

MODUL PRAKTIKUM

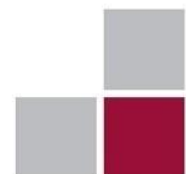
PEMROGRAMAN

BERORIENTASI

OBJEK



UNIVERSITAS YAPIS PAPUA
FAKULTAS TEKNIK DAN SISTEM INFORMASI
PROGRAM STUDI SISTEM INFORMASI
TAHUN AKADEMIK 2022/2023



MODUL PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK



PROGRAM STUDI SISTEM INFORMASI
FAKULTAS TEKNIK DAN SISTEM INFORMASI
UNIVERSITAS YAPIS PAPUA

2022

Modul Praktikum Pemrograman Berorientasi Objek

@ 2022, Program Studi Sistem Informasi

Penyusun : **Ir. Mursalim Tonggiroh, S.Kom., M.Eng.**

Email : mursalim.t@gmail.com

Program Studi Sistem Informasi
Fakultas Teknik dan Sistem Informasi
Universitas Yapis Papua
Jalan Sam Ratulangi No. 11 Dok. V Atas Jayapura

BAB I

PENGENALAN LINGKUNGAN KERJA JAVA

A. POKOK BAHASAN

- Instalasi Java Development Kit
- Pengesetan PATH dan CLASSPATH
- Cara kompilasi dan menjalankan program
- Latihan program sederhana
- Troubleshooting

B. TUJUAN BELAJAR

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

- Mengenal dan mempersiapkan lingkungan kerja Java
- Membuat program sederhana dengan Java
- Mengkompilasi dan menjalankan program sederhana Java
- Menganalisa beberapa problem yang terjadi saat pemrograman dan memberikan solusi.
- Menganalisa beberapa problem yang terjadi saat pemrograman dan memberikan solusi.

C. DASAR TEORI

➤ Pengenalan Java

Java adalah sebuah bahasa pemrograman berorientasi objek pada Computer dan dapat dijalankan pada berbagai platform sistem operasi yang dikembangkan oleh Sun Microsystems tahun 1995.

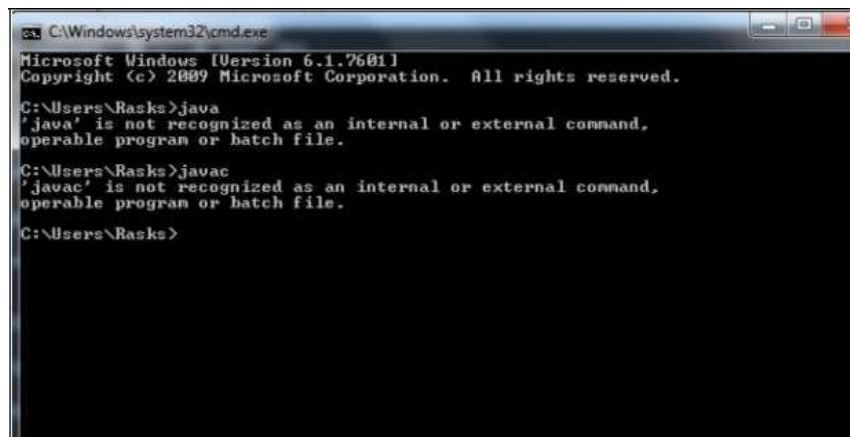
Sebagai sebuah bahasa pemrograman, Java dapat membuat seluruh bentuk aplikasi, desktop, web dan lainnya, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional yang lain.

➤ Instalasi Java JDK

Untuk bisa bekerja dengan Java, maka kita perlu melakukan instalasi Java Development Kit (JDK) atau Java 2 Software Development Kit (J2SDK). Berikut ini adalah cara instalasi JDK:

1. Buka aplikasi java
2. Klik install
3. Tunggu beberapa saat sampai proses instalasi selesai, dan close.

Setelah proses instalasi selesai, maka coba buka CMD kemudian ketikkan java untuk menjalankan program atau javac untuk cara compile java lewat cmd:



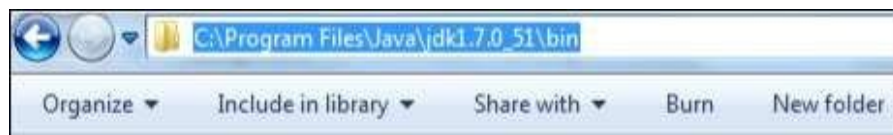
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Rask>java
'java' is not recognized as an internal or external command,
operable program or batch file.

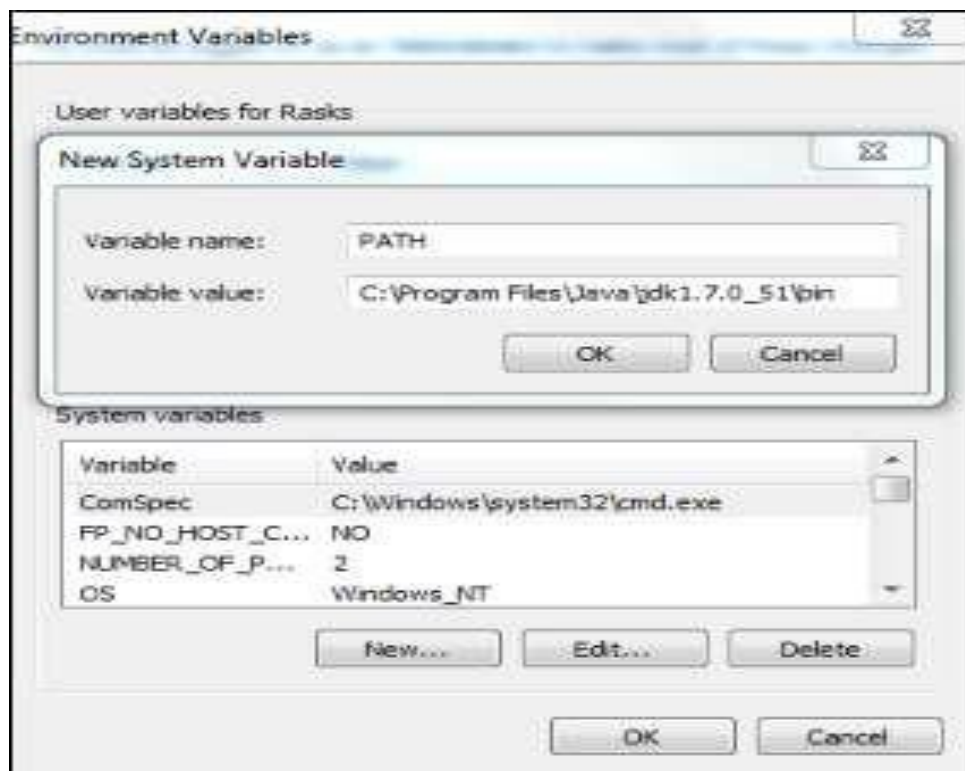
C:\Users\Rask>javac
'javac' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Rask>
```

Apabila terdapat kode seperti diatas, maka perintah compile dan menjalankan (running) java dengan command line belum bisa dilakukan karena java dan javac belum terbaca. Solusinya adalah menambahkan PATH pada My Computer. Masuk ke **C: > Program File > Java** disitu terdapat JDK dan JRE, tergantung versinya. Masuk ke JDK kemudian masuk ke folder bin, setelah itu copy alamat folder tersebut dengan klik alamat diatas:



Copy alamat tersebut lalu **Klik Kanan My Computer > Propertis > Advanced System Setting > pada Tab Advanced klik Environment Variables > Klik New pada System Variables**



Setelah itu cara compile java lewat cmd berikutnya restart komputer anda dan coba jalankan kode java dan javac pada command line. Jika hasilnya tidak seperti yang di atas, maka dapat dipastikan sudah berhasil. Untuk mengcompile gunakan kode berikut:

```
javac nama_program.java
```

Sementara untuk merunning program gunakan kode berikut :

```
java nama_program
```

Pastikan sebelum mengcompile, pada CMD anda menuju ke folder program anda. Anda dapat menggunakan kode CD untuk berpindah direktori pada CMD.

➤ Contoh Kompilasi dan Menjalankan Program

Di bawah ini adalah satu contoh program dasar sederhana yang dibuat menggunakan bahasa pemrograman java. Kita akan mencoba menampilkan

kata "Hello World".

