = studentObject; //Person menunjuk kepada
        // object Student
//beberapa kode di sini
        }
```

Sekarang dimisalkan kita punya method getName dalam superclass Person kita, dan

kita override method ini dalam kedua subclasses Student dan Employee,

```
        public class Person {
public String getName(){
System.out.println("Person Name:" + name);
return name;
        }
        }

        public class Student extends Perso {
public String getName(){
System.out.println("Student Name:" + name); return name;
        }
        }

          public class Employee extends Person   {
              public String getName(){
              System.out.println("Employee Name:" + name);
 return name;
                    }
```

Kembali ke method utama kita, ketika kita mencoba memanggil method getName dari reference Person ref, method getName dari object Student akan dipanggil. Sekarang, jika kita berikan ref ke object Employee, method getName dari Employee akan dipanggil.

```
public static main( String[] args )
{
Person ref;
Student   studentObject = new Student();
Employee     employeeObject = new Employee();

ref = studentObject; //Person menunjuk kepada
String temp = ref.getName();       //getName dari Student
System.out.println( temp );                //class dipanggil

ref = employeeObject; //Person menunjuk kepada
// object Employee

String temp = ref.getName(); //getName dari Employee
System.out.println( temp );       //class dipanggil

}
```

Kemampuan dari reference untuk mengubah sifat menurut object apa yang dijadikan acuan dinamakan polimorfisme. Polimorfisme menyediakan multiobject dari subclasses yang berbeda untuk diperlakukan sebagai object dari superclass tunggal, secara otomatis menunjuk method yang tepat untuk menggunakannya ke *particular* object berdasar subclass yang termasuk di dalamnya.

Contoh lain yang menunjukkan properti polimorfisme adalah ketika kita mencoba melalui reference ke method. Misalkan kita punya method static **printInformation** yang mengakibatkan object Person sebagai reference, kita dapat me-reference dari tipe Employee dan tipe Student ke method ini selama itu masih subclass dari class Person.

```
public static main( String[] args )
{
Student     studentObject = new Student();
Employee    employeeObject = new Employee();

printInformation( studentObject );
printInformation( employeeObject );
}
public static printInformation( Person p ){
. . . .
}
```

PRAKTIKUM

```
class EkspresiWajah{
    public String respons(){
    return ("Lihatlah reaksi wajah saya");
        }
}

class Gembira extends EkspresiWajah{
   publuc String respons(){
   return ("Ha..ha..ha... saya sedang gembira");
   }
}

class Sedih extends EkspresiWajah{
    publuc String respons(){
    return ("Hiks..hiks..hiks... tega sekali kau..");
    }
}

class Marah extends EkspresiWajah{
    publuc String respons(){
    return ("Hei!... jangan dekati saya!");
    }
}
```

```
public class EkspresiEmosi {
    public static void main (String [] srgs){

    EkspresiWajah Oekspresi = new EkspresiWajah();
    Gembira Ogembira = new Gembira();
    Sedih Osedih = new Sedih();
    Marah Omarah = new Marah();

    EkspresiWajah [] ekspresi = new EkspresiWajah[4];
    ekspresi[0] = Oekspresi;
    ekspresi[1] = OGembira;
    ekspresi[2] = OSedih;
    ekspresi[3] = OMarah;

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

Bagaimana output dari program tersebut?

.....

.....

.....

.....

.....

.....

Jelaskan jalan programnya sesuai dengan konsep polimorfisme!

.....

.....

.....

.....

TUGAS

Buatlah program yang mengaplikasikan abstract class, interface, dan polimorphism

11 Exception Handling (Penanganan Eksepsi)

TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

1. Memahami konsep, tipe, dan cara penanganan eksepsi.
2. Memahami cara melontar dan menangkap eksepsi.
3. Memahami konsep try catch

PERALATAN PRAKTIKUM

1. Personal Komputer
2. *Software Java Netbeans*

TEORI

Eksepsi (exeption) adalah suatu mekanisme yang digunakan oleh beberapa bahasa pemrograman untuk mendeskripsikan apa yang harus dilakukan jika ada suatu kondisi yang tidak diinginkan terjadi. Eksepsi adalah keadaan tidak normal yang muncul pada suatu bagian program pada saat dijalankan. Penanganan eksepsi pada java membawa pengelolaan kesalahan program saat dijalankan kedalam orientasi-objek. Eksepsi java adalah objek yang menjelaskan suatu keadaan eksepsi yang muncul pada suatu bagian program.

