PESANTREN PETIK

Jl. Mandor Basar No. 54 Rangkapanjaya
Pancoran Mas - Depok

Creates Future Skilled Professionals

K-103

PEMROGRAMAN JAVA TINGKAT DASAR

{ Pembuatan aplikasi berbasis dekstop menggunakan Bahasa Java }

SEMESTER 1
2019/2020

Pemrograman Java Tingkat Dasar
Kode Buku: JD.K-103
Revisi ke- 0
Tanggal: 01 Agustus 2019
Penulis: Prana Yanuar Dana, S.Kom.
Editor: Drs. Rusmanto, M.M.
Layout: Nanang Kuswana, S.Kom.

KATA PENGANTAR

Segala puji bagi Allah SWT, karena dengan rahmat dan karunia-Nya penulis mampu menyelesaikan modul Basis Data. Modul belajar ini disusun dengan tujuan untuk memperkenalkan dan memberikan pengertian pada sistem basis data untuk keperluan dalam dunia teknologi informasi. Dalam modul belajar ini juga penulis berusaha memberikan pemaparan materi yang mudah dipahami, sederhana, dan mudah dipraktikkan.

Penulis menyadari bahwa dalam penyusunan modul ini masih banyak kekurangan. Untuk itu kritik dan saran yang membangun sangat penulis harapkan agar dalam pembuatan modul selanjutnya dapat lebih baik. Harapan penulis, semoga modul ini dapat bermanfaat untuk penulis pada khususnya dan rekan-rekan pada umumnya.

Akhir kata, penulis ingin mengucapkan terima kasih kepada semua pihak yang telah membantu terbitnya modul ini, semoga bermanfaat untuk kita semua.

Depok, 09 Juli 2018

Penulis

DAFTAR ISI

BAB I
PENGENALAN JAVA

1.1. Tujuan

Setelah mengikuti modul ini, siswa diharapkan :

- Mengenal bahasa pemrograman Java
- Mengetahui konsep OOP
- Membuat program dan menjalankan program Java

1.2. Apa itu Java ?

Java adalah salah satu bahasa pemrograman ber-orientasi objek (OOP – Object Oriented Programming). Paradigma OOP menyelesaikan masalah dengan merepresentasikan masalah ke model objek.

Keutamaan Java dibanding bahasa pemrograman lain :

1. *Cross Platform*, dengan adanya Java Virtual Machine
2. *Robust* & *Secure* (tangguh dan aman)
3. Pengembangannya didukung oleh programmer secara luas
4. *Automatic garbage collection*, membebaskan programmer dari tugas manajemen memori

1.3. *Object Oriented Programming (OOP)*

Objek-objek dalam dunia nyata, mempunyai 2 karakteristik khusus / khas, yaitu : Status dan Perilaku. Contohnya, sepeda punya status (jumlah gir, jumlah pedal, dua buah ban, gir yang sedang dipakai-terkadang sebuah sepeda memiliki lebih dari satu gir) dan Perilaku (mengerem, melaju, mengubah gir, menabrak).

Bahasa yang berorientasi pada objek pun mempunyai karakteristik yang sama dengan objek-objek di dunia nyata, yaitu **Status** atau property objek yang dalam bahasa pemrograman disimpan sebagai **Variabel**, dan **Perilaku** yang diimplementasikan sebagai **Method**.

1.3.1. Karakteristik OOP

1. Enkapsulasi
2. Inheritansi
3. Polimorfisme

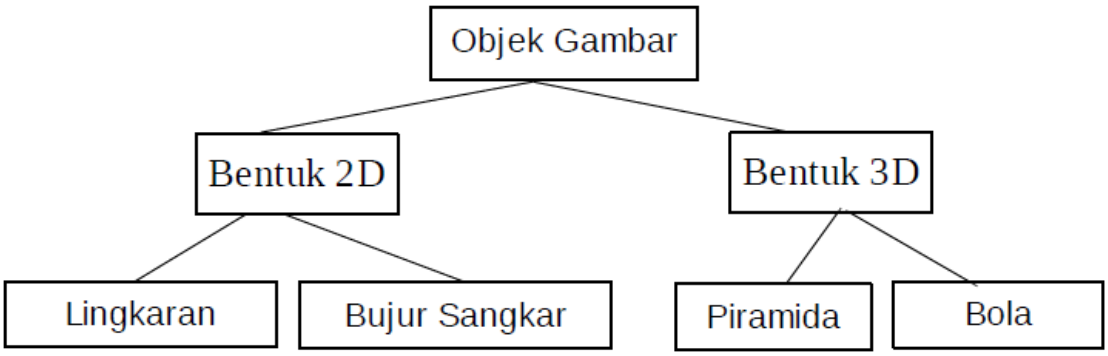
1. Enkapsulasi (pembungkusan)

Enkapsulasi adalah pelindung program dan data yang sedang diolah. Enkapsulasi mendefinisikan perilaku dan melindungi program dan data agar tidak diakses secara sembarangan oleh program lain.

Dalam java, dasar enkapsulasi adalah class. Anda membuat suatu class yang menyatakan bahwa variable atau method sebuah class tidak dapat diakses oleh class lain dengan menjadikan class tersebut *private*, atau menjadikan class tersebut *protected* – yaitu hanya bisa diakses oleh turunannya, atau menjadikan class tersebut *public* – yaitu bisa diakses oleh sembarang class.

2. Inheritansi (pewarisan)

Objek-objek yang berada di sekitar kita adalah objek-objek yang saling terhubung secara hierarkis. Misalnya :



gambar 1. Pewarisan pada object gambar

Lingkaran dan Bujur Sangkar adalah turunan dari bentuk 2D dan Bentuk 2D adalah turunan dari Objek Gambar.

Lingkaran dan Bujur Sangkar mewarisi (inherit) sifat-sifat dari Bentuk 2D, juga mewarisi sifat-sifat dari Objek Gambar.

Lingkaran dan Bujur Sangkar dapat dikatakan subclass dari Bentuk 2D. Bentuk 3D adalah parent-class dari Bola dan piramida, dan seterusnya.

3. Polimorfisme

Walaupun Lingkaran dan Bujur Sangkar sama-sama turunan dari Bentuk 2D, tetapi cara mengubah ukuran masing - masing berbeda, untuk lingkaran anda harus merubah besar jari-jarinya, sedang untuk bujur sangkar anda harus mengubah panjang sisinya.

Dalam java implementasi, method suatu parent-class dapat diubah oleh sub-classnya, hal ini dikenal dengan overriding method. Deklarasi Method sama tapi implementasi atau difinisinya berbeda (Method / perilaku yang sama tapi implementasinya/caranya yang berbeda-beda inilah yang disebut dengan Polimorfisme).

1.4 Program pertama

Format penulisan sintaks program Java adalah sebagai berikut :

```
class <nama_class>
{
      public static void main (String args[])
      {
            statement;
            ___________;
            ___________;
      }
}
```

Berikut contoh membuat program yang menampilkan kalimat : Selamat Belajar Java

```
1 //Nama File Salam.java
2 class Salam
3 {
4    public static void main (String args[])
5    {
6          System.out.println("Selamat Belajar Java");
7    }
8 }
```

Langkah selanjutnya

1. Simpan dengan nama: Salam.java, Nama file harus sama dengan nama classnya
2. Compile Salam.java: javac Salam.java
3. Hasilnya akan menghasilkan: Salam.class
4. Jalankan Salam.class: java Salam.class atau java Salam
5. Akan menghasilkan output:

```
Selamat Belajar Java
```

1.5 Aplikasi java vs applet java

Aplikasi java adalah program java yang dijalankan pada shell, sedangkan applet java adalah program java yang dijalankan dalam browser. Tampilan applet adalah tampilan grafis.

Contoh applet java (SalamApplet.java) :

```
1 import javax.swing.JApplet;
2 import java.awt.Graphics;
3 import java.awt.Color;
4
5 public class SalamApplet extends JApplet {
6    public void paint(Graphics g) {
7    g.drawRect(0, 0,
8    getSize().width 1,
9    getSize().height 1);
10   setBackground( Color.yellow );
11   g.drawString("Selamat Belajar Java!", 5, 20);
12   }
13
14 }
```

Langkah selanjutnya:

1. Compile SalamApplet.java
2. Menghasilkan SalamApplet.class
3. Jalankan di browser dengan memanggil salam.html di address bar-nya.

Cara menjalankan di browser:

1. Buat file salam.html

```
1 <html>
2 <head>
3 <title>aplikasi salam applet</title>
4 </head>
5
6 <body>
7  <applet  code="SalamApplet.class"  width="20"  height="40">
</applet>
8
9 </body>
</html>
```

2. Panggil Applet.html pada address bar browser

BAB II
Variable dan Operator

2.1 Tujuan

Setelah mengikuti Modul ini siswa diharapkan mengenal:

- Statement dan Identifier
- Java Keywords
- Variable dan Tipe Data
- Komentar
- Literal
- Operator dan Ekspresi
- Operator Aritmatic (*Arithmetic Operators*)
- Operator Pembanding (*Comparisons Operators*)
- Operator Logika (*Logical Operators*)

2.2 Statement dan Identifier

2.2.1 Statement

Bentuk Statement atau pernyataan dalam satu program di Java adalah sebagai berikut:

```
int i = 1;
import java.util.Calendar;
System.out.println ("Selamat Datang " + teman +
      " di Pesantren TIK");

pegawai.tetap = true;

total = a + b + c + d + e;
      sama dengan
total = a + b + c +
      d + e;
```

setiap statement selalu diakhiri dengan titik koma (;).

Blok adalah 2 tanda kurung kurawal ({}) yang menyatukan statement

```
{
        x = x + 1;
        y = y * 3;
}
```

Java memperbolehkan spasi dalam jumlah berapa saja (spasi, tab, baris baru).

```
class Hello
{
     public static void main (String args[])
     {
          System.out.println ("Hello World !")
     }
}
```

bisa ditulis dalam bentuk seperti di bawah ini:

```
class Hello {  public static void main (String args[])
{    System.out.println ("Hello World !");   }    }
```

2.2.2 Identifier

Dalam Java, identifier adalah nama yang diberikan untuk variabel, class atau method. Identifier hanya boleh dimulai dengan huruf, underscore(_) atau tanda dollar ($).

Identifier adalah case sensitive (membedakan huruf besar/kecil) dan tak ada batas maksimum.

Ingat: Aturan penamaan Java adalah memperhatikan besar kecil huruf atau Case Sensitive

Contoh:

username
user_name
_sys_var1
$change

2.3 Java Keywords

Java keywords adalah kata-kata / nama yang mempunyai arti khusus yang dikenali oleh java dan tak boleh digunakan untuk nama variabel, method atau class yang akan kita buat.

abstract	do	implements	private	throw
boolean	double	import	protected	throws
break	else	instanceof	public	transient
byte	extends	int	return	true
case	false	interface	short	try
catch	final	long	static	void
char	finally	native	super	volatile
class	float	new	switch	while
continue	for	null	synchronized	
default	if	package	this	

2.4 Variabel dan Tipe Data

2.4.1 Definisi Variabel

Variabel adalah suatu item dari data yang diberi nama identifikasi (identifier), variabel dapat diartikan lokasi di dalam memori yang mana suatu nilai (value) dapat disimpan.

2.4.2 Tipe Data

Java membagi tipe data menjadi 2 bagian :

1. Tipe data primitive

KeyWord	Size	Range
Bilangan Integer		
byte	8 bits	-128 s/d 127
short	16 bits	-32768 s/d 32767
int	32 bits	-2.147.483.648 s/d 2.147.483.647
long	64 bits	-9223372036854775808 s/d 9223372036854775807
Bilangan Real (Desimal)		
float	32 bits	Single Precision
double	64 bits	Double Precision

Tipe Data Lain		
char	16 bits	Single Character
boolean	true / false	Nilai Boolean

Contoh cara pendeklarasian dan inisialisasi Tipe Data Primitif sebagai berikut:

```
char ch;              // deklarasi variabel

ch = "R";             // inisialisasi variable

float jari = 8;       // deklarasi dan inisialisasi variabel

int x, y, z;          // deklarasi tiga variable integer

boolean tetap = true;
```

2. Tipe data reference

Reference adalah pointer ke data atau penyimpanan alamat.