```
public class Hello
{
    /**
     * Program java pertamaku, program dasar.
     */
    public static void main( String[] args )
    {
        //menampilkan String Hello World pada layar
        System.out.println("Hello World");
    }
}
```

D. PERCOBAAN

➤ Percobaan 1 Pengesetan PATH

```
set PATH=%PATH%;%JAVA_HOME%\bin
```

➤ Percobaan 2 Pengesetan CLASSPATH

```
set CLASSPATH=.;%JAVA_HOME%\lib\tools.jar
```

➤ Percobaan 3 Menampilkan suatu tulisan ke layar

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

Lakukan kompilasi melalui cmd dan juga netbeans!

E. LATIHAN

➤ Latihan 1

Menganalisa dan membenahi kesalahan pada program

Tulislah program berikut ini dan simpanlah dengan nama Greeting.java

Greeting.java

```
public class Testing {      public static
void main(String[] args) {
    System.out.println("What's wrong with this program?");
}
}
```

Lakukan kompilasi pada file tersebut dan amati hasilnya. Benahilah sehingga program tersebut dapat berjalan dengan baik.

➤ Latihan 2

Menganalisa dan membenahi kesalahan pada program

```
System.out.println("What's wrong with this program?");    }
```

```
public class Test {      public static void
main(String[] args) {
    System.out.println("What's wrong with this program?");    }
} public class TestAnother {      public
static void main(String[] args) {
```

➤ Latihan 3

Menganalisa dan membenahi kesalahan pada program

```
public class Test {      public static
void main(String args) {
    System.out.println("What's wrong with this program?");    }
}
```

➤ Latihan 4

Menganalisa dan membenahi kesalahan pada program

```
public class Test {      public void
main(String args[]) {
    System.out.println("What's wrong with this program?");    }
}
```


BAB II

DASAR PEMROGRAMAN JAVA

A. POKOK BAHASAN

- Identifier
- Kata kunci
- Tipe dasar
- Nilai default
- Variable

B. TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Mengetahui aturan penamaan identifier
- Mengenal kata-kata kunci yang ada di Java
- Mengetahui tipe-tipe dasar yang ada di Java
- Mengetahui penggunaan variable pada Java

C. DASAR TEORI

Penamaan identifier harus diawali dengan karakter unicode, tanda \$(dollar) atau tanda _ (underscore). Penamaan identifier ini bersifat case- sensitive dan tidak dibatasi panjang maksimum.

➤ **Keyword dalam Java**

Kata kunci adalah identifier yang telah dipesan untuk didefinisikan sebelumnya oleh Java untuk tujuan tertentu. Anda tidak dapat menggunakan keyword sebagai nama variabel, class, method.

Berikut Ini Keyword Java:

abstract	default	if	private	throws
boolean	do	import	public	Tr
break	double	int	return	void
byte	else	static	short	while
case	extends	long	super	const
catch	final	native	switch	For
char	finally	new	this	Continue
class	float	package	throw	Transient

Kata-kata kunci tersebut tidak bisa dipakai sebagai identifier. Selain kata kunci, Java juga mempunyai 3 kata literal, yaitu *true*, *false* dan *null*, yang juga tidak bisa dipakai untuk penamaan identifier.

➤ Tipe Dasar

Java mempunyai 8 tipe dasar, yaitu boolean, char, byte, short, int, long, float, dan double. Spesifikasi panjang bit dan range untuk masing-masing tipe adalah sebagai berikut:

Tipe	Panjang Bit	Range
Boolean	16	-
Char	16	0 – 2 ¹⁶ -1
Byte	8	-2 ⁷ – 2 ⁷ -1
Short	16	-2 ¹⁵ – 2 ¹⁵ -1
Int	32	-2 ³¹ – 2 ³¹ -1
Long	64	-2 ⁶³ – 2 ⁶³ -1
Float	32	-
Double	64	-

Nilai default untuk masing-masing tipe adalah sebagai berikut:

Tipe	Nilai Default
Boolean	False
Char	'\u0000'
Byte	0
Short	0
Int	0
Long	0L
Float	0.0F
Double	0.0

➤ 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. Berikut aturan penamaan variable :

- Terdiri dari huruf/abjad, angka dan underscore
- Tidak boleh mengandung karakter khusus atau spasi
- Tidak boleh diawali dengan angka

➤ Casting dan Promotion

Casting diperlukan untuk mengkonversi dari suatu tipe ke tipe data yang lebih kecil panjang bitnya. Sedangkan promotion terjadi pada saat mengkonversi dari suatu tipe data ke tipe data yang lebih panjang bitnya.

Contoh :

```
Int p = (int) 10L;  
Long i = 10;
```

D. PERCOBAAN

➤ Percobaan 1

Memberikan nilai ke suatu tipe

```
public class Tipe {  
  
    public static void main(String[] args) {  
  
        // Tipe data primitiflong  
        data1 = 767226531; int data2  
        =2235641; short data3 =714;  
        byte data4 = 34;  
  
        float data6 = (float) 1.733; // tipe data pecahandouble data5  
        = 4.967; // tipe data pecahan  
  
        char data7 = 'C'; boolean data8 = true;  
        System.out.println("Nilai Long : "+ data1);  
        System.out.println("Nilai Int : "+ data2);  
        System.out.println("Nilai Short : "+ data3);  
        System.out.println("Nilai Byte : "+ data4);  
        System.out.println("Nilai Double : "+ data5);  
        System.out.println("Nilai Float : "+ data6);  
        System.out.println("Nilai Char : "+ data7);  
        System.out.println("Nilai Boolean : "+ data8);  
  
    }  
}
```

➤ Percobaan 2

```

public class CobaUnicode { public static void
main(String args[]) { ch\u0061r a='a'; char \u0062
= 'b'; char c= '\u0063';

String kata="\u0061\u0062\u0063";
System.out.println("a: " + a);
System.out.println("a: " + b);
System.out.println("a: " + c);
System.out.println("kata: " + kata); } }

```

Memahami pemakaian Unicode

➤ Percobaan 3

Mengamati proses aritmatika dasar

```

class Aritmatika {

public static void main(String[] args){

System.out.println("Operasi aritmetika"+"padatipe integer");

int a = 2 + 1;int b = a - 1;
int c = a * b;int d = c / 3;
int e = -a;

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("Nilai      e:      "      +      e);
System.out.println();

System.out.println("Operasi aritmetika " + "padatipe floating-
point");

double fa = 2 + 1; double fb = fa -
1; double fc = fa * fb;double fd =
fc / 3; double fe = -fa;

System.out.println("Nilai fa: " + fa); System.out.println("Nilai fb:
" + fb); System.out.println("Nilai fc: " + fc);
System.out.println("Nilai fd: " + fd); System.out.println("Nilai fe:
" + fe);}

}

```

➤ Percobaan 4

Memahami cara pengimputan data dari keyboard dengan library Scanner

```
import java.util.Scanner; public class
InputKeyboard{

public static void main (String[] args) { Scanner masukan = new
Scanner(System.in);int var_a, var_b;

System.out.print("Masukkan nilai var var_a :");

var_a = masukan.nextInt(); System.out.print("Masukkan nilai
var var_b :");

var_b = masukan.nextInt(); System.out.println();
System.out.println("Variabel yang terdapatdalam program :");
System.out.println("var_a = " + var_a);
System.out.println("var_b = " + var_b);}

}
```

```
import java.util.Scanner; public class
InputKeyboard{

public static void main (String[] args) { Scanner masukan = new
Scanner(System.in);int var_a, var_b;

System.out.print("Masukkan nilai var var_a :");

var_a = masukan.nextInt(); System.out.print("Masukkan
nilai var var_b :");

var_b = masukan.nextInt(); System.out.println();
System.out.println("Variabel yang terdapatdalam program
:"); System.out.println("var_a = " + var_a);
System.out.println("var_b = " + var_b);}

}
```

```

import java.util.Scanner; public class
InputKeyboard{

public static void main (String[] args) { Scanner masukan
    = new Scanner(System.in);int var_a, var_b;

    System.out.print("Masukkan nilai var var_a :");

        var_a = masukan.nextInt(); System.out.print("Masukkan
nilai var var_b :");

        var_b = masukan.nextInt();
System.out.println(); System.out.println("Variabel
yang terdapatdalam program :");
System.out.println("var_a = " + var_a);
System.out.println("var_b = " + var_b);}