Eksepsi dapat dijumpai saat:

a. Mengakses method dengan argumen yang tidak sesuai.
b. Membuka file yang tidak ada.
c. Koneksi jaringan yang terganggu.
d. Manipulasi operand yang nilainya keluar dari batasan yang didefinisikan.
e. Pemanggilan class yang tidak ada.

Eksepsi dapat muncul tidak beraturan dalam suatu method, atau dapat juga dibuat secara manual dan nantinya melaporkan sejumlah keadaan kesalahan ke method yang memanggil.

1. Dasar-dasar penanganan Eksepsi

Penanganan eksepsi pada java diatur dengan lima kata kunci : **try, Catch, throw,**

throws dan finally. Pada dasarnya try digunakan untuk mengeksekusi suatu bagian program, dan jika muncul kesalahan, sistem akan melakukan throw suatu eksepsi yang dapat anda catch berdasarkan tipe eksepsinya, atau yang anda berikan finally dengan penangan default.

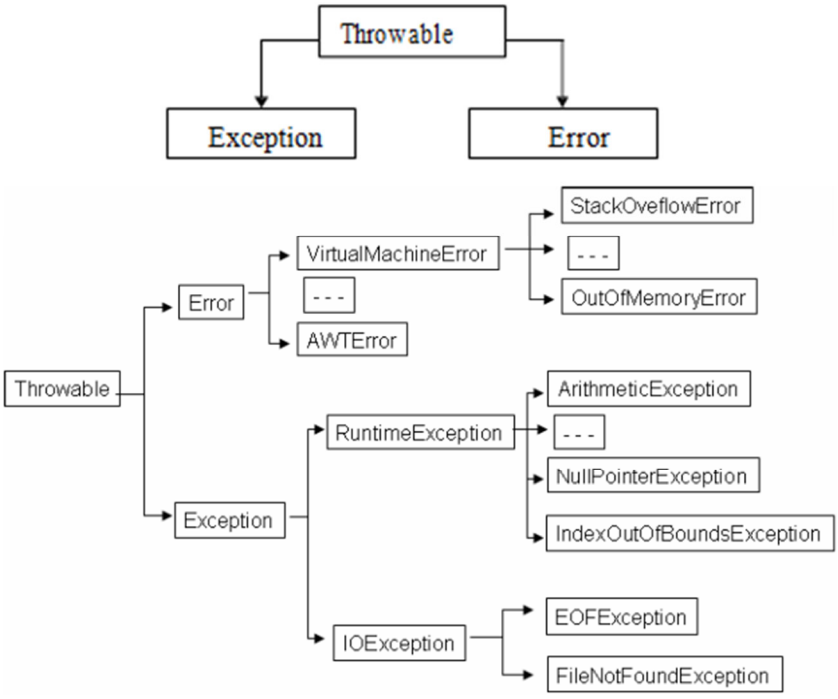
Berikut ini bentuk dasar bagian penanganan eksepsi :

```
try {
// Block of Code
}
catch (ExceptionType1 e) {     // Exception Handler for ExceptionType1
}
catch (ExceptionType2 e) {      // Exception Handler for
ExceptionType2 throw (e); // re-throw the Exception
}
finally { }
```

2. Tipe Eksepsi

Dipuncak hirarki class eksepsi terdapat satu class yang disebut throwable. Class ini digunakan untuk merepresentasikan semua keadaan eksepsi. Setiap *ExceptionType* pada bentuk umum diatas adalah subclass dari *throwable.*

Dua subclass langsung throwable didefinisikan untuk membagi class throwable menjadi dua cabang yang berbeda. Satu, class Exception, digunakan untuk keadaan eksepsi yang harus ditangkap oleh program yang kita buat. Cabang kedua throwable adalah class error, yang mendefinisikan keadaan yang tidak diharapkan untuk ditangkap dalam lingkungan normal.



Dalam hal ini class ***java.lang.Throwable*** merupakan class parent dari seluruh objek yang bisa dilempar dan ditangkap menggunakan mekanisme exception handling.

Sehingga dalam contoh di atas pernyataan ***catch (ArrayIndexOutOfBoundsException e)*** merupakan jenis *RuntimeException.* Jika tidak hafal jenis eksepsinya secara pasti, bisa mengambil parent classnya. Eksepsi Umum Java menyediakan beberapa eksepsi yang telah didefinisikan.