Terdapat 3 data reference, yaitu:

1. Array
2. Class
3. Interface

Mengenai tipe data reference akan diuraikan dalam bab selanjutnya.

2.5 Komentar

Ada tiga cara memberikan komentar:

Delimiters	Use
//	Used for commenting a single line.
/* ---------- */	Used for commenting a block of code.
/** ---------- */	Used for commenting a block of code. Used by the Javadoc tool for generating Java Documentation.

Berikut cara menyisipkan komentar pada program:

```
class Hello
{
      // Program Pertama

      /**
      * Method main akan dieksekusi
      * ketika class dijalankan
      */

      public static void main (String args[])
      {
            System.out.println ("Hello World !");
      }
}
```

2.6 Literal

Karakter literal adalah karakter yang di tulis diantara kutip satu: 'r', '#', '14' dan sebagainya. Karakter ini disimpan sebagai 16 bit Unicode Characters. Berikut daftar special kode yang mereprentasikan karakter-karakter yang tidak dapat diprint (nonprintable characters).

Escape	Meaning
\n	Newline
\t	Tab
\b	Backspace
\r	Carriege return
\f	Formfeed
\\	Backslash
\’	Single Quote
\’’	Double Quote
\ddd	Octal
\xdd	Hexadecimal
\udddd	Unicode Character

Contoh:

“dalam kalimat ini terdapat sebuah \t tab didalamnya”

“Java adalah \”OOP\” programming”

“Trade Mark dari Java \u2122”

Hasil output jika dicetak pada baris terakhir akan muncul:

Trade Mark dari JavaTM

2.7 Operator dan Ekspresi (*Expression*)

Ekspresi (Expression) : adalah statemen yang mengembalikan suatu nilai

Operator : suatu simbol yang biasanya digunakan untuk ekspresi

2.7.1 Operator Aritmatika

Operator	Meaning	Example
+	Addition	3 + 4
-	Substraction	9 – 5
*	Multiplication	5 * 5
/	Division	14 / 7
%	Modulus	20 % 7

Contoh:

```
1 //Nama File Aritmatika.java
2 class Aritmatika {
3    public static void main (String args[]) {
4
5          short x = 10;
6          int y = 4 ;
7          float a = 12.5f;
8          float b = 7f;
9
10         System.out.println(" x = " + x + ", y = " + y );
11         System.out.println("x + y = " + (x + y));
12         System.out.println("x - y = " + (x - y));
13         System.out.println("x / y = " + (x / y));
14         System.out.println("x % y = " + (x % y));
15
16         System.out.println(" a = " + a + ", b = " + b);
17         System.out.println(" a / b = " + (a / b));
18
19   }
20 }
```

Lebih jauh dengan Assignment

Variabel assignment adalah suatu bentuk ekspresi .

```
x = y = z = 0;
```

pada contoh diatas variabel x, y, z bernilai 0.

Assignment Operator

Expression	Meaning
x += y	x = x + y
x -= y	x = x – y
x *= y	x = x * y
x /= y	x = x / y

Incrementing dan Decrementing

Sebagaimana di C/C++ , operator ++ dan -- dapat digunakan untuk Incrementing (penambahan) dan Decrementing (pengurangan) nilai 1.

x++ : penambahan variabel x dengan 1 (x = x +1)
x-- : pengurangan variabel y dengan 1 (y = y+1)

Namun tidak seperti C / C++ , Java mengizinkan x untuk bilangan floating point.

```
1 //Nama File : IncrementX.java
2 Class IncremenX {
3    public static void main (String args[]) {
4
5          int x = 0;
6          System.out.println ( " Nilai awal x : " + x);
7          x++;

8          System.out.println ( " Nilai x : " + x);
9          x++;
10         System.out.println ( " Nilai x : " + x);
11   }
12 }
```

2.7.2 Operator Pembanding

Java mempunyai beberapa ekspresi untuk menguji hasil suatu perbandingan.

Operator	Meaning	Example
==	Equal	X == 3
!=	Not Equal	X != 3
<	Less than	X < 3
>	Greater than	X > 3
<=	Less than or equal to	X <= 3
>=	Greather than or equal to	X >= 3

2.8 Operator Logika

Operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT
&	AND
|	OR
^	XOR
<<	Left shift
>>	Right shift
>>>	Zero fill right shift
~	complement
<<=	Left shift assignment (x = x <<y)
>>=	Right shift assignment (x = x >>y)
>>>=	Zero fill right shift assignment (x = x >>> y)
x&=y	AND assignment (x = x & y)
x|=y	OR assignment (x + x | y)
x^=y	XOR assignment (x = x ^ y)

BAB III
Array, Struktur Kondisi dan Looping

3.1 Tujuan :

Setelah mengikuti modul ini siswa diharapkan mengetahui:

- Cara penggunaan Array
- Cara penggunaan kondisional *if* dan *switch*
- Cara penggunaan perulangan *for* dan *while*

3.2 Array

Array adalah kelompok variabel dengan tipe sejenis dan dinyatakan dengan nama yang sama. Untuk membuat sebuah array di java, kita gunaka tiga step berikut ini :

1. Deklarasi sebuah variabel yang betipe array
2. Buat object array baru dan asign ke variabe array
3. Simpan nilai dalam array

3.2.1 Deklarasi Variabel Array

```
TipeData namaVariabel[];
```

atau

```
TipeData[] namaVariabel;
```

Contoh:

```
int bulan[];
Point sudut[];
```

atau

```
int[] bulan;
Point[] sudut;
```

3.2.2 Membuat Object Array

Ada dua cara untuk membuat object Array

1. Menggunakan keyword new

```
int[] nilai = new int[100];

String[] nama = new String[10];
```

Baris statement terakhir diatas membuat suatu object dengan tipe data String dengan 10 element (slot) didalamnya.

Ketika membuat sebuah array dengan menggunakan operator new, semua elemen diinisialisasi:

- Array numeric bernilai 0
- Array Boolean bernilai false
- Array character bernilai '\0'
- Array object bernilai null

2. Langsung inisialisasi dari element array

```
String[] strbulan = {"Januari", "Februari", "Maret",
"April" };
```

Setiap elemen dalam kurung kurawal{}, harus mempunyai tipe data yang sama.

3.2.3 Akses Element dari Array

Untuk akses element array dapat diperhatikan contoh berikut:

```
String[] bulan = new String[12];
bulan[1] = "Januari";    // memberi nilai slot 1 dari array
int jml = bulan.length;  // banyaknya slot dalam array
```

3.3 Array Multidimensi

Contoh multidimensi array:

```
int koordinat []  [] = new int[10] [10];
koordinat [0] [0] = 1;
koordinat [0] [1] = 2;
```

3.4 Kondisional if

3.4.1 Sintaks Statement if

```
if (ekspresi boolean)

{

     Satu element atau block statement;

}
```

Contoh:

```
1 if ( x < y )
2 {
3 System.out.println(" Nilai x : " + x +" Nilai y : " + y
     );
4 System.out.println(" x lebih kecil dari y");
5 }
```

3.4.2 Sintaks if dan else statement

```
if (Kondisi adalah benar)
{
      satu statement atau block statement;
}
else
{
      satu statement atau block statement;
}
```

Contoh:

```
1 if ( x < y )
2 {
3 System.out.println(" x lebih kecil dari y");
4 }
5 else
6 {
7 System.out.println(" x lebih besar atau sama dengan y ");
8 }
```

Contoh kondisional dengan tipe data boolean:

```
1 if (statusMesin == true)
2 {
3 System.out.println("Mesin sedang hidup ");
4 }
5 else
6 {
7 System.out.println("Mesin sedang Mati ");
8 }
```

3.5 Kondisional Switch

Kadang kala seorang programer dihadapkan pada kondisi untuk mengetes suatu nilai dengan banyak kondisi, dengan menggunakan operator kondisional if (nested if) memang bisa dilakukan namun terkadang terlalu rumit.

Contoh menggunakan nested if

```
1 if (operator == "+" )
2    addargs(arg1, arg2)
3 else if (operator == "-")
4    subargs(arg1,arg2);
5 else if (operator == "*")
6    multargs(arg1,arg2);
7 else if (operator == ":")
8    divargs(arg1,arg2);
```

Bentuk ringkas dari nested if adalah dengan menggunakan switch

3.5.1 Bentuk switch statement

```
switch (expr1)
{
     case expr2:
          statemen-statemen;
          break;
     case expr3:
          statemenstatemen;
          break;
     case expr4:
          statemenstatemen;
          break;
     default:
          statemenstatemen;
          break;
}
```

Contoh:

```
1 switch (operasi) {
2    case "+" ;
3         addargs(arg1,arg2);
4         break;
5    case "-";
6         subargs(arg1,arg2);
7         break;
8    case "*" ;
9         mulargs(arg1,arg2);
10        break;
11   case ":" ;
12        divargs(arg1,arg2);
13        break;
14 }
```

3.5.2 Contoh Kasus Switch

Pada contoh kasus ini anda diminta untuk input bilangan antara 1 s/d 5 , kemudian dihasilkan output nama bilangan yang diinput.

```
//Nama File : ContohSwitch
import java.util.Scanner;

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

          Scanner sc = new Scanner(System.in);
          System.out.print("Input Bilangan Antara 1 s/d 5 :
");

          int bil = sc.nextInt();
          String nama_bil = "";

          switch (bil) {
               case 1:
                    nama_bil = "Satu";
                    break;
               case 2:
                    nama_bil = "Dua";
                    break;
               case 3:
                    nama_bil = "Tiga";
                    break;
               case 4:
                    nama_bil = "Empat";
                    break;
               case 5:
                    nama_bil = "Lima";
                    break;
               default :
                    System.out.println(bil + " bukan antara
1 s/d 5 !!");

          System.exit(1);          // keluar

          }
     System.out.println(nama_bil);

     }
}
```

3.6 Looping for

Looping for digunakan untuk mengulang statement atau block statement beberapa kali sampai kondisi terpenuhi.

3.6.1 Bentuk sintaks for

for (ekspresi inisialisasi; ekspresi boolean; ekspresi)

```
{
      statement block statement;
}
```

Contoh:

```
1 for (int i = 1; i < 10; i++)
2 {
3 System.out.println("looping ke "+ i );
4 }
5 System.out.println("Selesai ....");
```

Contoh lain:

```
1 String[] nama2 = {"","Nanang", "Dudi", "Wahyu","Prana"};
2
3 for ( int i = 1 ; i < nama2.length; i++)
4 {
5 System.out.println( i + ". " + nama2[i]);
6 }
7 System.out.println("Nama-nama manajemen PeTIK");
```

3.7 Looping while

Looping while digunakan untuk mengulang statement atau block statement selama kondisi tertentu benar.

```
while (boolean
{
      Statement atau block statement;
}
```

Contoh:

```
1 int i = 0 ;
2 while (i < 10)
3 {
4 System.out.println("Looping ke ") ;
5 i++ ;
6 }
7 System.out.println("Selesai ...");
```

Bab IV
Mengenal Object dan Class

4.1 Tujuan

Setelah mengikuti modul ini anda diharapkan mengetahui :

- Cara mencipatakan Java Object
- Testing dan modifikasi class dan variabel instance
- Memanggil Method dan Objek
- Bagaimana bekerja dengan Objek
- Overview Java class libraries

4.2 Definisi Class dan Object

Dalam dunia nyata, kita sering berinteraksi dengan banyak object. Kita tinggal di rumah, rumah adalah suatu object, dalam terminologi Pemrograman Berorientasi Object (OOP) rumah kita adalah instance dari suatu class rumah. Misal kita tinggal dalam suatu komplek perumahan, sebelum membangun rumah developer akan berpanduan pada rancang bangun rumah *(blue print)* yang telah dibuat seorang arsitek.