}

```

E. LATIHAN

➤ Menganalisa batasan maksimum dari suatu tipe

Amatilah dan tulislah program berikut ini:

```

public class BigInteger {

public static void main(String args[]) {
    long p=2147483648;

```

Lakukan kompilasi pada file tersebut dan amati pesan kesalahannya. Lakukan analisa mengapa bisa terjadi kesalahan padahal batasan nilai maksimum dari suatu bilangan bertipe long adalah $2^{63}-1$ (9223372036854775807)?. Kemudian berikanlah solusi yang tepat untuk mengatasi persoalan diatas.

- **Anak Ayam**

Anak ayam turun 10

Mati 1 tinggal 9

Anak ayam turun 9

Mati 1 tinggal 8

.....

Anak ayam turun 1

Mati 1 tinggal induknya

Buatlah cerita anak ayam tersebut dengan Bahasa Java !

- **A+B-C**

Diberikan 3 buah bilangan bulat, A, B, dan C, tentukan hasil operasi matematika $A + B - C$!

Input dimulai dengan satu integer T ($T \leq 10$) yang menunjukkan jumlah testcase yang akan diberikan. Setiap case akan terdiri dari 3 baris yang terdiri dari 3 integer A, B, dan C ($-100 \leq A, B, C \leq 100$).

Output untuk setiap case, cetak hasil dari $A + B - C$!

Sample Input

2 ← Menunjukkan jumlah Testcase

2 ← Nilai A1

3 ← Nilai B1

4 ← Nilai C1

-1 ← Nilai A2

-3 ← Nilai B2

-7 ← Nilai C2

Sample Output

1

3

Penjelasan

1. Case Pertama $A1+B1-C1$

$$2+3-4=1$$

2. Case kedua $A2+B2-C2$

$$-1+(-3)-(-7)=3$$

BAB III

PENGENALAN PEMROGRAMAN BERORIENTASI OBJEK

A. POKOK BAHASAN

- Pengenalan Pemrograman Berorientasi Objek
- Deklarasi class
- Deklarasi atribut
- Deklarasi metode
- Pengaksesan anggota obyek

B. TUJUAN BELAJAR

- Dengan praktikum ini mahasiswa diharapkan dapat:
- Mengetahui Pemrograman Berorientasi Objek
- Mendefinisikan suatu Class
- Mendefinisikan suatu atribut
- Mendefinisikan suatu metode
- Mengakses anggota suatu obyek

C. DASAR TEORI

➤ Pemrograman Berorientasi Objek

Pemrograman Berorientasi Objek (Object Oriented Programming/OOP) merupakan pemrograman yang berorientasikan kepada objek, dimana semua data dan fungsi dibungkus dalam class-class atau object-object.

PBO memiliki beberapa karakteristik mendasar, antara lain adalah abstraksi, encapsulation (pembungkusan), inheritance(pewarisan), dan polymorphism.

➤ Mendefinisikan suatu class

Class adalah wadah yang berisi abstraksi (pemodelan) dari suatu fungsi objek (benda), yang mendeskripsikan data (sifat karakteristik) dan fungsi yang dimiliki oleh objek tersebut.

Deklarasi class dapat dilakukan dengan sintaks sebagai berikut:

```
<modifier> class <nama_class> {  
    [isi class]  
}
```


➤ **Mendeklarasikan suatu atribut**

Attributes merupakan nilai (type) data yang terdapat pada suatu object yang berasal dari class. Attributes merepresentasikan karakteristik dari suatu object.

Deklarasi atribut dapat dilakukan dengan sintaks sebagai berikut:

```
<modifier> <type> <nama_atribut> ;
```

```
Contoh : public String warna;
```

➤ **Mendeklarasikan suatu metode**

Metode/method adalah sesuatu yang dapat dilakukan oleh objek. Method dalam implementasi program ditulis dalam bentuk fungsi. Metode menentukan apa yang terjadi ketika objek itu dibuat serta berbagai operasi yang dapat dilakukan objek.

Deklarasi metode dapat dilakukan dengan sintaks sebagai berikut:

```
<modifier> <return_type> <nama_metode>  
([daftar_argumen])
```

```
[<statement>]
```

```
}
```

```
Contoh : public void info(){  
System.out.println(warna);}
```

➤ Mengakses anggota suatu obyek

Untuk dapat mengakses anggota-anggota dari suatu obyek, maka harus dibuat instance dari class tersebut terlebih dahulu. Berikut ini adalah contoh pengaksesan anggota-anggota dari class Mobil:

```
public class Mobil {  
  
    public static void main(String args[])  
    {  
        Mobil m=new Mobil();  
        m.warna="hitam";  
        m.no_Plat="KT 2837  
        UE"; m.info();  
    }  
  
}
```

D. PERCOBAAN

➤ Percobaan 1

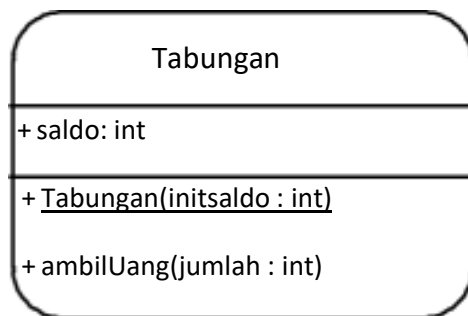
Mengakses anggota suatu class

Amati program dibawah ini:

```
public class Rumah {  
  
    String alamat_rumah;  
  
    public void setAlamat(String x) {  
        alamat_rumah=x;}  
  
    }  
  
    public class Test {  
  
        public static void main(String args[]) {Rumah  
        r=new Rumah();  
  
        r.setAlamat("Alamat saya di Jalan Pramuka");  
        System.out.println(r.alamat_rumah);}  
  
    }
```

➤ Percobaan 2

Mengimplementasikan UML class diagram dalam program untuk class Tabungan.



Transformasikan class diagram diatas ke dalam bentuk program? Tulislah listing program berikut ini sebagai pengetesan.

```
public class testTabungan {
    public static void main(String args[]) {
        Tabungan t=new Tabungan(5000);
        System.out.println("Saldo awal Tabungan Anda :
        "+t.saldo);

        t.ambiluang(1500);

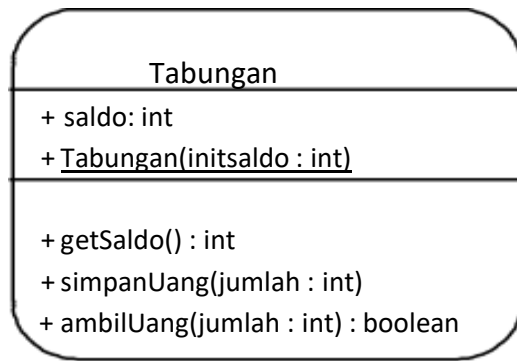
        System.out.println("Jumlah uang yang diambil : 1500");
        System.out.println("Saldo Tabungan Anda sekarang adalah
        : " + t.saldo);}
    }
```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan dilayar tampak seperti dibawah ini, maka program anda sudah benar.

```
Saldo awal Tabungan Anda : 5000
Jumlah uang yang diambil : 1500
Saldo Tabungan Anda sekarang adalah : 3500
```

E. LATIHAN

Mengimplementasikan UML class diagram dalam program untuk class Tabungan



Transformasikan class diagram diatas ke dalam bentuk program? Tulislah listing program berikut ini sebagai pengetesan.