Beberapa eksepsi yang umum diantaranya adalah :

a. ***ArithmeticException*** hasil dari operasi divide-by-zero terhadap integer. Contoh : int i = 12 / 0;
b. ***NullPointerException*** mengakses membernya (atribut atau method) ketika reference-nya masih menunjuk ke null, misalnya ketika belum dibuat obyek instannya. Contoh : Date d = null; //tanpa membuat instance object System.out.println(d.toString());
c. ***NegativeArraySizeException*** membuat array dengan size yang diset negatif.
d. ***ArrayIndexOutOfBoundsException*** mengakses array melebihi indeks terbesarnya.
e. ***SecurityException*** biasanya terjadi pada sebuah browser, ketika class SecurityManager melempar eksepsi kepada applet yang melakukan operasi yang membahayakan host atau file-filenya (tidak berhak mengaksesnya), misalnya mengakses file sistem lokal, membuka soket ke sebuah host yang berbeda dengan host yang melayani applet, dan lain-lain.

3. Eksepsi Yang Tidak Dapat Ditangkap

Obyek eksepsi secara otomatis dihasilkan oleh runtime java untuk menanggapi suatu keadaan eksepsi. Perhatikan contoh berikut :

```
class Exc0 {
public static void main (String args[]) {
int d = 0;
int a = 42 / d;
} }
```

Saat runtime java mencoba meng-eksekusi pembagian, akan terlihat bahwa pembaginya adalah nol, dan akan membentuk objek eksepsi baru yang menyebabkan program terhenti dan harus berurusan dengan keadaan kesalahan tersebut. Kita belum mengkodekan suatu penangan eksepsi, sehingga penanganan eksepsi default akan segera dijalankan. Keluaran dari program diatas : java.lang.ArithmeticExpression : /by zero at Exc0.main (Exc0.java:4)

Berikut adalah contoh lainnya dari eksepsi :

```
class Exc1 {
static void subroutine() {
int d = 0;
int a = 42 / d;
}

public static void main (String args[]) {
Exc1.subroutine();
}
```

```
}
```

Output-nya : "Exception in thread main java.lang.ArithmeticException : / by zero at Exc1.subroutine(Exc1.java :4) at Exc1.main(Exc1.java : 7)

a. Try dan Catch

Kata kunci try digunakan untuk menentukan suatu blok program yang harus dijaga terhadap semua eksepsi, setelah blok try masukkan bagian catch, yang menentukan tipe eksepsi yang akan ditangkap. Perhatikan contoh berikut:

```
class Exc2 {
public static void main (String args[]) {
try {
int d = 0;
int a = 42 / d;
}
catch (ArithmeticException e) {
System.out.println( "Division By Zero" );
} } }
```

Outputnya:
C:\Documents and Settings \My Documents>java Exc2 Division By Zero

b. Throw

Pernyataan throw digunakan untuk secara eksplisit melemparkan suatu eksepsi. Pertama kita harus mendapatkan penanganan dalam suatu instance throwable, melalui suatu parameter kedalam bagian catch, atau dengan membuatnya menggunakan operator new.

Bentuk umum pernyataan throw :

throw eksepsi

throw ThrowableInstance;

eksepsi yang dimaksud harus berupa objek Throwable maupun objek dari kelas-kelas turunannya. Kita dapat melempar objek non-throwable, misalnya objek string. Contoh jika kita ingin membangkitkan eksepsi NullPointerException, maka kita dapat menuliskan kodenya sebagai berikut: *throw NullPointerException();*

Aliran eksekusi akan segera terhenti setelah pernyataan throw, dan pernyataan selanjutnya tidak akan dicapai. Blok try terdekat akan diperiksa untuk melihat jika telah memiliki bagian catch yang cocok dengan tipe instance Throwable. Jika tidak ditemukan yang cocok, maka pengaturan dipindahkan ke pernyataan tersebut. Jika tidak, maka blok pernyataan try selanjutnya diperiksa, begitu seterusnya sampai penanganan eksepsi terluar menghentikan program dan mencetak penelusuran semua tumpukan sampai pernyataan throw. Contoh :

```
class throwDemo {
static void demoProc() {
try {
throw new NullPointerException( demo ); }
catch (NullPointerException e) {
System.out.println( "caught inside demoproc" );
```

```
        throw e; }
        }
        public static void main (String args[]) {
        try {
        demoProc();
        }
        catch (NullPointerException e)      {
        System.out.println( "recaugt" :    + e);
        }  }
}
```

Output :
caught inside demoproc
recaught : java.lang.NullPointerException : demo

PRAKTIKUM

Praktikum 1. *ArrayIndexOutOfBoundsException*

C

.

```
public class Eksepsi {
    public static void main(String[] args) {
        // TODO code application logic here
        int[] A = new int[5];
    A[5] = 100;
    }
```

Bagaimana output dari program tersebut?