Blue print dari rumah adalah class, sedang rumah yang kita tinggal (rumah-rumah dalam komplek) disebut instance atau object. Manusia adalah sebuah class; Anda, Saya, Kita adalah instance dari class Manusia. Contoh lain, adalah resep masakan yang ada di majalah kuliner atau buku resep, resep masakan adalah class, sedangkan makanan yang dimasak berdasarkan resep adalah object.

Object adalah instance dari class. Jika class secara umum merepresentasikan (template) suatu object, sebuah instance adalah representasi secara konkrit dari class itu sendiri.

4.3 Bekerja dengan Object

Ketika anda membuat program dengan Java, anda akan mendefinisikan beberapa class, anda juga akan menggunakan class untuk membuat suatu instance dan akan tentu saja akan bekerja dengan instance-instance tersebut.

4.3.1 Membuat Object

Untuk membuat object, kita gunakan keyword ***new*** dengan sebuah nama class yang akan dibuat sebagai instance dari class tersebut.

```
String str = new String();
Random r = new Random();
Pegawai p2 = new Pegawai();
```

Contoh :

Kita akan menggunakan class Date untuk membuat suatu object date. hari adalah object reference dari class Date yang akan digunakan untuk mengakses class Date. Sedangkan operator new adalah operator yang akan menghasilkan hari sebagai reference ke instance dari class Date().

```
import java.util.Date;

class CreateDates {

public static void main (String args[]) {
     Date d1, d2, d3;

     d1 = new Date();
     System.out.println("Hari 1 : " + d1);

     d2 = new Date(71, 4, 14, 8, 35);
     System.out.println("Hari 2 : " + d2);

     d3 = new Date("September 3 1976 2:25 PM");
     System.out.println("Hari 3 : " + d3);
     }
}
```

Ketika anda menggunakan operator new, sesungguhnya beberapa kejadian telah terjadi.

1. Instance baru yang telah diberikan oleh class diciptakan

2. Memori dialokasikan untuk instance tersebut.

3. Special method didefinisikan pada class (Konstruktor)

Konstruktor : Suatu method spesial yang digunakan inisialisasi sebuah instance baru dari class. Konstruktor menginisialisasi object-object baru dan variabel-variabel. Pemberian nama method Konstruktor harus sama dengan nama classnya (Penjelasan tentang kontruktor akan dibahas dalam bab berikutnya).

4.4 Akses dan Seting Class dan Variabel Instance

4.4.1 Akses variabel instance

Untuk mengambil value dari suatu variabel instance kita gunakan notasi titik (.). Dengan notasi titik, sebuah instance atau variabel class dibagi dua bagian. Object berada disebelah kiri titik, dan variabel berada di kanan titik.

```
Pegawai.status;
```

Pegawai adalah object, status adalah variabel. Misalkan status adalah object yang mempunyai variabel instance tersendiri yaitu tetap, penulisan dapat ditulis sebagai berikut:

```
Pegawai.status.tetap;
```

4.4.2 Menberi Nilai Variabel

Untuk memberi nilai suatu variabel kita gunakan operator “samadengan” (=) disebelah kanan ekspresi.

```
Pegawai.status.tetap = true;
```

Contoh :

```
//Nama File : TestPoint.java
import java.awt.Point;

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

          Point poin = new Point(10,10);
          System.out.println(" X = " + poin.x);
          System.out.println(" Y = " + poin.y);

          System.out.println(" Setting X = 6.");
          poin.x = 6;
          System.out.println(" Setting Y = 14.");
          poin.y = 14;

          System.out.println(" X = " + poin.x);
          System.out.println(" Y = " + poin.y);

     }
}
```

4.4.3 Variabel-variabel class

Variabel-variabel class adalah variabel-variabel yang didefinisikan dan tersimpan dalam class tersebut (lebih jelas akan di bahas dalam bab selanjutnya).

Contoh :

```
class Pegawai {
     static String nama_kantor = "PT NCS";
     String nama_panggilan;
     int umur;
     boolean status;
     . . .
}
```

Untuk mengakses variabel class sama seperti sebelumnya dengan menggunakan notasi titik (dot). Sintak berikut akan menghasilkan output yang sama.

```
Pegawai NCSStaff = new Pegawai();
Pegawai NCSSatpam = new Pegawai();
System.out.println("Nama Kantor : " + NCSStaff.nama_kantor);
System.out.println("Nama Kantor : " + NCSSatpam.nama_kantor);
```

4.4.4 Memanggil Method

Untuk memanggil method didalam object, sama seperti memanggil variabel instance; yaitu dengan menggunakan notasi titik (dot). Object berada disebelah kiri titik, dan method beserta argumen-argumen berada di kanan titik.

```
ObjectSatu.methodDua(arg1, arg2, arg3);
```

method tanpa argumen:

```
ObjectSatu.methodNoArg();
```

Jika method yang dipanggil mempunyai object yang mempunyai method tersendiri.

```
ObjectSatu.getClass().getName();
```

Method dengan kombinasi memanggil variabel instance

```
Pegawai.Golongan.getGaji(arg1, arg2)
```

Contoh :

```
//Nama File : TestString.java
class TestString {
     public static void main (String args[]) {
          String str = "Awalilah segala sesuatu pekerjaan
dengan Bismillah";

          System.out.println("Kalimat bijak : " + str);

          System.out.println("Panjang (Length) dari String : "
+ str.length());

          System.out.println("Character pada Posisi ke 4
adalah : " + str.charAt(4));

          System.out.println("SubString antara posisi 25
sampai 33 adalah : "+str.substring(25,33));

          System.out.println("index dari huruf B adalah ; " +
str.indexOf('B'));

          System.out.println("Tampilan kalimat bijak dengan
huruf besar : "+str.toUpperCase());
     }
}
```

4.5 Object Reference

Ketika bekerja dengan object-object, salah satu hal yang penting untuk dimengerti adalah bagaimana menggunakan reference ke suatu object. Ketika kita meng-asign suatu object ke variabel, atau menjadikan object-object sebagai argumen pada suatu method, sesungguhnya kita telah membuat reference ke object-object tersebut, bukan object atau duplikasi (copy) dari object yang membuat suatu reference.

Contoh berikut akan membuat kita jelas :

```
//Nama File : ReferencesTest.java
import java.awt.Point;

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

          Point poin1, poin2;
          poin1 = new Point(100,100);
          poin2 = poin1;

          poin1.x = 200;
          poin2.y = 200;

          System.out.println("Point 1 : " + poin1.x + ", " +
poin1.y);

     System.out.println("Point 2 : " + poin2.x + ", " +
poin2.y);

     }
}
```

Dalam program diatas , kita mendeklarasikan dua variabel bertipe Point, dan meng-asign suatu Point baru ke poin1. Kemudian meng-asign poin2 dengan nilai dari poin2.

Output yang terjadi adalah :

```
Point 1 : 200, 200
Point 2 : 200, 200
```

terlihat poin2 juga berubah. Ketika kita meng-asign suatu nilai dari poin1 ke poin2, sesungguhnya kita menciptakan sebuah reference dari poin2 menunjuk kesatu object yang sama dengan poin1.

4.6 Casting dan Converting Object-Object dan Tipe Data Primitif

Kadangkala kita perlu menyimpan suatu nilai dalam tipe data yang berbeda. Casting adalah mekanisme untuk menconvert (mengubah) tipe data dari suatu object atau data primitif

4.6.1 Casting Tipe Data Primitif

Casting antara tipe data primitif biasa dilakukan untuk tipe data numeric; tipe data boolean tidak dapat di casting ke tipe data primitif lain.

Perintah Casting : **`(TypeData) value`**
`TypeData : short, int, float,boolean`

Contoh :

```
float x = 9 ;
float y = 2;
int z = (int) x/y;
```

4.6.2 Casting Object

Casting dapat diartikan pula mengubah type / class suatu variabel. Variabel hanya dapat di Casting ke class lain yang merupakan super-class dari class lainnya. Instance dari Class (object) dapat di Casting menjadii instance dari class lain. Object hanya bisa di Casting menjadi instance dari super-classnya.

Perintah Casting : **`(nama class) object`**

Contoh :

```
Window win = new Window();
Container ctr = (Container) win;
```

4.6.3 Konversi Tipe Data Primitif ke Object dan Sebaliknya.

Sebelumnya kita telah mengetahui bagaimana mengcasting tipe data primitif ke tipe data primitif lainnya, dan dari satu class casting ke kelas lainnya. Bagaimana Casting tipe data primitif ke type reference (object), ...Oops Anda Tidak bisa melakukannya, karena dalam Java, Tipe Data Primitif dan Object adalah dua hal yang bebeda.

Namun demikian package java.lang telah menyediakan beberapa class yang berhubungan dengan tipe data primitif. Integer -> ints, Float -> floats, Boolean -> boolean dan sebagainya.

Contoh : Membuat object (instance) dari class Integer

```
Integer intObject = new Integer(14);
```

Kita gunakan method yang terdapat dalam class Integer, method intValue() untuk mendapatkan tipe data int (tipe data primitif) dari object Integer.

```
int num = intObject.intValue();
```

Lebih jauh lagi anda bisa pelajari dokumentasi Java API.

4.7 Overview Java class library

Berikut beberapa class dalam package bagian dari Java class library :

1. java.lang : Class-class yang mengaplikasikan penggunaan bahasa java, temasuk didalamnya object class, String class, dan System class. Terdapat pula spesial class dari Tipe data primitif (Integer, Character,Float dan sebagainya).
2. java.util : Berisi class-class utilitas, seperti Date, Calendar berisi pula collection class seperti Vector, List, ArrayList dan Hashtable
3. java.io : Berisi class-class untuk proses input dan menulis output (standard input dan standar output) juga berhubungan dengan menghandel file-file.
4. java.awt (abstract window toolkit) dan javax.swing ; class yang mengimplementasikan graphical user interface, dalam class ini terdapat class / komponen untuk membangun aplikasi grafis dengan Java
5. java.applet : class-class untuk implementasikan java applet.

Bab V
Class

5.1 Tujuan

Setelah anda mempelajari Bab ini anda diharapkan mengetahui :

- Memahami struktur dari suatu class
- Memahami dan mengenal Member dari Class Variabel dan Method
- Cara penggunaan Member Class

5.2 Mendeklarasikan Class

Class adalah kumpulan kode atau cetak biru (blue print) dari suatu object. Di dalam cetak biru menerangkan sipat dari object dan identitas suatu variabel.

Sintak untuk mendeklarasikan class :

```
[ 'public'] [( 'abstract' | 'final' ) ] 'class' nama_class
' { '
// statement / blok statement
// member class
'}'
```

Menggunakan keyword public pada class, berarti class tersebut bisa di akses class-class di seluruh package. Perlu diingat jangan menggunakan public jika class yang dibuat hanya di akses oleh class-class dalam satu package dengan class itu sendiri.

Gunakan keyword abstract untuk mengidentifikasi suatu class abstract dimana object tidak bisa diciptakan dalam class abstract. Gunakan keyword final untuk mengidentifikasi suatu class final, dimana tidak dapat diciptakan class baru turunan (inheretance) dari class final.

Penamaan class dengan ketentuan huruf character pertamanya adalah huruf besar.

Contoh:

```
class Account {
}
```

Jika class adalah sub class (turunan) dari class lain gunakan keyword extends.

```
Class AccountBank extends Account {
. . .
}
```

Dapat diartikan class AccountBank adalah turunan dari class Account. Jika class mengimplementasikan suatu interface :

```
class AccountBank implements Asset {
. . .
}
```

5.3 Member Class : Variable

5.3.1 Deklarasi Variable

Sintaks:

```
[ ( 'public' | 'private' | 'protected') ]
[ ( 'final' | 'volatile' ) ]
[ 'static' ] [ 'transient' ]
Tipe_Data Nama_Variabel [ '=' ekspresi ] '; '
```

Untuk nama variable biasakan gunakan huruf kecil .