```

public class TestTabungan {

    public static void main(String args[]) {boolean
    status;

    Tabungan tabungan=new Tabungan(10000); System.out.println("Saldo
    awal : "+tabungan.getSaldo());tabungan.simpanUang(8000);

    System.out.println("Jumlah uang yang disimpan : 8000");
    status=tabungan.ambilUang(7000); System.out.print("Jumlah uang
    yang diambil : 7000");

    {if (status) System.out.println("
    ok");else

    {System.out.println(" gagal");}
    tabungan.simpanUang(1000);

    System.out.println("Jumlah uang yang disimpan : 1000");
    status=tabungan.ambilUang(10000); System.out.print("Jumlah uang
    yang diambil : 10000");

    {if (status) System.out.println("
    ok");else

    System.out.println(" gagal");} status=tabungan.ambilUang(2500);
    System.out.print("Jumlah uang yang diambil : 2500");

    {if (status) System.out.println("
    ok");else

    System.out.println(" gagal");}
    tabungan.simpanUang(2000);

    System.out.println("Jumlah uang yang disimpan : 2000");

    System.out.println("Saldo          sekarang          =          "          +
    tabungan.getSaldo());

    }}}
  
```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan dilayar tampak seperti dibawah ini, maka program anda sudah benar.

Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

Saldo awal : 10000 Jumlah uang yang disimpan : 8000 Jumlah uang yang diambil : 7000 ok Jumlah uang yang disimpan : 1000 Jumlah uang yang diambil : 10000 ok Jumlah uang yang diambil : 2500 gagal Jumlah uang yang disimpan : 2000 Saldo sekarang = 4000

BAB IV

DASAR PEMROGRAMAN BERORIENTASI OBJEK

A. POKOK BAHASAN

- Information hiding
- Enkapsulasi
- Constructor
- Overloading constructor

B. TUJUAN BELAJAR

- Dengan praktikum ini mahasiswa diharapkan dapat:
- Menerapkan konsep enkapsulasi pada class
- Mendeklarasikan suatu constructor

C. DASAR TEORI

Information Hiding adalah menyembunyikan atribut dan method suatu objek dari objek lain. Informasi dari suatu class disembunyikan agar anggota-anggota tersebut tidak dapat diakses dari luar. Adapun caranya adalah cukup dengan memberikan akses control **private** ketika mendeklarasikan suatu atribut atau method. Contoh:

```
private int nilai;
```

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

- a. *Information hiding.*
- b. Menyediakan suatu perantara (method) untuk pengaksesan data. Contoh:

```
public class Siswa {  
    private int nilai;  
  
    public void setnilai(int n) {  
        nilai=n; }  
}
```

Constructor (konstruktor) adalah suatu method yang pertama kali dijalankan pada saat pembuatan suatu obyek. Konstruktor ini merupakan method yang berfungsi untuk menginisialisasi variabel-variabel instans yang akan di miliki oleh objek. Konstruktor dipanggil pada saat proses instansiasi kelas menjadi objek. Beberapa karakteristik yang dimiliki oleh constructor:

- a. Method constructor harus memiliki nama yang sama dengan nama class.
- b. Tidak mengembalikan suatu nilai (tidak ada keyword return).
- c. Satu class memiliki lebih dari satu constructor (overloading constructor).
- d. Dapat ditambah access modifier public, private, protected maupun default.
- e. Suatu constructor bisa dipanggil oleh constructor lain dalam satu class.

Struktur dari konstruktor yaitu :

```
Class Nama_kelas {  
    Nama_kelas() //ini adalah constructor  
    {  
        //isi konstruktor  
    }  
}
```

Contoh :

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

Overloading Constructor merupakan suatu class yang mempunyai lebih dari 1 constructor dengan syarat daftar parameternya tidak boleh ada yang sama. Contoh:

```

public class Siswa {
    private int nilai;
    public Siswa() {
        nilai=0;}

    public Siswa(int n) {
        nilai=n;}

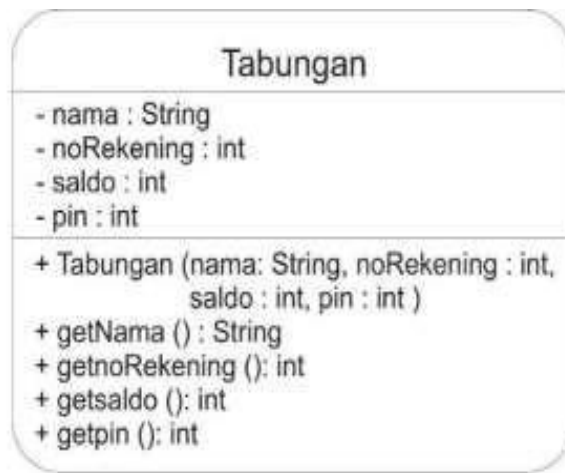
}

```

Constructor tidak dapat dipanggil secara langsung, namun harus dipanggil dengan menggunakan operator new pada pembentukan sebuah class.

D. PERCOBAAN

➤ Percobaan 1



```

public class main {

    public static void main(String[] args) {Tabungan
        saya = new Tabungan("Barca",
        50410420,1000000,12345);

        System.out.println("Nama \t : " +
        saya.getNama());
        System.out.println("noRekening \t : " +
        saya.getnoRekening());
        System.out.println("saldo \t : " +
        saya.getSaldo());

        System.out.println("pin \t : " + saya.getpin());

    }
}

```


Melakukan enkapsulasi pada suatu class

Output

```
run:
Nama           : Barca
noRekening     : 50410420
saldo          : 1000000
pin            : 12345
```

➤ Percobaan 2



```
public class mainTabungan {

    public static void main (String args[])

    {

        Tabungan saya = new Tabungan
("Barca",12345);

        Tabungan kamu = new Tabungan
("Manu",12467,15000,22222);

        System.out.println("Saldo awal saya :

"+saya.getSaldo());

        System.out.println("Saldo awal kamu :

"+kamu.getSaldo());
```

Melakukan overloading constructor

Output :

```
run:
Saldo awal saya : 10000
Saldo awal kamu : 15000
Nomor pin saya : 11111
Nomor pin kamu : 22222
```

Keterangan:

Saldo awal dan pin awal saya adalah 15000 dan 11111

E. LATIHAN

➤ Student Record

Buatlah 2 class dalam project kalian kemudia beri nama dan isisebagai berikut :

```
StudentRecord

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 static int studentCount;

    public String getName(){return
    name;

    }

    public void setName( String temp ){name =
    temp;

    }

    public double getAverage(){double
    result = 0;

    result = ( mathGrade+englishGrade+scienceGrade )/3;return
    result;

    }
```

StudentRecordExample

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

        StudentRecord Wahyu = new StudentRecord();
        StudentRecord Ini = new StudentRecord();
        StudentRecord Nama = new StudentRecord();

        Wahyu.setName("Wahyu");
        Ini.setName("Ini");
        Nama.setName("Nama");

        System.out.println( getName() );
        System.out.println("Hitung="+StudentRecord.getStudentCount
        ()
        );
    }
}
```

Hasil dari running program diatas adalah sebagai berikut:

```
Wahyu
Hitung=0
```

Analisislah dan tentukan konsep apa saja yang digunakan (dari modul ini) serta mengapa hasil akhirnya seperti itu!

BAB V

MENGELOLA CLASS

A. POKOK BAHASAN

- Package
- Import class
- Kata kunci *this*

B. TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Memahami konsep package dan import
- Menggunakan kata kunci *this*

C. DASAR TEORI

➤ Package

Package adalah suatu cara untuk memenej class-class yang kita buat package akan sangat bermanfaat jika class-class yang kita buat sangat banyak sehingga perlu dikelompokkan berdasarkan kategori tertentu. Yang harus diperhatikan adalah biasanya nama class utama harus sama dengan nama package. Berikut ini adalah sintak pernyataan package untuk meletakkan hasil kompilasi sebuah kelas ke dalam paket:

```
package nama-paket;  
  
<identifier> class <nama_class sama dengan package>  
Contoh : package Utama;
```

➤ Import Class

Perintah import digunakan untuk memberitahukan kepada program untuk mengacu pada class-class yang terdapat dalam package tertentu, buka menjalankan class-class tersebut.