……

……

Jelaskan alasannya !

……

……

.Modifikasi program pada praktikum 1 menjadi script berikut: **Eksekusi dan analisa**

```
11  public class Eksepsi {
12      public static void main(String[] args) {
13          // TODO code application logic here
14          int[] A = new int[5];
15      try{
16          A[5] = 100;
17      }
18      catch (Exception e){
19          System.out.println ("Anda memasukkan index yang melewati batas");
20       }
21      }
22  }
```

Bagaimana hasilnya?

……

……

Mengapa demikian?

……

.....

.....

.....

Praktikum 2. Kata Kunci **Try dan catch**

```
1   package eksepsi2;
2
3   public class Eksepsi2 {
4       public static void main(String[] args) {
5          int pembilang = 2;
6       int penyebut = 0;
7       try {
8         int hasil = pembilang/penyebut; // menimbulkan eksepsi
9         System.out.println("Hasil = " + hasil); // tidak dieksekusi
10      } catch (ArithmeticException ae) {
11        System.out.println("KESALAHAN: " +
12          "Terdapat pembagian dengan nol");
13      }
14      System.out.println("Statemen setelah blok try-catch");
15    }
16      }
```

Tasks | Output - eksepsi2 (run) | Breakpoints | Variables

Bagaimana hasilnya?

.....

.....

Mengapa demikian, jelaskan hasil analisa anda!

.....

.....

.....

.....

Praktikum 3. Try dan Catch

```
6       public static void main(String[] args) {
7           int pembilang = 2;
8       int penyebut = 0;
9       try {
0         int hasil = pembilang/penyebut;          // SALAH
1         System.out.println("Hasil = " + hasil); // tidak dieksekusi
2       } catch (Exception e) {
3         System.out.println("KESALAHAN: " +
4           "Terdapat pembagian dengan nol");
5       }
6       System.out.println("Statemen setelah blok try-catch");
7     }
```

Bagaimana hasilnya?

.....

.....

Mengapa demikian, jelaskan hasil analisa anda!

……

……

……

Pada kasus tertentu (biasanya pada proses debugging) mungkin saja anda menginginkan pesan sebenarnya yang terkandung dalam eksepsi yang ditimbulkan. Untuk melakukan hal ini dapat menggunakan method **getMessage()** yang terdapat pada objek eksepsi yang dilewatkan.

Praktikum 4. Method **getMessage()**

```
16      public static void main(String[] args) {
17    int pembilang = 2;
18      int penyebut = 0;
19      try {
20        int hasil = pembilang/penyebut;          // SALAH
21        System.out.println("Hasil = " + hasil); // tidak dieksekusi
22      } catch (Exception e) {
23        System.out.println(e.getMessage());
24      }
25      System.out.println("Statemen setelah blok try-catch");
26      }
27  }
```

Bagaimana hasilnya?

……

……

Praktikum 5. Throw

```
1   package demothrow;
2
3   class Barang {
4     private String kode;
5     private String nama;
6     private double harga;
7
8     public void setKode(String vKode) {
9       try {
10        kode = vKode;
11        if (kode == null) {
12          throw new NullPointerException();
13        }
14      } catch (NullPointerException npe) {
15        System.out.println("KESALAHAN: " +
16          "Kode barang tidak boleh null");
17      }
18    }
19
20    public String getKode() {
21      return kode;
22    }
```

```
24    public void setNama(String vNama) {
25      try {
26        nama = vNama;
27        if (nama == null) {
28          throw new NullPointerException();
29        }
30      } catch (NullPointerException npe) {
31        System.out.println("KESALAHAN: " +
32          "Nama barang tidak boleh null");
33      }
34    }
35
36    public String getNama() {
37      return nama;
38    }
39
40    public void setHarga(int vHarga) {
41      harga = vHarga;
42    }
43
44    public double getHarga() {
45      return harga;
46    }
47  }
49  public class DemoThrow {
50      public static void main(String[] args) {
51         Barang obj = new Barang();
52
53      obj.setKode(null);
54      obj.setNama("Buku tulis");
55      obj.setHarga(2500);
56
57      System.out.println("\nKode  : " + obj.getKode());
58      System.out.println("Nama   : " + obj.getNama());
59      System.out.println("Harga  : " + obj.getHarga());
60      }
61  }
```

Bagaimana output program?