Contoh :

```
class Account {
     String no_acc = "" ;
     double saldo = 0;
}
```

5.3.2 Akses dan Scope Variabel

Untuk akses dan skup variabel digunakan keyword: **public , private** atau **protected**.

1. Tanpa Keyword public, private atau protected

```
class MyClass
{
     int nama;
}
```

Jika tidak digunakan keyword public , private atau protected didepan tipe data dari variable

berarti : Hanya kode-kode yang terdapat dalam MyClass dan class-class lain yang dideklarasikan

dalam package dimana MyClass dideklarasikan yang hanya dapat mengakses variabel nama.

2. private

```
class Account
{
     private double saldo;
}
```

Berarti : Hanya kode-kode yang terdapat dalam class Account yang dapat mengakses variabel saldo.

3. Public

```
public class Pegawai
{
     public String nama;
}
```

Berarti : Kode-kode yang terdapat didalam class Pegawai dan class-class yang terdapat didalam package lain dapat mengakses variabel nama (class Pegawai harus dideklarasikan public juga agar dapat diakses class-class dalam package lain)

4. protected

```
public class Pegawai
{
     protected String nama;
}
```

Berarti : Hanya kode-kode yang terdapat dalam class Pegawai dan class-class lain dalam satu package yang sama dengan class Pegawai dan seluruh sub class dari class Pegawai (yang dideklarasikan dalam package lain) dapat mengakses variabel nama.

5.3.3 Table Akses Member Class Variable

Akses	Public	Protected	Private	Default
Dari Class Yang Sama	Ya	Ya	Ya	Ya
Dari Non Sub Class dalam package yang sama	Ya	Ya	Tidak	Ya
Dari Non Sub Class diluar package	Ya	Tidak	Tidak	Tidak
Dari Sub Class dalam package yang sama	Ya	Ya	Tidak	Ya
Dari Sub Class diluar package	Ya	Ya	Tidak	Tidak

5.3.4 Modifier

Pilihan Modifier menggunakan keyword : final atau volatile dan atau static dan atau transient.

Contoh:

```
class Pegawai
{
     final int AKUNTAN = 1;
     final int HRD_STAFF = 2;
     final int MANAGER = 3;
}
```

- Jika suatu variabel dideklarasikan dengan final, variabel dianggap sebagai variable konstan (konstanta) yang bersipat read only. Biasanya pemberian nama variabel dengan huruf besar.
- Variabel yang dideklarasikan dengan volatile akan dapat diakses oleh multiple thread.
- Keyword static digunakan untuk mendeklarasikan variabel, dimana seluruh object dapat mengambil (share) nilai variabel tersebut, jika variabel yang dideklarasikan dengan static diberi nilai baru maka seluruh object akan mengambil/melihat nilai terbaru.
- Variabel yang dideklarasikan dengan transient tidak akan disimpan selama object serialization.

5.3.5 Variabel Instance

Sebuah variabel instance adalah variabel yang dideklarasikan dengan tidak menggunakan keyword modifier static.

Contoh :

```
class Lingkaran {

int jari = 14;

public static void main (String args[]){
     Lingkaran lingkar1 = new Lingkaran();
     System.out.println ("JariJari : " + lingkar1.jari);
     }
}
```

5.3.6 Variabel Class

Variabel class adalah variabel yang dideklarasikan dengan menggunakan keyword modifier static. Variabel-variabel class berhubungan dengan class-class bukan object-object. Variabel class di ciptakan ketika class di load dan dihapus (destroyed) ketika keluar dari class (unload).

Contoh :

```
class Lingkaran {
     static float PHI = 3.14;

     public static void main (String args[]) {
          System.out.println ("NILAI PHI : " + PHI );
     }
}
```

Untuk mengakses variabel class dari suatu method dalam satu class lain, kita hanya perlu memanggil nama variabel dengan terlebih dahulu menyebut nama classnya, misal: Lingkaran.PHI

Contoh :

```
class Lingkaran {
     static float PHI = 3.14;
}

class UseLingkaran {
     public static void main (String args[]) {
     System.out.println ("NILAI PHI : " + Lingkaran.PHI
);
     }
}
```

5.3.7 Konstanta

Konstanta adalah variabel yang hanya bisa di baca (read only variabel) dan tidak bisa diubah nilainya. Untuk mendeklarasi variabel konstan gunakan keyword modifier final.

Contoh :

```
class Lingkaran {
     final static float PHI = 3.14;
}
```

5.4 Member Class : Method

5.4.1 Deklarasi Method

Java menggunakan “method” untuk mengacu fungsi atau prosedur dalam class. Jika variabel object digunakan untuk menyimpan status object. Maka method menggambarkan sifat-sifat (behaviours) dari object atau class.

Sintaks:

```
[ ('public' | 'private' | 'protected') ]
( [ 'abstract' ] | [ 'final' ] [ 'static'] [ 'native']
[ 'synchronized' ] )

NilaiKembalian_TipeData nama_method ' ( ' [
list_parameter ] ' )'
block_statement
```

5.4.2 Nama Method

Nama method akan mengidentifikasi suatu method, biasanya nama method menggunakan kata kerja, dalam bahasa lainnya method disebut juga function, subroutine atau procedure. Di Java kita bisa mempunyai beberapa method dengan nama yang sama tapi mempunyai tipe nilai balik yang berbeda atau argumen-argumen yang berbeda pula. Penamaan method jika satu kata gunakan huruf kecil untuk awal kata dan jika mengandung arti lebih dari satu kata (dua kata atau lebih) maka awal kata berikutnya gunakan huruf besar.

Contoh: **`hitung(), setNama(), getNamaLengkap()`**.

5.4.3 Nilai Kembalian (Return Value)

Nilai kembalian dapat berupa tipe data primitif , atau nama dari class ataupun tidak mempunya nilai kembalian dengan keyword void. Nilai Kembalian yang bertipe data primitif menggunakan keyword boolean, byte, char, double, float, int, long atau short.

Contoh:

```
public class Account {
     private String no_acc = "";
     private double saldo = 0;

     public String getNoAccount() {
          return this.no_acc;
     }

     public double getSaldo() {
          return this.saldo;
     }

     public void showInfo() {
          System.out.println("No Account: " + this.no_acc
+ ", Saldo" + this.no_acc;
     }
}
```

5.4.4 List parameter

List Parameter pada method adalah variabel-variabel yang di-passing sebagai masukan method yang dipisahkan oleh tanda coma (,)

Contoh :

```
// Nama File Kalkulasi.java
class Kalkulasi {

     int sum (int x, int y){
          int jumlah;
          jumlah = x + y ;
          return jumlah;
     }

     public static void main (String args[]) {

     Kalkulasi kal = new Kalkulasi();
     int hasil;

     System.out.println("Penjumlahan Menggunakan Method
sum : Bil 6 dan 7");

     hasil = kal.sum(6,7);
```

```
        System.out.println ( "Jumlah : " + hasil);
        }
}
```

5.4.5 Spesifikasi Acces Method

Deklarasi dari method menggunakan keyword : public , private atau protected.

1. Tanpa Deklarasi Method

Contoh :

```
class Temperature {
      private int derajat;
      void setDerajat(int d) {
            derajat = d;
      }

      int getDerajat()
      {
            return derajat;
      }
}
```

Dari program diatas hanya method-method pada class Temperature dan class-class lain dalam satu package dengan class Temperature hanya bisa mengakses method `setDerajat()` dan `getDerajat()`.

2. Deklarasi Method dengan private

Contoh :

```
class MathUtilitas {
      private int factorial (int n) {
            if ( n == 0)
            return 1; // 0! (0 factorial) nilainya 1

            int kali = 1;

            for (int i = 2; i <= n ; i++)
            kali *= i;

            return kali;
      }

      int permutasi (int n, int r) {
            return factorial (n) / factorial (n r);
      }
}
```

Hanya kode-kode dalam class MathUtilitas yang dapat mengakses method `factorial()`.

3. Deklarasi method dengan public

Jika kita mendeklarasikan suatu method dengan public mengakibatkan seluruh kode yang terdapat didalam class tersebut dan class-class dalam package lain dapat memanggil method tersebut.

Contoh :

```
public class Account {
      private String no_acc = "";
      private double saldo = 0;

      public String getNoAccount() {
            return this.no_acc;
      }
}
```

4. Deklarasi method dengan protected

Contoh :

```
class Pegawai {
      // kodekode lain tidak ditampilkan dalam program ini

      protected double hitungGaji() {
            return jam_kerja * gaji_perjam;
      }
}
```

Hanya kode-kode yang terdapat dalam class Pegawai, class-class lain yang dideklarasikan dalam satu package yang sama dengan class Pegawai dan kode-kode pada subclass Pegawai (yang dideklarasikan di package manapun) dapat mengakses method hitungGaji().

5.4.6 Modifier

Method dapat dideklarasikan dengan Modifier *abstract* atau *final* dan atau *static* dan atau *native* dan atau *synchronized.* Jika method dideklarasikan dengan abstract, maka method hanya terisi dengan nama-nama method saja, tidak ada block statement, Method dengan deklarasi abstract hanya terdapat pada class abstract. Method dengan deklarasi abstract tidak bisa dideklarasikan dengan final, static, native, synchronized atau private secara bersamaan.

Jika method dideklarasikan dengan final, merhod tersebut tidak bisa di override oleh subclass dari class tersebut. Jika method dideklarasikan dengan static, method tersebut dikenal sebagai sebuah class method karena method ini hanya bisa diakses oleh variabel-variabel class.

Method dengan deklarasi native adalah hanya berisi nama method dan argumenargumennya (method signature), dimana body dari method (kode-kode) ditulis dengan program C++ dan tersimpan dalam library C++. Sedangkan method dengan

deklarasi synchronized berhubungan dengan thread, Hanya satu thread pada satu saat yang dapat mengeksekusi method synchronized.

5.4.7 Instance Method

Sebuah instance method dapat mengakses variabel instance atau variabel class yang dideklarasikan di dalam class. Untuk mendeklarasikan instance method tidak menggunakan keyword static di dalam body dari instance method. Perhatikan contoh di bawah ini.

```
class Pegawai {
      private double gaji;

      void setGaji (double g)
      {
            this.gaji = g;
      }
}
```

setGaji() adalah sebuah instance method, **setGaji()** mengakses variabel instance gaji.

5.4.8 Penggunaan keyword this

Misalkan kita mempunyai sebuah instance method dengan parameter yang mempunya nama yang sama dengan variabel instance. Adalah dimungkinkan untuk meng-asign suatu nilai parameter ke variabel dengan nama yang sama.

Contoh :

```
/**
* Nama File : Account.java
*/
public class Account {
      private String no_acc = "";
      private double saldo = 0;

      public void setNoAccount(String no_acc) {
            this.no_acc = no_acc;
      }
}
```

Pada class Account terdapat sebuah variable class bernama no_acc. Pada method **setNoAccount()** dengan parameter argumen dengan nama yang sama dengan nama variable class, maka untuk membedakannya gunakan kata kunci this untuk akses variable class.

Keyword this digunakan untuk mengakses member class (variable atau method) dari dalam class, atau keyword this digunakan untuk mengidentifikasi object saat ini (current object).

1. Memanggil instance method dari instance method lain dalam class yang sama

Contoh :

```
public class Account {
	private String no_acc = "";
	private double saldo = 0;

	public void setNoAccount(String no_acc) {
		this.no_acc = no_acc;
	}


	public void deposit(double uang ) {
		this.saldo = this.saldo + uang;
	}

	public void withdraw( double uang ) {
		this.saldo = this.saldo uang;
	}


	public void transfer(Account ac_tujuan, double uang)
	{
		ac_tujuan.deposit( uang );
		this.withdraw( uang );
	}
}
```

Pada program diatas instance method **`transfer()`** memanggil instance method **`deposit()`** dan **`withdraw()`**.

