Petunjuk Praktikum

Pemrograman Berorientasi Objek

User

Program Studi Teknik Informatika

Fakultas Teknologi Industri

Universitas Ahmad Dahlan

2017

Kata Pengantar

Puji syukur kami panjatkan kehadirat Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan pembuatan petunjuk praktikum ini. Petunjuk praktikum ini dibuat untuk membantu praktikan dalam praktikum mata kuliah Pemrograman Berorientasi Objek. Petunjuk praktikum ini telah disusun dalam dua belas pertemuan dan disesuaikan dengan Rencana Pembelajaran Semester mata kuliah Pemrograman berorientasi Objek. Hal ini bertujuan agar praktikan dapat mempelajari isi petunjuk praktikum dengan baik.

Dengan selesainya petunjuk praktikum ini, maka kami tidak lupa mengucapkan banyak terima kasih. Kami juga menyampaikan terima kasih kepada semua pihak yang terlibat dalam penyusunan petunjuk praktikum Pemrograman Berorientasi Objek ini.

Demikian petunjuk praktikum Pemrograman Berorientasi Objek yang telah kami buat. Kami mohon kritik dan sarannya apabila terdapat kekurangan dalam penyusunan petunjuk Praktikum ini. Semoga petunjuk Praktikum Pemrograman Berorientasi Objek ini dapat bermanfaat.

Yogyakarta, 1 September 2017

Tim penyusun:
Drs. Tedy Setiadi, M.T. – NIY. 60030475
Herman Yuliansyah, S.T., M.Eng. – NIY. 60110648
Supriyanto, S.T., M.T. – NIY. 60160952

Editor:
Rizky Aulia Dwiyanty - 1600018216

Daftar Isi

Kata Pengantar i
Daftar Isi ii
Pengenalan Bahasa Java 5
A. Teori Pendukung 5
B. Pre Test [Nmaks = 100] 8
C. Langkah Praktikum 8
D. Post Test [Nmaks = 100] 9
E. Daftar Pustaka 9
Struktur Kendali dan Array 5
A. Teori Pendukung 5
B. Pre Test [Nmaks = 100] 8
C. Post Test [Nmaks = 100] 9
D. Daftar Pustaka 9
Class dan Objek pada Java 10
A. Teori Pendukung 10
B. Pre Test [Nmaks =100] 12
C. Langkah Praktikum 12
D. Post Test [Nmaks =100] 12
E. Daftar Pustaka 12
Konstruktor, Enkapsulasi & Hidding Information 13
A. Teori Pendukung 13
B. Pre Test (Nmaks =100) 15
C. Langkah Praktikum 15
D. Post Test [Nmaks = 100] 15
E. Daftar Pustaka 15
Pewarisan(Inheritance) 16
A. Teori Pendukung 16
B. Pree Test (Nmaks =100) 17
C. Langkah Praktikum 18
D. Post Test (Nmaks =100) 18
E. Daftar Pustaka 18
Polymorphisme 19
A. Teori Pendukung 19
B. Pre Test [Nmaks =100] 21
C. Langkah Praktikum 21
D. Post Test [Nmaks =100] 21
E. Daftar Pustaka 21
Implementasi Relasi Diagram Kelas 22
A. Teori Pendukung 22
B. Pre Test (Nmaks =100) 27
C. Langkah Praktikum 27
D. Post Test (Nmaks =100) 27
E. Daftar Pustaka 29
Abstract Class 30
A. Teori Pendukung 30
B. Pre Test [Nmaks =100] 31
C. Langkah Praktikum 31
D. Post Test [Nmaks =100] 32
E. Daftar Pustaka 32

Interface 33
A. Teori Pendukung 33
B. Pre Test [Nmaks =100] 34
C. Langkah Praktikum 34
D. Post Test [Nmaks =100] 35
Input Output dan Exception Handling 36
A. Teori Pendukung 36
B. Pre Test [Nmaks =100] 36
C. Langkah Praktikum 36
D. Post Test [Nmaks =100] 39
E. Daftar Pustaka 39
Package 40
A. Teori Pendukung 40
B. Pre Test [Nmaks =100] 41
C. Langkah Praktikum 41
D. Post Test [Nmaks =100] 42
E. Daftar Pustaka 42
Aplikasi Games 54
A. Teori Pendukung 55
B. Pre Test [Nmaks =100] 58
C. Langkah Praktikum 58
D. Post Test [Nmaks =100] 60
E. Daftar Pustaka 60
F. Lembar Kerja 61

Pengenalan Bahasa Java

Pertemuan ke : **I**
Alokasi Waktu : **1,5 Jam**
Kompetensi Dasar : **Dasar-dasar Java**
Indikator : **mampu membuat program sederhana dengan bahasa pemrograman java**

A. Teori Pendukung

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

Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai platform sistem operasi. Perkembangan Java tidak hanya terfokus pada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat open source.