……

……

……

……

Mengapa demikian, jelaskan hasil analisa anda!

……

……

……

……

12 Input dan Output

TUJUAN PRAKTIKUM

Setelah praktikum ini, mahasiswa diharapkan dapat:

1. Memahami dasar-dasar I/O
2. Melakukan input
3. Menampilkan output
4. Kelas dan interface yang berkaitan dengan I/O

PERALATAN PRAKTIKUM

1. Personal Komputer
2. *Software Java Netbeans*

TEORI

Pengenalan Stream

Dalam Java, operasi I/O menggunakan *streams*.*Streams* adalah abstraksi dari sesuatu yang digunakan untuk menulis/menghasilkan dan membaca/mendapatkan suatu informasi. Semua streams memiliki sifat yang sama walaupun peralatan fisik yang berhubungan dengan suatu stream berbeda-beda. Secara umum streams dalam Java dibagi dalam 2 bagian besar :

a. *Byte streams, Byte Streams* digunakan untuk operasi I/O yang menggunakan data biner (byte)
b. *character streams, Character Stremas* digunakan untuk menangani operasi I/O yang menggunakan character.

Karakter dalam java menggunakan Unicode, sehingga penggunaan *character streams* dapat digunakan untuk menangani karakter-karakter internasional (karakter diluar kode ASCII Standar). Semua class & interface yang berhubungan dengan streams ada dalam package **java.io.**

Byte Stream

Java menyediakan dua class abstrak yang merupakan superclass tertinggi untuk Byte Stream, yaitu :

1) *InputStream* untuk membaca input
2) *OutputStream* untuk menuliskan output

Character Stream

Untuk character streams, Java menyediakan dua class abstrak yang merupakan superclass tertinggi yaitu :

1) *Reader* untuk membaca input
2) *Writer* untuk menuliskan output

Variable Stream Standar

Secara default, Java telah menyediakan 3 buah variabel streams yang dapat langsung digunakan, karena variabel ini member **public static** dari class **System**, yaitu : **in,out,err**.

1) **System.out** : output stream standar. Secara default outputnya adalah console.
2) **System.in** : input stream standar. Secara default inputya adalah keyboard.
3) **System.err** : output stream untuk mencetak pesan kesalahan pada console (default).

Membaca Input dari Console menggunakan Byte Streams

1) Untuk membaca dari console digunakan variabel stream standar yang telah disediakan oleh class **System**, yaitu **in**.
2) Variabel ini memegang referensi dari objek dengan tipe **InputStream** sehingga untuk membaca dari console (yang diketik lewat keyboard), dapat menggunakan method **read**

PRAKTIKUM

Praktikum 1. Membaca Input dari Console menggunakan Byte Streams (karakter)

```
1   package demostream;
2   import java.io.*;
3
4   public class Demostream {
5
6       public static void main(String[] args) throws IOException{
7           System.out.print("Masukkan sembarang karakter: ");
8
9       char ch;
10
11      InputStreamReader isr = new InputStreamReader(System.in);
12      BufferedReader br = new BufferedReader(isr);
13
14      ch = (char) br.read();
15
16      System.out.println("Karakter yang dimasukkan adalah \'" + ch + "\'");
17      }
18  }
```

Tasks | Output - demostream (run) | Breakpoints | Variables

Praktikum 2. Membaca Input dari Console menggunakan Data String

Untuk melakukan input berupa string digunakan method *readLine(), bukan read()*

Deklarasi dari method : *String readLine() throws IOException*

```
1  package demoinput;
2  import java.io.*;
3
4  public class Demoinput {
5
6      public static void main(String[] args) throws IOException{
7      System.out.print("Masukkan nama Anda: ");
8
9      String nama;
10
11     InputStreamReader isr = new InputStreamReader(System.in);
12     BufferedReader br = new BufferedReader(isr);
13
14     nama = br.readLine();
15
16     System.out.println("Halo " + nama + ", sudahkah Anda mengerti Java?");
17     }
18 }
```

Praktikum 4. Membaca Input dari Console menggunakan Byte Streams

```
1.  import java.io.*;
2.  public class DemoStream2
3.  {
4.  public static void main(String[] args) {
5.  byte[] data = new byte[10];
6.  int panjang=0;
7.  System.out.print("Masukkan data    : ");
8.  try {
9.  panjang=System.in.read(data);
10. } catch (IOException e) {
11. System.out.print("Terjadi Exception");
12. }
13. System.out.println("Yang anda ketik  : ");
14. for (int i=0;i<panjang;i++) {
15. System.out.print((char)data[i]);
16. }
17. System.out.println("Panjang Karakter : "+panjang);
18. }
19. }
```