1. Memanggil instance method dari class lain terlebih dahulu dibuat object dari class instance method yang akan dipangil.

Contoh:

```
/**
* UseAccount.java
**/
public class UseAccount {
     public static void main (String args[]) {
          Account ac = new Account();
          ac.setNoAccount("007");
          ac.deposit(400);
     }
}
```

Contoh program pada class UseAccount pada method main didefinisikan terlebih dahulu object instance dari classs Account bernama ac, baru kemudian memanggil method setNoAccount() dan deposit().

2. Memanggil instance method dengan teknik method chaining

Contoh :

```
// Nama File : HelloGoodBye

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

          HelloGoodBye gb = new HelloGoodBye();
          gb.hello().goodbye();
     }

     HelloGoodBye hello() {
          System.out.println("Hello .. ");
          return this;
     }


     void goodbye() {
          System.out.println("Goodbye ...");
     }
}
```

Pada program diatas method **`main()`** pada HelloGoodBye menciptakan object (gb) yang akan memanggil method **`hello()`**. Karena **`hello()`** mengembalikan suatu reference ke object yang sama (pada statement return this) sehingga method **`goodbye()`** dapat dipanggil setelah method **`hello()`**.

5.4.9 Class Method

Class method adalah suatu method yang dapat diakses oleh variabel yang dideklarasikan dalam class tersebut. Untuk mendeklarasikan sebuah class method menggunakan keyword static pada awal nama class method.

Contoh :

```
class Pegawai {
      private static int nomorPegawai;

      static void setNomorPegawai (int nopeg) {
            nomorPegawai = nopeg;
      }
}
```

Method setNomorPegawai meng-asign suatu parameter nilai nopeg ke variabel class nomorPegawai. Sebuah class method dapat dipanggil dari class method lain dalam class yang sama dengan menyebutkan nama methodnya dan argumen yang diberikan.

Contoh :

```
// Nama File : AnakPegawai.java
class AnakPegawai {
      static void namaPegawai(String nama, int
jumlah_anak) {
            System.out.println("Nama Ayah : " + nama);
            jumlahAnak(jumlah_anak);
      }


      static void jumlahAnak(int anak) {
            System.out.println("Jumlah anak = " + anak);
      }
}
```

Contoh diatas menyatakan method **`namaPegawai()`** memanggil class method **`jumlahAnak()`**. Untuk memanggil class method dari class lain akan diperlihatkan dalam contoh dibawah ini.

```
// Nama File : UseAnakPegawai.java

class UseAnakPegawai {
     public static void main (String args[]) {
          AnakPegawai.namaPegawai("Yan Farmawan",2);
     }
}
```

5.4.10 Overloading methods

Mendeklarasikan beberapa method dengan nama yang sama tetapi dengan parameter-parameter yang berbeda dikenal sebagai Overloading Method. Ketika method dipanggil, compiler akan memilih method yang sesuai dengan parameter-parameter yang diberikan.

```
/**
* Nama File : Hitung.java
*
**/

public class Hitung {
     public static int jumlah(String a, String b) {
          return Integer.parseInt( a ) +
Integer.parseInt( b );
     }

     public static int jumlah(int a, int b ) {
          return a + b ;
     }

     public static int jumlah(float a, float b) {
          return (int) a + (int) b;
     }
}
```

Contoh diatas memperlihatkan tiga method overloading, method jumlah mempunyai parameter-parameter dengan tipe data yang berbeda.

5.4.11 Overredding Method

Suatu method dalam subclass dapat mengganti (meng-overide) method yang ada di dalam superclassnya. (akan dijelaskan dalam bab berikutnya)

5.5 Konstruktor Class

Konstrukor adalah spesial method yang dipergunakan untuk menginisialisasi object. Nama Konstruktor harus sama dengan nama class dimana konstruktor dideklarasikan.

Konstruktor tidak boleh dideklarasikan dengan menggunakan tipe data kembalian (return data type).

5.5.1 Konstruktor Kosong

Jika kita membuat suatu class sebagai berikut :

```
class X {
}
```

Class X kelihatannya kosong, tetapi dalam sebenarnya compiler melihat class X dengan sebuah contruktur tanpa argumen-argumen, seperti sebagai berikut:

```
class X {
      X(){
      }
}
```

Jika class lain akan menggunakan object , terlebih dahulu class tersebut harus menciptakan suatu object sesuai dengan argumen yang diberikan konstruktornya.

```
class X {
      private int i;
      X (int i)
      {
            this.i = i;
      }
}


class UseX {

      public static void main (String args[]){
            X x1 = new X(2);
      X x2 = new X(); // akan terjadi error
      }
}
```

5.5.2 Konstruktor untuk inisialisasi variable class

Konstruktor dapat digunakan untuk inisialiasi awal member variabel class, berikut contoh class Account dimana argumen konstruktornya adalah nomor account dan saldo awal.

```
public class Account {
      private String no_acc = "";
      private double saldo = 0;

      public Account ( String no, double saldo) {
            this.no_acc = no;
            this.saldo = saldo;
      }

      public String toString() {
            return "No.Acc " + this.no_acc + ", Saldonya:"
+ this.saldo;

      }

      public static void main (String args[]) {
            Account ac1 = new Account("007",400);
            Account ac2 = new Account("010",256);
            System.out.println( ac1 );
            System.out.println( ac2 );
      }
}
```

5.5.3 Overloading Konstruktor

Sebagaimana method, konstruktor dapat mempunyai parameter-parameter yang berbeda. Sehingga dimungkinkan untuk membuat object dengan input argumen yang berbeda-beda pula.

Contoh :

```
//Nama File : Persegi.java
import java.awt.Point;

class Persegi {
     int x1 = 0;
     int y1 = 0;
     int x2 = 0;
     int y2 = 0;

     Persegi (int x1, int y1, int x2, int y2) {
          this.x1 = x1;
          this.y1 = y1;
          this.x2 = x2;
          this.y2 = y2;
     }

     Persegi (Point topLeft, Point bottomRight) {
          x1 = topLeft.x;
          y1 = topLeft.y;
          x2 = bottomRight.x;
          y2 = bottomRight.y;
     }

     Persegi (Point topLeft, int lebar, int tinggi) {
          x1 = topLeft.x;
          y1 = topLeft.y;
          x2 = (topLeft.x + lebar);
          y2 = (topLeft.y + tinggi);
     }

     void CetakPersegi() {
          System.out.print("Koordinat Persegi : < " + x1
+ ", " + y1);


     System.out.println(", " + x2 + ", " + y2 + " >");
     }
}
```

Berikut contoh program untuk memanggil class Persegi.

```
//Nama File PanggilPersegi.java

import java.awt.Point;
class PanggilPersegi {

public static void main (String args[]){
      Persegi psegi;

      System.out.println ("Persegi dengan koordinat
20,20,40,40");
      psegi = new Persegi(20,20,40,40);
      psegi.CetakPersegi();
      System.out.println("_________________________");

      System.out.println ("Persegi dengan Point (10,10),
Point (50,50)");
      psegi = new Persegi(new Point(10,10) , new
Point(50,50));
      psegi.CetakPersegi();
      System.out.println("_________________________");

      System.out.println ("Persegi dengan Point (15,15)
dan Lebar = 25 Tinggi = 20");
      psegi = new Persegi(new Point(15,15), 25,20);
      psegi.CetakPersegi();
      System.out.println("_________________________");
      }
}
```

5.5.4 Overredding Konstruktor

Sebagaimana method, konstruktor pun kita bisa override. Misalnya kita ingin membuat object baru modifikasi dari java.awt.Point

Contoh :

```
// Nama File : Point3D.java
import java.awt.Point;
class Point3D extends Point {
      int z;
      Point3D(int x, int y, int y) {
            super(x,y);
            this.z = z;
      }
}
```

```
// Nama File : Point3D.java
import java.awt.Point;
class Point3D extends Point {
      int z;
      Point3D(int x, int y, int y) {
            super(x,y);
            this.z = z;
      }
}
```

5.6 Finalizer Method

Finalizer method bisa dikatakan sebagai lawan dari construktor method, Jika constructor method digunakan untuk inisialisasi sebuah object, maka finalizer method di panggil pada saat sebelum object dilakukan garbage collection (suatu proses yang dilakukan JVM untuk menghapus alokasi di memori /optimisasi memori).

Deklarasi finalizer method : finalize().

```
Protected void finalize() {
. . .
}
```

Dalam body **finalize()** contohnya berisi perintah-perintah untuk membersihkan reference-reference dari object yang tidak digunakan lagi. Method **finalize()** berfungsi juga sebagai optimasi program.

5.7 Contoh Class Account

5.7.1 Informasi Class Account

Class Account memiliki member class seperti berikut ini :

1. Member Variable
   - variabel untuk menyimpan nomor account
   - variabel untuk menyimpan jumlah uang / saldo yang tersimpan

2. Member Method
   - Method untuk menambahkan uang (deposit) kedalam account
   - Method untuk mengurangi / mengambil uang (withdraw) dari account
   - Method mengembalikan informasi dari account berupa String (toString)

3. Konstruktor
   - Konstruktor kosong
   - Konstruktor dengan dua argumen : nomor account dan saldo awal

5.7.2 Kode Program Class Account

Berikut kode program class Account

```
/**
* Nama File : Account.java
*
**/
public class Account {
	private String no_acc = "";
	private double saldo = 0;

	public Account() {
	}

	public Account(String no, double saldo) {
		this.no_acc = no;
		this.saldo = saldo;
	}

	public void deposit(double uang) {
		this.saldo = this.saldo + uang;
	}

	public void witdraw(double uang) {
		this.saldo = this.saldo uang;
	}


	public String toString() {
		return "No.Account : " + this.no_acc + ", Saldo
: " + this.saldo;
	}
}
```

5.7.3 Menggunakan Class Account

Berikut kode program yang menggunakan class Account.

```
/**
* Nama File : AccountDemo.java
**/

public class AccountDemo {
     public static void main (String args[]) {
          Account ac1 = new Account("007",400);
          Account ac2 = new Account("001",500);

          // ac1 ditambah $20
          ac1.deposit(20);


          // cetak info ac1
          System.out.println( ac1 ); // akan eksekusi
method toString


          // ac1 dikuran $75
          ac1.withdraw(75);


          // cetak info ac1
          System.out.println( ac1 );
     }
}
```

Bab VI
Inheritance

6.1 Tujuan

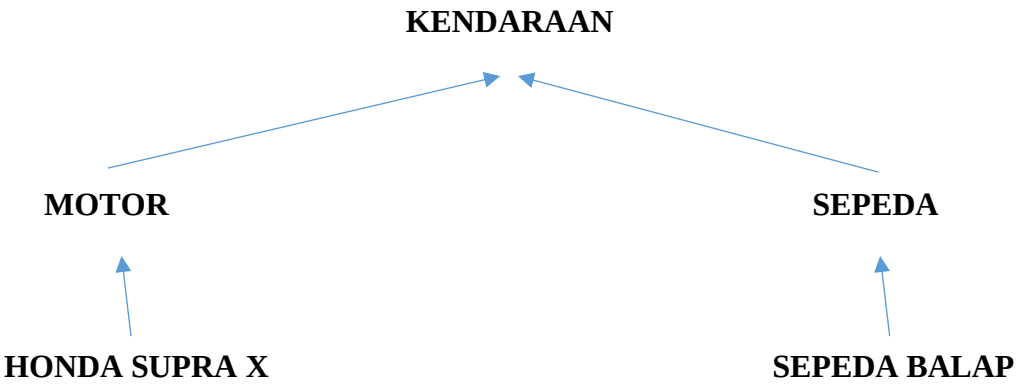
Setelah mempelajari bab ini anda diharapkan mengetahui :

- Definisi Inheretance
- Subclass dan Superclass
- Overredding Method
- Casting Object

6.2 Apa itu inheretance

Kita sering berhadapan dengan kata "Inheretance" -suatu kemampuan spesifik yang didapat dari sesuatu yang generik- di keseharian hidup kita. Contohnya, Honda Astrea SupraX warna hitam garis merah di parkir di halaman LP3T NurulFikri margonda adalah sebuah instance dari apa yang dikategorikan sebagai motor. Sementara waktu anda melihat sepeda balap melaju di depan anda yang juga adalah sebuah instance dari apa yang dikategorikan sepeda. Jika kita bandingkan sepeda dan motor pada level yang lain, keduanya mempunyai kategori yang sama yaitu sama-sama termasuk kategori kendaraan.