Deklarasi import class:

```
import <nama_package>.*;  
Contoh : import matematik.*;
```

Sedangkan untuk mengimpor class tertentu anda

dapat menuliskan nama class setelah nama package. Deklarasi import :

```
import <nama_package>.<nama_class>;  
Contoh : import matematik.bilangan;
```

Contoh penggunaan import (dengan implementasinya):

```
import matematik.bilangan;  
class Utama {  
  
    public static void main(String[] args) {  
        Bilangan x = new Bilangan(); x.setDesimal(-  
        44); System.out.println("Bilangan biner dari  
        "+  
  
        x.getDesimal() + " adalah "+ x.biner()); }}
```

➤ Kata kunci *this*

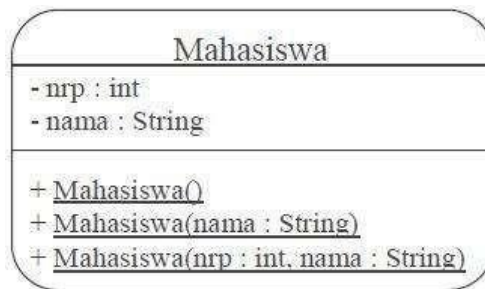
Kata kunci ini digunakan dalam sebuah kelas untuk menyatakan object sekarang. Kata kunci *this* sangat berguna untuk menunjukkan suatu member dalam class-nya sendiri. *this* dapat digunakan baik untuk data member maupun untuk function member, serta dapat juga digunakan untuk konstruktor. Adapun format penulisannya adalah:

this.data_member	merujuk pada data member
this.function_member()	merujuk pada function member
this()	merujuk pada konstruktor

D. PERCOBAAN

➤ Percobaan 1

Memakai kata kunci *this* pada overloading constructor



Dari class diagram tersebut, dapat diimplementasikan ke dalam program sebagai berikut :

```
public class Mahasiswa {
    private int nrp;
    private String nama;
    public Mahasiswa() {

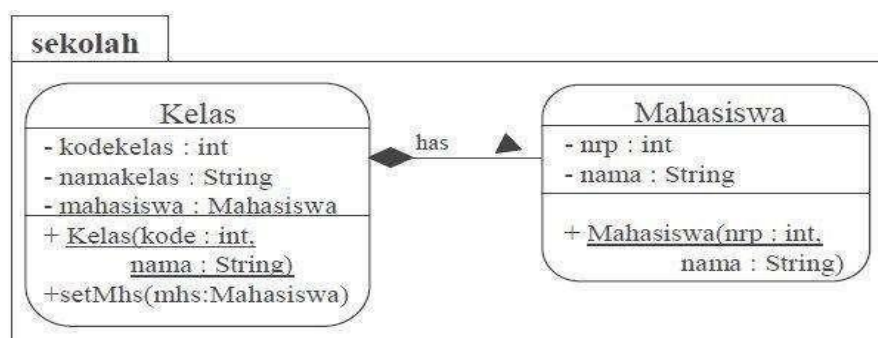
        this(0, "");
    }

    public Mahasiswa(String nama) {
        this(0, nama);
    }

    public Mahasiswa(int nrp, String nama) {
        this.nrp=nrp;
        this.nama=nama; } }
```

➤ Percobaan 2

Menggunakan Package dan import



Dari class diagram tersebut, dapat diimplementasikan ke dalam program dibawah ini. Sebelum melakukan kompilasi, daftarkan direktori tempat package diatas disimpan.

```

package sekolah; public
class Kelas { private int
kodekelas; private String
namakelas; private
Mahasiswa mahasiswa;
public Kelas(int kode,
                String nama)
{
    this.kodekelas=kode;
    this.namakelas=nama;
}

public void setMhs
(Mahasiswa mhs) {
    this.mahasiswa=mhs;}
}

package sekolah;

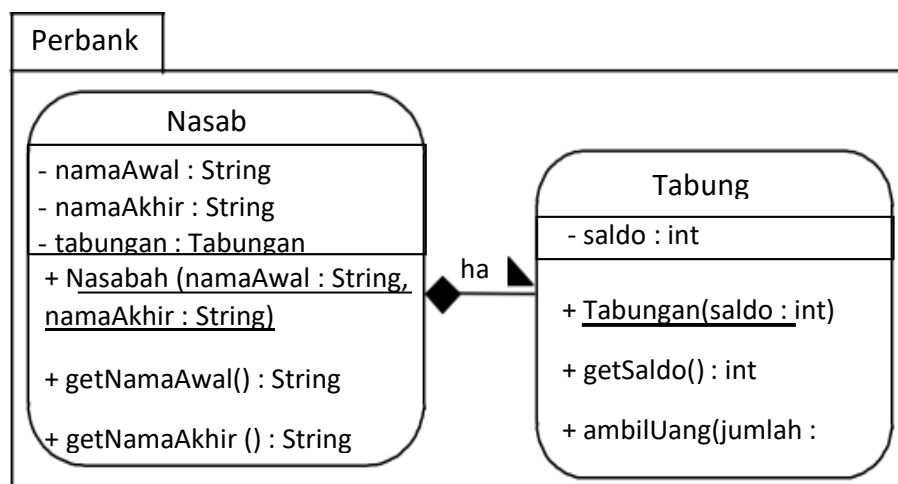
public class Mahasiswa {
    private int nrp;
    private String nama;
    public Mahasiswa(int
nrp, String nama) {
        this.nrp=nrp;
        this.nama=nama;
    }
}

```

E. LATIHAN

Mengimplementasikan UML class diagram dalam program untuk package perbankan

Transformasikan class diagram diatas ke dalam bentuk program! Tulislah listing program berikut ini sebagai pengetesan.



```

import perbankan.*;

    private class TesLatihan {

        private static void main(String args[]) {
            int tmp;

            boolean status;

            Nasabah nasabah=new Nasabah("Agus","Daryanto");
            System.out.println("Nasabah atas nama : " +
            nasabah.getNamaAwal() + " " + nasabah.getNamaAkhir());
            nasabah.setTabungan(new Tabungan(5000));
            tmp=nasabah.getTabungan().getSaldo();
            System.out.println("Saldo awal : " + tmp);
            nasabah.getTabungan().simpanUang(3000);
            System.out.println("Jumlah uang yang disimpan :
            3000"); status=nasabah.getTabungan().ambilUang(6000);
            System.out.print("Jumlah uang yang diambil : 6000");
            if (status) System.out.println(" ok"); else

            System.out.println(" gagal");
            nasabah.getTabungan().simpanUang(3500);
            System.out.println("Jumlah uang yang disimpan : 3500");
            status=nasabah.getTabungan().ambilUang(4000);
            System.out.print("Jumlah uang yang diambil : 4000");

            if (status) System.out.println(" ok");
            else

            System.out.println(" gagal");
            status=nasabah.getTabungan().ambilUang(1600);
            System.out.print("Jumlah uang yang diambil : 1600");

            if (status) System.out.println(" ok");
            else

            System.out.println(" gagal");
            nasabah.getTabungan().simpanUang(2000);
            System.out.println("Jumlah uang yang disimpan :
            2000"); tmp=nasabah.getTabungan().getSaldo();
            System.out.println("Saldo sekarang = " + tmp);} }

```

Lakukan kompilasi pada program diatas dan jalankan. Jika tampilan di layar tampak seperti dibawah ini, maka program anda sudah benar. Jika tidak sama, benahi kembali program anda dan lakukan hal yang sama seperti diatas.

```

Nasabah atas nama : Agus Daryanto
Saldo awal : 5000
Jumlah uang yang disimpan : 3000
Jumlah uang yang diambil : 6000 ok
Jumlah uang yang disimpan : 3500
Jumlah uang yang diambil : 4000 ok
Jumlah uang yang diambil : 1600 gagal
Jumlah uang yang disimpan : 2000
Saldo sekarang = 3500

```


BAB VII

KONSEP INHERITANCE

A. POKOK BAHASAN

- Pengertian inheritance
- Deklarasi inheritance dan Single inheritance
- Penerapan inheritance
- Pengaksesan member dari parent class
- Kontrol pengaksesan
- Kata kunci *super*
- Konstruktor tidak diwariskan

B. TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Memahami dan menerapkan konsep inheritance dalam pemrograman
- Melakukan pengontrolan akses pada pengkodean
- Memahami pengaksesan member pada parent class
- Menggunakan kata kunci *super*
- Menghindari kesalahan pada pewarisan konstruktor

C. DASAR TEORI

Dengan konsep 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-nya. Kesimpulannya, boleh dikatakan bahwa suatu sub class adalah tidak lain hanya memperluas (extend) parent class-nya.

Di dalam Java untuk mendeklarasikan suatu class sebagai subclass dilakukan dengan cara menambahkan kata kunci *extends* setelah deklarasi nama class, kemudian diikuti dengan nama parent class-nya. Berikut adalah contoh deklarasi inheritance.

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

Contoh:

Contoh diatas memberitahukan kompiler 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 memperkenalkan adanya single inheritance. Konsep single inheritance hanya memperbolehkan suatu subclass mempunyai satu parent class. Dengan konsep single inheritance ini, masalah pewarisan akan dapat diamati dengan mudah.