Menulis Output ke Console menggunakan Byte Streams

Untuk menulis ke console digunakan variabel stream standar yang telah disediakan oleh class **System**, yaitu **out**. Variabel ini memegang referensi dari objek dengan tipe **PrintStream**. PrintStream merupakan turunan dari class OutputStream. Method yang biasa digunakan : **print(),println(),write().**

Praktikum 5. Menulis Output ke Console menggunakan Byte Streams

```
1.  import java.io.*;
2.  public class DemoStream3
3.  {
4.  public static void main(String[] args) {
5.  byte[] data = new byte[10];
6.  int panjang=0;
7.  System.out.print("Masukkan data    : ");
8.  try {
9.  panjang=System.in.read(data);
10.System.out.print("Yang anda ketik  : ");
11.System.out.write(data);
12.System.out.println("Panjang Karakter : "+panjang);
13.System.out.print("index ke-1 sebnyk 3 : ");
14.System.out.write(data,1,3);
15.} catch (IOException e) {
16.System.out.print("Terjadi Exception");
17.}
18.
19.}
20.}
```

Praktikum 6. Menampilkan Output

Untuk menampilkan output ke layar console, dapat dilakukan dengan method print()maupun println(). Method yang digunakan untuk melakukan proses ini adalah write().

```
1   package demowrite;
2   public class Demowrite {
3
4       public static void main(String[] args) {
5           int i = 'A';
6       byte b = 65;
7       char c = 'B';
8
9       System.out.write(i);
10      System.out.write(b);
11      System.out.write(c);
12      }
13  }
```

Tasks | Output - demowrite (run) | Breakpoints | Variables

```
run:
BUILD SUCCESSFUL (total time: 2 seconds)
```

Membaca Input dari Console menggunakan Character Streams.

Deklarasi konstruktor nya : **InputStreamReader (Input Stream in)**

Membaca Input dari Console menggunakan Character Streams

byte streams (8 bit) ➔ ukurannya lebih kecil dari char (16bit). Bagaimana mengkonversi dari byte streams menjadi char streams dengan benar?

Solusi : input stream sebaiknya dibaca dari buffer, bukan dari peralatan fisik langsung. Untuk bungkus objek dari **InputStreamReader** ke dalam class **BufferedReader.**

Deklarasi konstruktornya : **BufferedReader (Reader in)**

InputStreamReader input = new InputStreamReader(System.in);

BufferedReader buff = new BufferedReader(input);

Atau dengan cara :

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

Untuk membaca character streams, dapat menggunakan method :

1) int read() throws IOException
2) int read(char[] cbuf)throws IOException
3) int read(char[] cbuf,int off,int len) throws IOException
4) String readLine() throws IOException

Praktikum 7.

```
1.  import java.io.*;
2.  public class DemoStream6 {
3.  public static void main(String[] args) throws IOException {
4.  char data;
5.  String str="";
6.  BufferedReader buff =
7.  new BufferedReader(new InputStreamReader(System.in));
8.  System.out.println("Ketik  : ");
9.  data = (char) buff.read();
10. while (data!='\r') {
11. str+=data;
12. data = (char) buff.read();
13. }
14. System.out.println("Yang diketik : "+str);
15. System.out.println("Program Selesai");
16. }
17. }
```

TUGAS

Buatlah program kalkulator sederhana dengan menggunakan input dan output dari keyboard

. Tuliskan listing program pada lembar kerja berikut :

DAFTAR RUJUKAN

Raharjo, Budi dkk. 2012. *Mudah Belajar Java*. Bandung: Informatika

Graham, I. 1991. *Object Oriented Methods.* New York: Addison Wesley Inc.

Subiyantoro, Eko. 2013. *Pemrograman Berorientasi Object. Kementerian Pendidikan & Kebudayaan*

Sun Java Course. 2004. *Java Fundamental Programming*.

Sutopo, Aristo Hadi. 2005. *Pemrograman Berorientasi Objek dengan java*. Graha Ilmu

Sukamto, Rosa A., & Shalahuddin, M. 2010. *Modul Pembelajaran: Pemrograman Berorientasi Objek*. Bandung: Modul

ISBN 978-602-5914-58-4