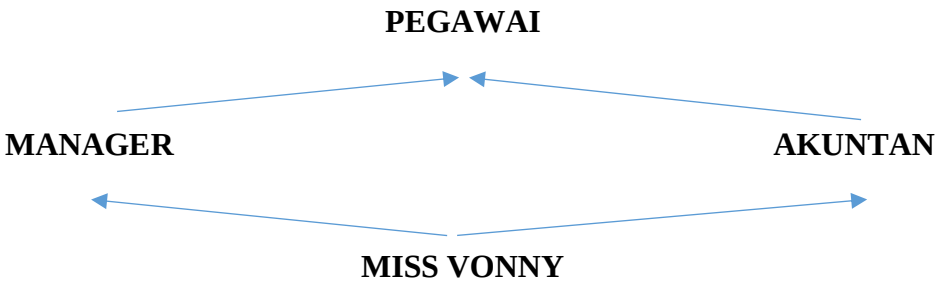
Keadaan diatas bisa digambarkan pada Gambar 6.1



Gambar 6.1: Inheretance pada Kendaraan

Contoh diatas mengilustrasikan single inheretance. Dengan single inheretance suatu entitas secara langsung adalah mewarisi (inherits) keadaan dan sipat-sipat dari satu dan hanya satu kategori. Berlainan dengan ***multiple inheretance*** yang menperbolehkan suatu entitas secara langsung mewarisi keadaan dan sipat-sipat dari dua atau lebih kategori.

Contoh Miss Vonny adalah karyawan dari suatu perusahaan. Lebih spesifik lagi Miss Vonny adalah seorang manager dan akuntan. Miss Vonny mewarisi dua kemampuan baik sebagai seorang manager maupun seorang akuntan. Bisa dilihat seperti Gambar 6.2.



Gambar 6.2 Multiple Inheretance

6.3 Extend Class

Dari perspektif Java, Inheretance adalah hasil dari perpanjangan ***(extend)*** class, disebut juga implementation inheretance dan juga hasil dari implementasi dengan interface (interface inheretance).

Berikut konsep sintak dari konsep extend class :

```
'class' classIdentifier1 'extends' classIdentifier2
'{'
// Variabelvariabel
dan methodmethod
'}'
```

Sintak diatas dibaca : `classIdentifier1` ***`extends`*** `classIdentifier2`, dengan kata lain `classIdentifier1` adalah turunan (mendapatkan kemampuan yang diekspresikan melalui variable dan method) dari `classIdentifier2`. `ClassIdentifier1` dikenal sebagai *subclass* atau **child class**, sedang `classIdentifier2` dikenal sebagai **superclass** atau **parent class.**

Keyword extends mengakibatkan subclass mewarisi semua variabel-variabel dan method-method yang didekalarasikan oleh superclass (termasuk yang ada di superclass dari superclass) dimana superclass bukan final class .

Subclass dapat mengakses semua variabel-variabel dan method-method yang bukan private atau subclass tidak dapat mengakses variabel-varaibel dan method-method yang didekalarasikan private pada superclassnya (paling tidak secara tidak langsung).

Berikut contoh Class Base sebagai parent class dari class Child.

```
/**
* Nama File Base.java
*/
class Base {
     private String nama = "";

     Base(String nama) {
          this.nama = nama;
     }

     void info()
     {
          System.out.println("Nama : " + this.nama );
     }

     String getNama() {
          return this.nama;
     }
}
```

Berikut class Child.

```
/**
* Nama File : Child.java
*/
class Child extends Base {
     private int umur = 0 ;

     Child (String nama, int umur ) {
          super( nama ) ; // panggil konstruktor parent class
          this.umur = umur;
     }

     int getUmur() {
          return this.umur;
     }


// overriding method parent
     void info() {
          System.out.println("Nama " + super.getNama() + ",
Umur " + this.umur);
     }
}
```

Kode diatas terdiri dari dua class, Base dan Child. Class Child adalah turunan dari class Base. Member class yang bukan private dapat diakses oleh Sub Class.

6.4 Mengakses Member Class dari Superclass

Class dapat mengakases member class (variabel, method dan atau konstruktor) milik superclass dengan menggunakan keyword super. Perlu diperhatikan yang hanya dapat diakses adalah member class yang mempunyai akses modifier ***selain private***.

Kode program pada class Child diantaranya menggunakan keyword super untuk memanggil contruktur dari Base.

```
Child (String nama, int umur )
{
      super( nama ) ; // panggil konstruktor parent class
      this.umur = umur;
}
```

Dua hal yang perlu diperhartikan ketika memanggil konsturktor dari superclass :

1. Konstruktor dari superclass hanya bisa dipanggil dari subclass
2. Urutan baris kode dalam menggunalan keyword harus tepat, contoh kode program padaclass Child, jika penulisan dari statement **this.umur = umur;** lebih dahulu dari statement **super(nama);** akan mengakibatkan error.

Hal ini terjadi karena kode pada konstruktor dari subclass tergantung pada variabel dan method-method yang didekalarasikan oleh superclassnya.

6.5 Overredding Method

Suatu method dalam subclass dapat menulis ulang atau mengganti (meng-overide) method yang ada di dalam superclassnya. Pada contoh sebelumnya, pada Class Child terdapat method info() yang mendefinisikan ulang method info() milik superclassnya.

```
void info()
{
      System.out.println("Nama " + super.getNama() + ", Umur "
+ this.umur);
}
```

6.6 Casting object

Setiap object pada subclass adalah juga sebuah object pada superclassnya. Misalnya object Child adalah juga object Base. Pernyataan diatas akan berimplikasi bahwa kita dapat meng-asign suatu object pada subclass reference ke object reference variabel supeclass.

Contoh :

```
Base b1 = new Child ( "Rosalie", 6);
```

Kode diatas berarti membuat sebuah object Base dan di -asign reference ke object yang reference ke variabel dari Child. Perintah casting object dapat juga dilakukan dengan contoh berikut :

```
Child c = new Child("Rosa",5);
c.info();
Base b = (Base) c; // casting object
b.info();
```

6.7 Class AccountBank extends Account

Misalkan dibuat sebuah class AccountBank sebagai turunan dari class Account (lihat bab 5)

6.7.1 Diskripsi Class AccountBank

Class AccountBank disamping memiliki nomor account bank dan saldo juga mempunyai variable untuk menyimpan nama pemilik account. Class AccountBank memiliki fitur untuk transfer antar rekening bank, dan class AccountBank juga mendefinisikan ulang method toString() milik parent classnya.

6.7.2 Kode Program AccountBank

```
/**
* Nama File : AccountBank.java
*/
public class AccountBank extends Account {
 private String nama_customer = "";
 public AccountBank (String no, String nama, double
saldo_awal ) {
      super (no, saldo_awal);
      this.nama_customer = nama;
 }

 public String toString() {
      return super.toString() + ", Pemilik " +
this.nama_customer;
 }

 public void transfer(AccountBank ac_tujuan, double uang)
 {
      ac_tujuan.deposit( uang ) ; // saldo rekening tujuan
bertambah
      withdraw( uang ); // uang rekening ini berkurang
 }
}
```

6.7.3 Menggunakan class AccountBank

```
* Nama File : TransaksiBank.java
*/
public class TransaksiBank {
     public static void main (String args[]) {
          AccountBank ab1 = new
AccountBank("007","Rosalie Naurah",650);
     // cetak info rekening rosalie
          System.out.println( ab1 );

          AccountBank ab2 = new AccountBank("010","Faiz
Fikri",300);
     // cetak info rekening faiz
          System.out.println( ab2 );

          System.out.println("Kaka Rosalie mentransfer
uang $200 ke ade Faiz ");


          ab1.transfer( ab2 , 200 );

          // cetak informasi rekening setelah proses
transfer
          System.out.println( ab1 );
          System.out.println( ab2 );
     }
}
```

Bab VII
Komposisi (Membuat object dari object-object lain)

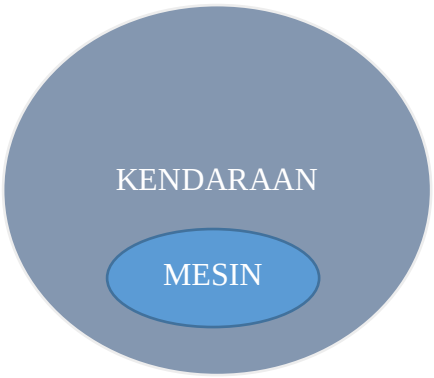
7.1 Tujuan

Setelah mempelajari bab ini anda diharapkan mengetahui :

- Difinisi komposisi
- Cara mengaplikasikan komposisi

7.2 Apa itu Komposisi ?

Didunia nyata, suatu entitas adalah komposisi dari entitas-entitas lain, contohnya entitas CPU Komputer adalah gabungan dari Motherboard, VGA Card, Casing, dan hardware lainnya. Dalam kasus lain Mobil tidak akan dikatakan sebagai mobil yang utuh dan bisa bekerja dengan baik bila tanpa ban, mesin, pintu, kaca dan lain-lain. Suatu tindakan membuat (composing) suatu object dari object-object lain disebut Komposisi (Composition / aggregation).



Gambar 7.1: Kendaraan adalah Komposisi dari Mesin

```
class Mesin {
     /* variabelvariabel dan method-method didefinisikan untuk
mengidentifikasi struktur Mesin */
}

class Kendaraan {
     Mesin mesin;
     Kendaraan (Mesin m) {
          mesin = m;
     }
}
```

Contoh diatas, di deklarasikan dua class , Mesin dan Kendaraan. Didalam class Kendaraan dideklarasikan sebuah variabel bertipe Mesin dengan nama mesin. Class Kendaraan mempunyai konstruktor dengan parameter (Mesin m) yang menginisialisasi mesin sebagai referensi dari sebuah object Mesin, yaitu dengan meng-asign parameter m ke variabel mesin.

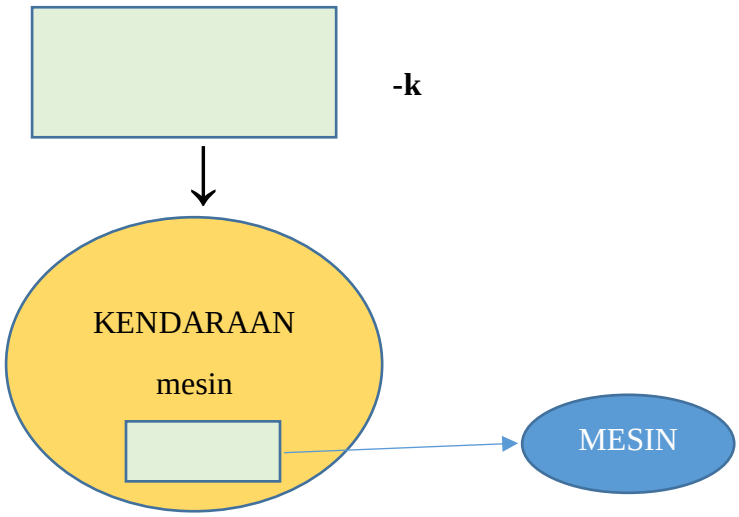
Setelah meng-asign, mesin akan mereferensi ke object yang sama sebagaimana object m (lihat gambar).