Suatu parent class dapat tidak mewariskan sebagian member-nya kepada subclass-nya. Sejauh mana suatu member dapat diwariskan ke class lain, ataupun suatu member dapat diakses dari class lain, sangat berhubungan dengan access control (kontrol pengaksesan). Di dalam java, kontrol pengaksesan dapat digambarkan dalam tabel berikut ini:

Modifier	class yang sama	package yang sama	subclass	class manapun
private	√			
default	√	√		
protected	√	√	√	
public	√	√	√	√

Kata kunci *super* dipakai untuk merujuk pada member dari parent class, sebagaimana kata kunci *this* yang dipakai untuk merujuk pada member dari class itu sendiri.

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

1. Pemanggil super () HARUS DIJADIKAN PERNYATAAN PERTAMA DALAM constructor.

2. Pemanggil `super()` hanya dapat digunakan dalam definisi constructor.
3. Termasuk constructor `this()` dan pemanggil `super()` TIDAK BOLEH TERJADI DALAM constructor YANG SAMA.

```
public class Siswa {  
    private int nilai;  
  
    public setNilai(int nilai) {  
        this.nilai=nilai;  
    }  
}
```

Contoh:

D. PERCOBAAN

➤ Percobaan 1

Menggunakan kata kunci *super*

Berikut ini listing penggunaan kata kunci `super` untuk membedakan atribut superclass dengan atribut subclass.

```
class Bentuk {  
    protected int p,l; }  
  
class Persegi extends Bentuk {  
    protected int p,l;  
  
    public void setSuperP(int p){  
        super.p = p; }  
  
    public void setSuperL(int l){  
        super.l = l; }  
  
    public void setP(int p){  
        this.p = p; }  
  
    public void setL(int l){  
        this.l = l; }  
}
```

```
public void getLuas() {  
    System.out.println("Luas super:"+(super.l*super.p));  
    System.out.println("Luas:"+(this.l*this.p)); }  
  
}  
  
class PersegiTest {  
  
    public static void main(String[] args){  
        Persegi kotak=new Persegi();  
        kotak.setSuperP(5);  
        kotak.setSuperL(10);  
  
        kotak.setP(3);  
        kotak.setL(13);  
        kotak.getLuas();  
  
    } }  
}
```

Ketika program tersebut dijalankan, akan tampak hasil seperti dibawah ini:

```
Luas Super : 50  
Luas : 39
```

➤ Percobaan 2

Mendefinisikan Superclass dan Subclass

Buatlah 3 class dalam project kalian kemudia beri nama dan isi sebagai berikut:

Person

```
public class Person { protected String name;protected
String address;

public Person(){

System.out.println("Inside Person:Constructor");name
= "";

address = "";

}

public Person( String name, String address) {

this.name = name;

this.address = address;

}


public String getName() {

return name;

}

public String getAddress() {

return address;

}

public void setName(String name) {

this.name = name;

}

public void setAddress(String add)

{ this.address = add; }

}
```

Student

```
public class Student extends Person{  
    public Student()  
  
    {  
        //super( "Ini Nama", "Ini Alamat");  
        //super();  
        //super.name = "Ini Nama";  
        System.out.println("Inside Student:Constructor");  
    }  
}
```

InheritDemo

```
public class InheritDemo {  
    public static void main(String[] args) {  
        Student Wahyu = new Student();  
    }  
}
```

Hasil dari running program diatas adalah sebagai berikut:

```
Inside Person:Constructor  
Inside Student:Constructor
```

➤ Percobaan 3

Kontrol pengaksesan

Cobalah listing program berikut:

```

class B extends A {private int z;

public void getJumlah(){ System.out.println("jumlah:"+
(x+y+z));

}

public void setZ(int z){this.z = z; } }

class A { private int x;private int y;

public void setX(int x){

this.x = x;

public void setY(int y){this.y = y;

}

public void getNilai(){

System.out.println("nilai x:"+ x +" nilai y:" + y);

}

class InheritanceTest{

public static void main(String [] args)

{

A ortu = new A();B anak = new B();

System.out.println("superclass");ortu.setX(10);

ortu.setY(20); ortu.getNilai(); System.out.println("sub
Class");anak.setX(5);

anak.setY(4); anak.getNilai(); anak.setz(50);
anak.getJumlah();

}

}

```

Sekarang cobalah untuk mengkompilasi program diatas. Apa yang terjadi? Mengapa timbul pesan kesalahan dan buatlah listing program yang benar sehingga tidak timbul pesan kesalahan tersebut.

➤ Percobaan 4

Konstruktor tidak diwariskan

Buatlah class kosong bernama Parent seperti dibawah:

```
public class Parent {  
}
```

Buatlah class Child yang menurunkan class Parent seperti dibawah ini

```
public class Child extends Parent {  
    int x;  
  
    public Child() { x = 5; super(); } }
```

Lakukan kompilasi pada Child diatas. Apa yang terjadi?. Pasti disana terjadi error. Sekarang ubahlah sedikit class Child diatas seperti dibawah ini:

```
public class Child extends Parent {  
    int x;  
  
    public Child() {  
        super();  
  
        x = 5; }}
```

Setelah dikompilasi, anda tidak mendapatkan error sebagaimana yang sebelumnya. Ini yang harus kita perhatikan bahwa untuk pemanggilan konstruktor parent class, kita harus melakukan pemanggilan tersebut di baris pertama pada konstruktor subclass.

BAB 7

OVERLOADING DAN OVERRIDING

A. POKOK BAHASAN

- Overloading
- Overriding
- Aturan tentang Overridden method

B. TUJUAN BELAJAR

Dengan Praktikum ini mahasiswa diharapkan dapat :

- Memahami tentang overloading
- Memahami tentang overriding
- Memahami aturan tentang overridden

C. DASAR TEORI

Overloading adalah suatu keadaan dimana ada beberapa method dengan nama yang sama pada suatu class tetapi dengan parameter yang berbeda (mempunyai implementasi dan return value). Tujuan dibuatnya overloading yaitu memudahkan pengguna method dengan fungsi yang hampir sama. Contoh:

```
public void print( String temp ){
    System.out.println("Name:" +
    name);
    System.out.println("Address:" +
    address);
    System.out.println("Age:" +age);
}

public void print(double eGrade, double
mGrade, double sGrade)
System.out.println("Name:" + name);
System.out.println("Math Grade:" + mGrade);
System.out.println("English Grade:" +
eGrade); System.out.println("Science Grade:"
+ sGrade);
}
```

Overloading ini dapat terjadi pada class yang sama atau pada suatu parent class dan subclass-nya. Overloading mempunyai ciri-ciri sebagai berikut :

- a. Nama Method harus sama
- b. Daftar parameter harus berbeda
- c. Return type boleh sama, juga boleh berbeda.

Overriding tidak sama dengan overloading, Overriding merupakan mekanisme dimana sebuah metode dapat dideklarasikan ulang pada kelas turunannya. Overriding mempunyai ciri-ciri sebagai berikut :

- a. Nama Method harus sama
- b. Daftar parameter harus sama
- c. Return type harus sama

Berikut ini contoh terjadinya overriding dimana method Rupiah Vs Dolar() pada class Sekarang meg-override method Rupiah Vs Dolar() pada class Dulu.

```
class Dulu {  
    public String RupiahVsDolar() {  
        System.out.println("Rp 10.000");  
    }  
}  
  
class Sekarang extends Dulu { public  
    String RupiahVsDolar() {  
        System.out.println("Rp 14.000");  
    }  
}
```

Method yang terkena override (overiden method) diharuskan tidak boleh mempunyai modifier yang lebih luas aksesnya dari method yang meng-override (overriding method).

D. PERCOBAAN

➤ Percobaan 1

Melakukan Overloading pada method.