Untuk membuat object Mesin dan object Kendaraan dari object Mesin digunakan sintak berikut ini :

```
Mesin m = new Mesin();
Kendaraan k = new Kendaraan(m);
```

Pada baris pertama dibuat object Mesin dan di-asign dengan alamat dari object Mesin reference ke variabel m. Di baris kedua dibuat object Kendaraan k dengan diberi nilai object Mesin melalui (didalam m) konstruktor dari Kendaraan.

Lebih jauh lagi, konstruktor akan menyimpan isi dari m didalam variabel private mesin. Sehingga dapat dilihat bahwa object Kendaraan direferensi oleh object k untuk mengkomposisi object Mesin yang direferensi oleh variabel internal dari Kendaraan. Hubungan antara object baru dari Kendaraan dan Mesin dapat terlihat dalam gambar 7.2



Gambar 7.2: Hubungan antara Object Kendaraan dan Object Mesin dari Kendaraan

Komposisi akanlah sangat berguna ketika mengerjakan aplikasi besar.

7.3 Komposisi pada class AccountBank

Pada class AccountBank pada contoh bab 6, diwakilkan sebagai variabel String yang menyimpan nama pemilik rekening bank. Pada class AccountBank ini bisa kita perbaiki dengan menerapkan konsep komposisi dengan variabel String nama pemilik akan diubah menjadi sebuah object instance dari class Customer yang dapat menampung informasi bukan hanya sekedar nama customer saja melainkan banyak hal seperti informasi biodata lainnya.

7.3.1 Class Customer

Pada contoh berikut class Customer hanya menyimpan data nama customer dan alamat saja. Pada class ini disediakan juga method untuk mengambil member variabel nama dan alamat.

```
/**
* Nama File : Customer.java
*/
public class Customer {
	private String nama = "";
	private String alamat = "";

	public Customer (String nama, String alamat) {
		this.nama = nama;
		this.alamat = alamat;
	}

	public String getNama() {
		return this.nama;
	}

	public String getAlamat() {
		return this.alamat;
	}

	public String toString() {
		return "Nama Customer" + this.nama + ", Alamat: " +
this.alamat;
	}
}
```

7.3.2 Modifikasi class AccountBank

Pada class AccountBank informasi customer diwakili oleh object instance dari class Customer.

```
/**
* Nama File : AccountBank.java
*/

public class AccountBank extends Account {
	private Customer cust = null;

	public AccountBank (String no, Customer cust , double
saldo_awal ) {
		super (no, saldo_awal);
		this.cust = cust;
	}
```

```
	public String toString() {
		return super.toString() + ", Pemilik " +
this.cust.getNama();
	}

	public void transfer(AccountBank ac_tujuan, double uang)
	{
		ac_tujuan.deposit(uang); // saldo rekening tujuan
bertambah
		withdraw( uang ); // uang rekening ini berkurang
	}
}
```

7.3.3 Menggunakan class AccountBank

Pada implementasinya sebelum membuat object AccountBank, maka terlebih dahulu dibuat object Customer. Object Customer adalah argumen kedua dari konstruktor class AccountBank yang telah dimodifikasi.

```
/**
* Nama File : TransaksiBank.java
*/
public class TransaksiBank {
	public static void main (String args[]) {
		Customer cust1 = new Customer("Rosalie
Naurah","Jakarta");
		AccountBank ab1 = new AccountBank("007",cust1,650);
		// cetak info rekening rosalie
		System.out.println( ab1 );

		Customer cust2 = new Customer("Faiz
Fikri","Manchester");
		AccountBank ab2 = new AccountBank("010",cust2,300);


		// cetak info rekening faiz
		System.out.println( ab2 );

		System.out.println("Kaka Rosalie mentransfer uang
$200 ke ade Faiz ");
```

```
            ab1.transfer( ab2 , 200 );

            // cetak informasi rekening setelah proses transfer
            System.out.println( ab1 );
            System.out.println( ab2 );
        }
}
```

Bab VIII
Interface

8.1 Tujuan

Setelah mempelajari bab ini anda diharapkan mengetahui :

- Penggunaan interface
- Cara implementasi interface pada sebuah class

8.2 Apa itu Interface ?

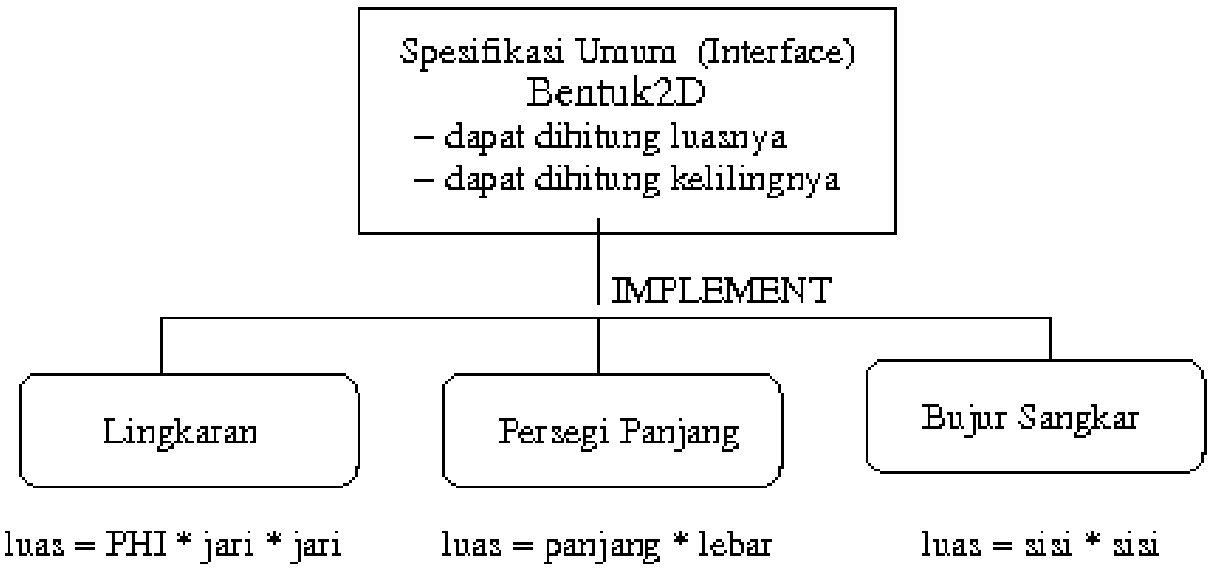
Interface adalah sebuah spesifikasi umum yang dapat dimiliki oleh sebuah object atau beberapa object yang berbeda. Contoh sederhana adalah spesifikasi umum tentang benda-benda 2 dimensi, dimana didalamnya berisi behaviours / perilaku dari bentuk 2 dimensi.

Spesifikasi Benda 2 Dimensi (interface Bentuk2D) adalah :

1. Benda 2 dimensi dapat dihitung luasnya
2. Benda 2 dimensi dapat dihitung kelilingnya

Pada program Java interface didefinisikan sebagai method-method dengan isi yang kosong atau belum didefinisikan isi dari methodnya. Method akan didefinisikan oleh class-class yang mengimplemen interface.

```
interface Bentuk2D {
float getLuas();
float getKeliling();
}
```



Gambar 8.1: Spesifikasi Umum / Interface Bentuk2D

Pada interface Bentuk2D didefinisikan method kosong untuk menghitung luas dan menghitung keliling benda-benda 2D. Bagaimana menghitung luas dari benda-benda bentuk 2 dimensi tergantung dari bentuk bendanya, contohnya untuk menghitung luas lingkaran dan luas persegi panjang tentu caranya tidak sama (lihat gambar 8.1), namun nama perilakunya tetap sama yaitu untuk mendapatkan luas dari benda 2 dimensi.

Penggunaan dari interface berfungsi sebagai alternatif dari penggunaan konsep multiple inheretance yang tidak memungkinkan pada program Java. Sebuah class di Java hanya boleh memiliki satu parentclass tetapi dapat mengimplemen beberapa interface. Sebuah class yang mengimplemen beberapa interface dapat dikatakan telah menggunakan konsep multiple inheretance.

8.3 Sintaks dari Interface

Penamaan interface mengikuti kaidah penamaan class, yaitu nama file harus sama dengan nama interface.

Berikut sintaks penulisan interface :

```
[visibility] interface InterfaceName [ extends other
interfaces] {
constant declarations
member type declarations
abstract method declarations
}
```

Berikut contoh interface Bentuk2D.

```
/**
* Nama File : Bentuk2D.java
*/
public interface Bentuk2D {
      final float PHI = 3.14f;
      public float getLuas();
      public float getKeliling();
      public String getNamaBentuk();
}
```

8.4 Class yang mengimplementasi interface

Setiap class yang mengimplementasikan suatu interface harus terdapat method-method yang ada dalam interface .

Berikut contoh tiga class bentuk 2 dimensi yang mengimplemen interface Bentuk2D.

8.4.1 Class Lingkaran

```
/**
* Nama File : Lingkaran.java
*/

public class Lingkaran implements Bentuk2D {
     private float jari = 0f;

     public Lingkaran(float jari ) {
          this.jari = jari;
     }

     public float getLuas() {
          return PHI * this.jari * this.jari;
     }

     public float getKeliling() {
          return 2f * PHI * this.jari ;
     }

     public String getNamaBentuk() {
          return "Lingkaran";
     }
}
```

8.4.2 Class PersegiPanjang

```
/**
* Nama File : PersegiPanjang.java
*/

public class PersegiPanjang implements Bentuk2D {
     private float panjang = 0f;
     private float lebar = 0f;

     public PersegiPanjang(float p , float l ) {
          this.panjang = p;
          this.lebar = l;
     }


     public float getLuas() {
          return this.panjang * this.lebar;
     }
```

```
	public float getKeliling() {
		return 2f * ( this.panjang + this.lebar);
	}

	public String getNamaBentuk() {
		return "PersegiPanjang";
	}
}
```

8.4.3 Class BujurSangkar

```
/**
* Nama File : BujurSangkar.java
*/

public class BujurSangkar implements Bentuk2D {
	private float sisi = 0f;

	public BujurSangkar(float sisi ) {
		this.sisi = sisi;
	}

	public float getLuas() {
		return this.sisi * this.sisi;
	}

	public float getKeliling() {
		return 4f * this.sisi ;
	}

	public String getNamaBentuk() {
		return "Bujur Sangkar";
	}
}
```

8.5 Implementasi Bentuk2D

Berikut program yang akan menggunakan class-class Bentuk 2 dimensi dan interface Bentuk2D

```
/**
* Nama File : Benda2D.java
*/
public class Benda2D {
     public static void main (String args[]) {
          Lingkaran l = new Lingkaran(8f);
          PersegiPanjang p = new PersegiPanjang( 7.2f, 4.5f);
          BujurSangkar b = new BujurSangkar(5f);

          // Array dari benda benda Bentuk2D
          Bentuk2D[] benda2d = { l , p , b };

          for (int i = 0 ; i < benda2d.length ; i++) {
               System.out.println( "Luas " +
benda2d[i].getNamaBentuk() + " : " + benda2d[i].getLuas());
          }
     }
}
```

Bab IX
Collection Class

9.1 Tujuan

Setelah anda mempelajari Bab ini anda diharapkan mengetahui :

- Pengertian collection
- Menggunakan class-class collection (List dan Map)

9.2 Difinisi Collection

Collection adalah sebuah object yang mewakili suatu group dari object atau elemen. Tujuan utama dari interface Collection adalah untuk mengirim object-object Collection agar secara umum bisa digunakan. Object -object collection digunakan untuk menyimpan, mengambil dan memanipulasi data.

Contoh di dunia nyata seperti Buku Telpon adalah kumpulan dari nomor-nomor telepon dan nama-nama pemilik telpon tersebut. Buku Telpon adalah Collection dan nomor telpon adalah elementnya.

9.3 Interface List

Interface List adalah turunan dari Interface Collection Class yang mengimplemen Interface List diantaranya :AbstractList, ArrayList, LinkedList, Vector.

Contoh Program :

```
//Buat list
List list = new LinkedList();// Doubly-linked list
list = new ArrayList(); // List implemented as growable array


// Memasukan sebuah elemen ke list
list.add("a");


// Memasukan sebuah element ke urutan pertama pada list
list.add(0, "b");


// Mengambil jumlah ukuran dari list
int size = list.size(); // 2


// Ambil element terakhir pada list
Object element = list.get(list.size()1); // a


// Ambil element 1 pada list
element = list.get(0); // b
```

```
// Remove element pada list yang berkesesuaian
boolean b = list.remove("b"); // true
b = list.remove("b"); // false


// Remove element pada index tertentu
element = list.remove(0); // a
String[] strArray = new String[] {"z", "a", "C"};
List list = Arrays.asList(strArray);


// Sort
Collections.sort(list); // C, a, z


// Case-insensitive sort
Collections.sort(list, String.CASE_INSENSITIVE_ORDER);//a,C,z


// Reverse-order sort
Collections.sort(list, Collections.reverseOrder());// z, a, C


// Copy semua element dari list2 ke list1 (list1 += list2)
// list1 adalah gabungan dari list1 and list2
list1.addAll(list2);


// Remove semua element pada list1 yang terdapat pada list2
// (list1 =list2)
list1.removeAll(list2)


// Ambil irisan element yang sama pada list1 dan list2 atau
// irisan list1 dan list2
list1.retainAll(list2)


// Remove semua element pada list
list1.clear();
```

9.4 Interface Map

Map adalah object key yang di mapping ke suatu nilai (value), dalam map tidak boleh ada key yang sama, setiap key dapat di map paling tidak ke satu nilai.

Class yang mengimplemen Interface List diantaranya : AbstractMap, Attributes, HashMap, Hashtable, IdentityHashMap, RenderingHints, TreeMap, WeakHashMap

Contoh Program :

```
// Create a hash table
Map map = new HashMap(); // hash table
map = new TreeMap(); // sorted map

// Menambahkan pasangan key/value ke map
map.put("a", new Integer(1));
map.put("b", new Integer(2));
map.put("c", new Integer(3));

// ambil jumlah entry pada map
int size = map.size(); // 2

// Menambahkan sebuah entry pada key yang telah ada pada map
map.put("a", new Integer(9)); // 1

// Remove sebuah entry dari map yang memiliki key
map.remove("c"); // 3
Map map = new LinkedHashMap();

// Add some elements
map.put("1", "value1");
map.put("2", "value2");
map.put("3", "value3");
map.put("2", "value4");

// List the entries
for (Iterator it=map.keySet().iterator(); it.hasNext(); ) {
Object key = it.next();
Object value = map.get(key);
}
// [1=value1, 2=value4, 3=value3]
```

Bab X
Exception

10.1 Tujuan

Setelah anda mempelajari Bab ini anda diharapkan mengetahui :

- Pengertian exception
- Menggunakan keyword try, catch dan finally
- menghandel exception dalam suatu program

10.2 Apa itu exception ?

Dalam bahasa pemrograman Java, class Exception didefinisikan untuk kondisi-kondisi kesalahan ringan (error) ketika prorgam dijalankan. Kesalahan (error) dapat ditangkap atau lebih tepatnya di handle dengan suatu exception dan program anda dapat terus berjalan.

Contoh Program jika dijalankan akan terjadi error atau exception :

- File yang coba dibuka tidak ada
- koneksi jaringan terputus
- class yang akan di load misalkan hilang

Contoh kode program berikut akan terjadi error :

```
public class HelloWorld {
     public static void main (String args[]) {
          int i = 0;

          String hello[] = {
          "Hallo Dunia",
          "Maksud saya",
          "Hallo Semua !! "
          };

          while (i < 4) {
               System.out.println(hello[i]);
               i++;
          }
     }
}
```

Ketika di compile tidak ada masalah dengan kode program diatas, namun ketika program di jalankan akan muncul Exception, seperti dibawah ini :

```
$ java HelloWorld
Hallo Dunia
Maksud saya
Hallo Semua !!
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 3
at HelloWorld.main(HelloWorld.java:14)
```

10.3 Handle Exception

Pemrograman bahasa Java menyediakan suatu mekanisme untuk menangkap exception lalu membuang exception tersebut.

1. Statement try and catch

Untuk menangani (meng-handle) suatu exception, letakan kode yang mengakibatkan exception di dalam blok :

```
try {
// kodekode
yang mengakibatkan Exception
} catch (Tipe_Exception parameter_Exception
```

Bisa saja terjadi blok catch berkali-kali (multiple catch), dimana setiap block catch menghandal exception yang berbeda-beda.

Contoh :

Pada kode program sebelumnya error terjadi ketika mengakses variabel hello[] yang bertipe array pada saat looping dengan while dijalankan. Berikut penggunaan Exception untuk meng-handle Kesalahan yang terjadi:

```
public class HelloWorld {
     public static void main (String args[]) {
          int i = 0;
          String hello[] = {
               "Hallo Dunia",
               "Maksud saya",
               "Hallo Semua !! "
          };

          while (i < 4) {
               try {
                    System.out.println(hello[i]);
               } catch (ArrayIndexOutOfBoundsException e) {
                    System.out.println("Error ... pada nilai
i");
               }

               i++;

          // tutup dari while
          }
     }
}
```

2. Menggunakan keyword finally

Keyword finally di definisikan pada block kode setiap kali di eksekusi, walaupun exception terjadi atau tidak kode finally tetap dijalankan.

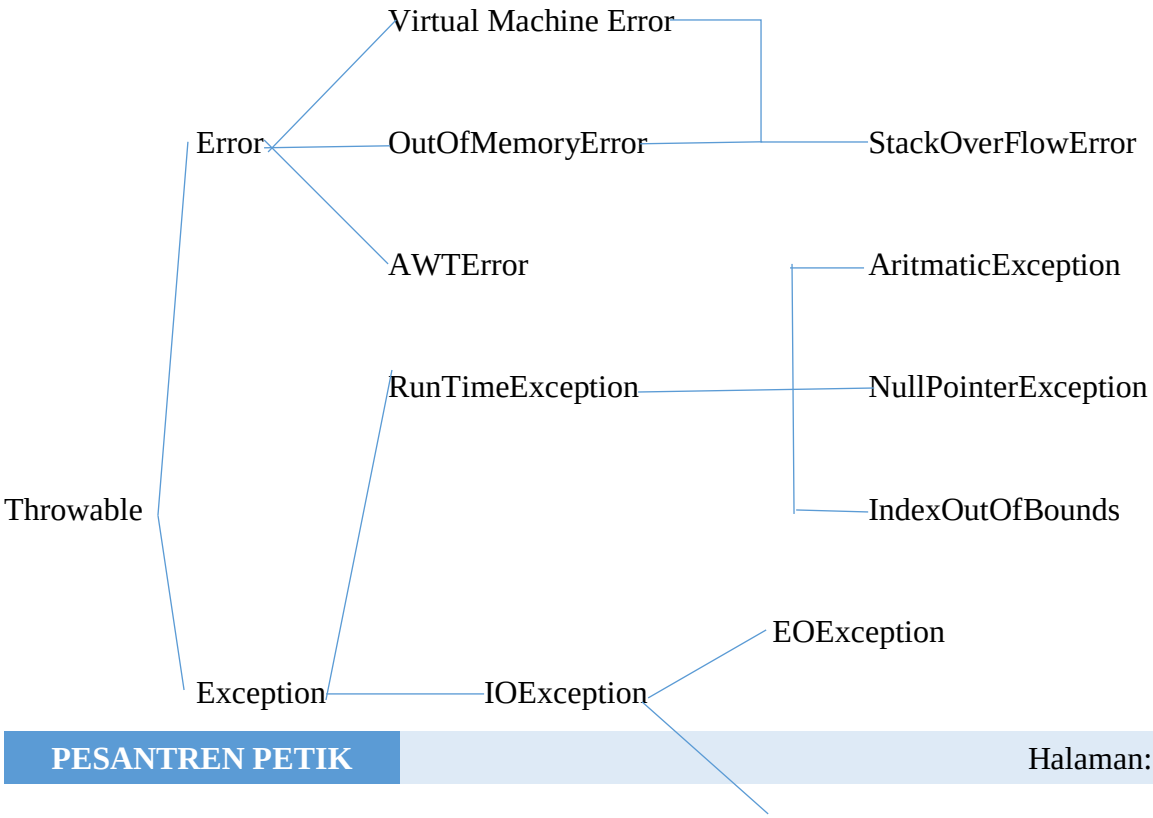
Contoh :

```
public class HelloWorld {
     public static void main (String args[]) {
          int i = 0;

          String hello[] = {
               "Hallo Dunia",
               "Maksud saya",
               "Hallo Semua !! "
          };

          while (i < 4) {
               try {
                    System.out.println(hello[i]);
               } catch (ArrayIndexOutOfBoundsException e) {
                    System.out.println("Error ...");
               } finally {
                    System.out.println("Tulisan ini akan
selalu cicetak");
               }
          i++;
          } // tutup dari while
     }
}
```

10.4 Kategori Exception



FileNotFoundException

10.5 Membuat Exception

User atau programmer dapat membuat sendiri exception yaitu dengan menggunakan class Exception yang telah disediakan.

Contoh :

```
public class ServerTimeOutException extends Exception {
      private String reason;
      private int port;
      public ServerTimeOutException (String reason, int port) {
            this.reason = reason;
            this.port = port;
      }

      public String getReason() {
            return reason;
      }


      public int getPort() {
      }
}
```

Dari program client-server, misalkan user (client) untuk koneksi ke server diperlukan hanya dalam waktu 5 detik. Jika dalam waktu tersebut server tidak merespon, kode yang kita buat dapat melempar (throw) sebuah exception (misalnya exception SeverTimeOutException) yang telah kita buat. Berikut contoh kode program bagaimana melempar (throw) sebuah exception.

```
public void connectMe(String serverName) throws
ServerTimedOutException {
      int success;
      int portToConnect = 80;
      success = open(serverName, portToConnect);


      if (success == 1) {
            throw new ServerTimedOutException ("Could not
connect", 80);
      }
}
```

Kemudian untuk menangkap exception yang telah kita buat. Kita gunakan keyword try .

```
Public void findServer() {
      . . .
      try {
            connectMe(defaultServer);
      } catch (ServerTimedOutException e) {
            System.out.println("Server time out, try
alternative");

      try {
            connectMe(alternateServer);
      } catch (ServerTimedOutException e1) {
            System.out.println("No server curremtly available");
      }
}
```

10.6 Contoh Penggunaan Exception

```
// File : Excep01.java
import java.lang.Thread;

class Excep01 {
      public static void main( String[] args){
            Excep01 obj = new Excep01();
            obj.myMethod();
      }

      void myMethod(){
            Thread.currentThread().sleep(1000);
      }
}
```

Ketika Program di Compile akan muncul pesan Error.

```
$ javac Excep01.java
Excep01.java:11: unreported exception
java.lang.InterruptedException;
must be caught or declared to be thrown
Thread.currentThread().sleep(1000);
```

Pembetulan kode program :

```
// File : Excep02.java
import java.lang.Thread;
class Excep02{
	public static void main(String[] args){
		Excep02 obj = new Excep02();
		obj.myMethod();
	}

	void myMethod() throws InterruptedException {
		Thread.currentThread().sleep(1000);
	}
}
```

Program masih error ketika di compile

```
$]javac Excep02.java
Excep02.java:5: unreported exception
java.lang.InterruptedException;
must be caught or declared to be thrown
obj.myMethod();
```

Error sekarang terjadi di main method , Error harus dihandle dengan try - catch

```
// File Excep03.java
import java.lang.Thread;
class Excep03{
	public static void main(String[] args){
		Excep03 obj = new Excep03();

		try {
			obj.myMethod();
		}catch(InterruptedException e) {
			System.out.println("Handle exception here");
		}
	}

	void myMethod() throws InterruptedException {
		Thread.currentThread().sleep(1000);
	}
}
```