Tulislah listing program berikut ini dan amati yang terjadi pada saat terjadinya overloading pada method.

```
class A {}

class B extends A {} class C
extends B {} public class
Overload
{
    void myOverload(A a)
    {
        System.out.println("Overload.myOverload(A a)");
    }

    void myOverload(B b)
    {
        System.out.println("Overload.myOverload(B b)");
    }

    public static void main(String[] args)
    {
        Overload o = new Overload(); C c =
new C(); o.myOverload(c);

        /*
            *statement di atas akan menjalankan
myOverload(B b), karena
            *method tersebut lebih "dekat" dengan method yang
dicari
            *bila dibandingkan dengan myOverload(A a)
        */
    }
}
```

➤ Percobaan 2

Melakukan Overloading pada method

Tulislah listing program berikut ini dan amati yang terjadi pada saat terjadinya overloading pada method.

```
public class Overload1
{
    void myMethod(short s)
    {
        System.out.println("short");
    }

    void myMethod(int i)
    {
        System.out.println("int");
    }

    void myMethod(long l)
    {
        System.out.println("long");
    }

    public static void main(String[] args)
    {
        byte b = 1;
        Overload1 o = new Overload1();
        o.myMethod(b);
        /*
        *statement di atas akan menghasilkan "short",
        *hal ini karena short lebih "dekat" dengan
        *byte bila dibandingkan dengan int ataupun long.
        */
    }
}
```

BAB 8

POLIMORFISME

A. POKOK BAHASAN

- Konsep dasar polimorfisme
- Virtual Method Invocation
- Polymorphic arguments
- Pernyataan instanceof
- Casting object

B. TUJUAN BELAJAR

Dengan praktikum ini mahasiswa diharapkan dapat:

- Memahami dan menerapkan konsep polimorfisme dalam pemrograman
- Memahami proses terjadinya Virtual Method Invocation
- Memahami dan menerapkan polymorphic arguments dalam pemrograman
- Memahami penggunaan instanceof dan cara melakukan casting object

C. DASAR TEORI

Polymorphism (polimorfisme) adalah kemampuan untuk mempunyai beberapa bentuk class yang berbeda. Polimorfisme ini terjadi pada saat suatu obyek bertipe parent class, akan tetapi pemanggilan constructornya melalui subclass. Misalnya deklarasi pernyataan berikut ini:

dimana Manager() adalah kontruktor pada class Manager yang merupakan subclass dari class Employee.

```
Employee employee=new Manager();  
  
//<nama class> <variable bebas> = new <kontruktor>();
```

Virtual Method Invocation (VMI) bisa terjadi jika terjadi polimorfisme dan overriding. Pada saat obyek yang sudah dibuat tersebut memanggil overridden method pada parent class, kompiler Java akan melakukan invocation (pemanggilan) terhadap overriding method pada subclass, dimana yang seharusnya dipanggil adalah overridden method. Berikut contoh terjadinya VMI:

```

class Parent {
    int x = 5;

    public void Info() {
        System.out.println("Ini class Parent");
    }
}

class Child extends Parent {
    int x = 10;

    public void Info() {
        System.out.println("Ini class Child");
    }
}

public class Test {
    public static void main(String[] args) {
        Parent tes=new Child();

        System.out.println("Nilai x = " + tes.x);
        tes.Info();
    }
}

```

Hasil dari running program diatas adalah sebagai berikut:

```

Nilai x = 5
Ini class Child

```

Polymorphic arguments adalah tipe suatu parameter yang menerima suatu nilai yang bertipe subclass-nya. Berikut contoh dari polymorphics arguments:

```

class Pegawai {
    class Pegawai {
        ...
        ...
    }
}

class Manajer extends Pegawai {
    class Manajer extends Pegawai {
        ...
    }
}

public class Tes {
    public class Tes {
        public static void Proses(Pegawai peg) {
            ...
        }
    }

    public static void main(String args[]) { Manajer man = new public staticvoid main(String
args[]) { Manajer man = new Manajer(); Proses(man); Manajer(); Proses(man);
    }
}
}
}

```

Pernyataan instanceof sangat berguna untuk mengetahui tipe asal dari suatu polymorphic arguments. Untuk lebih jelasnya, misalnya dari contoh program sebelumnya, kita sedikit membuat modifikasi pada class Tes dan ditambah sebuah class baru Kurir, seperti yang tampak dibawah ini:

```

...

class Kurir extends Pegawai {
    ...
}

public class Tes {
    public static void Proses(Pegawai peg) {if
(peg instanceof Manajer) {
        ...lakukan tugas-tugas manajer...
    } else if (peg instanceof Kurir) {
        ...lakukan tugas-tugas kurir...
    } else {
        ...lakukan tugas-tugas lainnya...
    }
}
}

```

Seringkali pemakaian instanceof diikuti dengan casting object dari tipe parameter ke tipe asal. Misalkan saja program kita sebelumnya. Pada saat kita sudah melakukan instanceof dari tipe Manajer, kita dapat melakukan casting object ke tipe asalnya, yaitu Manajer. Caranya adalah seperti berikut:

```
...
if (peg instanceof Manajer) { Manajer man = (Manajer) peg;
    ...lakukan tugas - tugas manajer...
}
```

D. PERCOBAAN

Virtual Method Invocation

Pegawai

```
public class Pegawai { private String name;
    private String address;private int
    number;

    public Pegawai(String name, String address,int number){

        System.out.println("Menyusun Pegawai");this.name = name;

        this.address = address;this.number =
        number;

    }

    public void mailCheck(){ System.out.println("Memeriksa Surat Untuk " +
this.name+ " " + this.address);

    }

    public String toString(){

        return name + " " + address + " " + number;

    }

    public String getName(){return name;

    }

    public String getAddress(){return address;

    }

    public void setAddress(String newAddress){ address =
        newAddress;

    }

    public int getNumber(){return number;

    }

}
```


Gaji

```
public class Gaji extends Pegawai {  
    private double salary; //Gaji Tahunan  
  
    public Gaji(String name, String address, int number, double  
        salary){  
        super(name, address, number); setSalary(salary);  
    }  
  
    public void mailCheck(){  
        System.out.println("Memeriksa kelas gaji dalam surat ");  
        System.out.println("Surat tertuju untuk "  
+ getName()  
        + " dengan gaji " + salary);  
    }  
  
    public double getSalary(){ return salary;  
    }  
  
    public void setSalary(double newSalary){ if(newSalary >=  
        0.0){  
        salary = newSalary;  
    }  
    }  
  
    public double computePay(){  
        System.out.println("Menghitung pembayaran gaji untuk  
" + getName());  
        return salary/52;  
    }  
}
```

VirtualDemo

```
public class VirtualDemo{

    public static void main(String [] args){

        Gaji s = new Gaji("Wahyu", "KUBAR", 3, 5000.00); Pegawai e =
        new Gaji("Ini nama", "Samarinda", 2, 2500.00);

        System.out.println("Memanggil mailCheck Berdasarkan Referensi Gaji
--");

        s.mailCheck();

        System.out.println("\nMemanggil mailCheck
Berdasarkan Referensi Pegawai--");

        e.mailCheck();

    }

}
```

Hasil dari running program diatas adalah sebagai berikut:

```
Menyusun Pegawai
Memanggil mailCheck Berdasarkan Referensi Gaji --
Memeriksa kelas gaji dalam surat
Surat tertuju untuk Wahyu dengan gaji 5000.0

Memanggil mailCheck Berdasarkan Referensi Pegawai--
Memeriksa kelas gaji dalam surat
Surat tertuju untuk Ini nama dengan gaji 2500.0
```

Analisislah mengapa hasil akhirnya seperti ini!

BAB 9

EXCEPTION

A. POKOK BAHASAN

- Exception handling

B. TUJUAN PEMBELAJARAN

- Memahami mengenai exception.
- Memahami tipe exception yaitu Checked Exception dan Unchecked Exception.
- Menangani exception menggunakan try, catch, finally, throw dan throws

C. DASAR TEORI

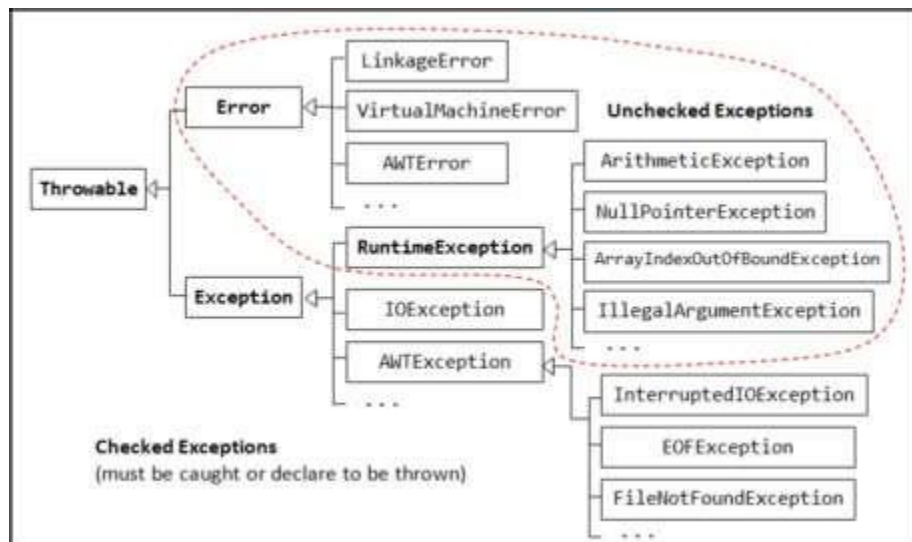
Exception adalah suatu kondisi abnormal yang terjadi pada saat menjalankan program. Teknik yang dipakai dalam Java untuk menangani kondisi yang tidak biasa saat menjalankan operasi normal dalam program dinamakan exception handling. Karena dalam java segala sesuatu merupakan objek, maka exception juga direpresentasikan dalam sebuah objek yang menjelaskan tentang exception tersebut. Contoh exception adalah pembagian bilangan dengan 0, pengisian elemen array diluar ukuran array, kegagalan koneksi database, file yang akan dibuka tidak ada, dan mengakses objek yang belum diinisialisasi. Terdapat dua penanganan exception yaitu:

- a. Menangani sendiri exception tersebut.
- b. Meneruskannya ke luar dengan cara membuat objek tentang exception tersebut dan melemparkannya (throw) keluar agar ditangani oleh kode yang memanggil method (method yang didalamnya terdapat exception) tersebut.

Ada lima keyword yang digunakan oleh Java untuk menanganixception yaitu try, catch, finally, throw dan throws.

• Tipe-Tipe Exception

Pada exception, superclass tertinggi adalah class Throwable, tetapi kita hampir tidak pernah menggunakan class ini secara langsung. Di bawah class Throwable terdapat dua subclass yaitu class Error dan class Exception.



• Penggunaan Blok try-catch

Untuk menangani exception dalam program, dengan meletakkan kode program yang memungkinkan terjadinya exception didalam blok try, diikuti dengan blok catch yang menentukan jenis exception yang ingin ditangani.

Contoh :

```
public class Percobaan2 {  
  
    public static void main(String[] args)  
  
    {  
        int a[] = new  
        int[5]; try{  
  
        a[5] = 100 ;  
  
        } catch (ArrayIndexOutOfBoundsException  
        e){ System.out.println("Indeks Array  
        melebihi batas");  
  
        }  
    }  
}
```

Output :

Terjadi exception karena Indeks Array melebihi batas Setelah blok try-catch

Dapat terjadi kode yang terdapat dalam blok try mengakibatkan lebih dari satu exception. Dalam hal ini, kita dapat menuliskan lebih dari satu blok catch. Contoh :

```
public class Percobaan5 {  
  
    public static void main(String[] args)  
  
    { int bil=10;  
  
    String b[] = {"a","b","c"};  
    try{  
        System.out.println(bil/0);  
        System.out.println(b[3]);  
  
    }catch(ArithmeticException e){  
        System.out.println("Error Aritmetik");  
  
    }catch(ArrayIndexOutOfBoundsException e){  
        System.out.println("Error Kapasitas Array Melebihi Batas");  
  
    }catch(Exception e){  
        System.out.println("Terdapat Error");  
  
    }  
}
```

• Menggunakan Keyword "finally"

Terdapat kode yang harus dijalankan walaupun terjadi atau tidak terjadi exception, misalkan kita membuka file, hal ini memungkinkan terjadinya exception misal file tidak ada, file tidak bisa dibuka, selanjutnya yang harus dilakukan adalah menutup file tersebut.

```
public class Percobaan2 {  
  
    public static void main(String[] args)  
  
    { int a[] = new int[5]; try{  
  
  
  
    a[5] = 100 ;  
  
    }catch(ArrayIndexOutOfBoundsException e){  
        System.out.println("Terjadi exception karena Indeks Array melebihi batas");  
  
    }finally{  
  
        System.out.println("Selalu Dijalankan");  
  
    } System.out.println("Setelah blok try-catch");  
  
    }  
}
```

- **Menggunakan Keyword "throw " dan "throws"**

Secara eksplisit, kita dapat melempar (throw) exception dari program menggunakan keyword throw. Jika exception tersebut adalah checked exception, maka pada method harus ditambahkan throws. Jika exception tersebut adalah unchecked exception, maka pada method tidak perlu ditambahkan throws.

```
public class Percobaan6 { public static void method1()
throws FileNotFoundException{

throw new FileNotFoundException("File Tidak Ada");

} public static void main(String[] args)
{ try {
method1();
} catch (FileNotFoundException ex){
System.out.println(ex.getMessage());
}
} }
```

D. PERCOBAAN

➤ Percobaan 1

Memahami cara menangkap Exception dengan tipe Array Index Out Of Bounds Exception

```
public class Percobaan2 {

public static void main(String[] args) {
int a[] = new int[5];

try{

a[5] = 100 ;

} catch (ArrayIndexOutOfBoundsException e){
System.out.println("Indeks Array melebihi batas");
}

}
```

➤ Percobaan 2

Jalankan percobaan 2, bagaimana output program? Perbaiki dengan Percobaan22 untuk menangani exception.

```
public class Percobaan2 {

    public static void main(String[] args) {
        int bil = 10 ;
        System.out.println(bil/0);
    }
}
```

```
public class Percobaan22 {

    public static void main(String[] args) {
        int bil = 10 ; try{
            System.out.println(bil/0;

        }catch(ArithmeticException e){
            System.out.println("Tidak boleh membagi
            bilangan dengan 0");
        }
    }
}
```

➤ Percobaan 3

Memahami try bertingkat.

```
public class Percobaan4 {

    public static void main(String[] args)

    { int bil = 10 ; try{
        System.out.println(bil/0)

    }catch(ArithmeticException e){
        System.out.println("Terjadi exception
        karena tidak boleh membagi bilangan dengan
        0"); }catch(Exception e){
        System.out.println("Terdapat Error");
    }
}
```

➤ Percobaan 4

Penggunaan finally

```
public class ExceptTest{

    public static void main(String args[]){
        int a[] = new int[2];

        try{

            System.out.println("Access element three : " +
            a[3]); }catch(ArrayIndexOutOfBoundsException e){
            System.out.println("Exception thrown : " + e);

        }

        finally{ a[0] = 6;
            System.out.println("First element value: "
            +a[0]); System.out.println("The finally statement
            is executed");
        }
    }
}
```

E. LATIHAN

➤ Latihan 1

Method yang melempar checked exception

```
import java.io.FileNotFoundException;
public class percobaan7 { public
static void method1() throws
FileNotFoundException{

throw new FileNotFoundException("File Tidak Ada");

}

public static void main(String[] args) {

try {method1();}catch (FileNotFoundException ex)

{ System.out.println(ex.getMessage()); }}
```

➤ Latihan 2

Method yang melempar unchecked exception

```
public class percobaan {

public static void main (String args[]) {
Scanner sc = new Scanner(System.in);

try {

System.out.print ("Masukan Angka : ");
int num = sc.nextInt();

if (num>10) throw new Exception();
System.out.println("Angka kurang dari atau sama dengan
10");}

catch (Exception s) {
System.out.println("Angka lebih dari 10");}
System.out.println("Selesai");}}
```

➤ Latihan 3

Menggunakan konsep Inheritance untuk membuat superclass dan subclass exception. Program menangani exception dengan menangkap subclass exception dengan superclass.


```
import javax.swing.*;

class ExceptionA extends Exception {}
class ExceptionB extends ExceptionA {}
class ExceptionC extends ExceptionB {}
public class Demo {

    public static void main( String args[] )

    {try {

        throw new ExceptionC();}
        catch( ExceptionA a ) {

            JOptionPane.showMessageDialog( null,
            a.toString(), "Exception",
            JOptionPane.INFORMATION_MESSAGE ); }

        try {

            throw new ExceptionB(); } catch(
            ExceptionA b ) {
                JOptionPane.showMessageDialog(
                null, b.toString(), "Exception",
                JOptionPane.INFORMATION_MESSAGE );}

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