Bahasa pemrograman Java pada awalnya dibuat oleh James Gosling pada tahun 1995 sebagai bagian dari Sun Microsystem Java Platform. Sintaks Java banyak diturunkan dari C dan C++ tetapi lebih sederhana, ketat dan mempunyai akses ke OS yang lebih terbatas. Hal ini karena Java ditujukan sebagai bahasa pemrograman yang cukup sederhana untuk dipelajari dan mudah dibaca.

Aplikasi Java ditulis sebagai file berekstensi .java yang dicompile menjadi file .class. File .class ini adalah bytecode yang bisa dijalankan di semua Java Virtual Machine, tidak peduli apapun OS-nya ataupun arsitektur processornya. Java adalah bahasa yang ditujukan untuk semua kebutuhan, concurent, berbasis class, object oriented serta didesain agar tidak tergantung terhadap lingkungan dimana aplikasi dijalankan (OS dan processor).

Java Platform terdiri dari tiga buah profile : Java ME (Java Micro Edition) adalah java yang bisa berjalan di dalam embedded system seperti Java Card dan Handphone. Java SE (Java Standard Edition) adalah java yang bisa berjalan di dalam PC maupun server sebagai aplikasi standalone maupun aplikasi desktop. Java EE (Java Enterprise Edition) adalah profile java yang ditujukan untuk membuat aplikasi Enterprise seperti Web Application (Servlet) dan Enterprise Java Bean (EJB).

Sebelum mengenal yang namanya kelas, terlebih dahulu belajar tentang tipe data, variabel, operator, statement kontrol pada bahasa pemrograman java. Yang pertama kita harus kenal tipe data dan cara menggunakan variabel didalam lingkungan java. Berikut tipe data yang digunakan dalam Java.

Tipe Data

Terdapat beberapa tipe data primitive yang ada di Java yaitu :

Tipe Data	Keterangan
boolean	true atau false
char	Karakter
byte	-128 - 127
short	-32768 - 32767
int	-2147483648 - 2147483647
long	-9223372036854775808 - 9223372036854775807
double	4.9E-324 - 1.7976931348623157E308
float	1.4E-45 - 3.4028235E38

String bukanlah merupakan tipe data di Java, String merupakan Object. Namun string memiliki keunikan yaitu String dapat langsung dibuat tanpa harus membuat Object.

Variabel

Variabel merupakan sesuatu yang digunakan untuk menampung sebuah data. Sebuah variabel harus ada dalam sebuah kelas atau metode. Pembuatan sebuah variabel di Java terlihat pada

kode dibawah ini.
Tipe namaVariabel; // mendeklarasikan variabel
Tipe namaVariabel= nilai; //inisialisasi variabel
Contoh 1.1 Pendeklarasian variabel:

```
Int nilai;
Int nilai=0;
```

Setelah membahas tentang variabel, kita lanjut pengenalan operator di java. Operator di Java tidak beda dengan di C++. Berikut beberapa operator dalam pemrograman Java.

Operator

Operator merupakan sebuah karakter khusus yang digunakan untuk menghasilkan suatu nilai. Berikut beberapa macam operator yang ada dalam java.

Operator Logika

Operator	Keterangan
+	Penjumlahan
-	Pembagian
*	Perkalian
/	Pembagian
%	Sisa

Operator Penugasan

Operator	Keterangan
=	Pemberian nilai
+=	Penambahan bilangan
-=	Pengurangan bilangan
*=	Perkalian bilangan
/=	Pembagian bilangan
%=	Pemerolehan sisa bagi

Operator Pembanding

Operator	Keterangan
==	sama
!=	Tidak sama
>=	Lebih dari sama dengan
<=	Kurang dari sama dengan
>	Lebih dari
<	Kurang dari

Operator Logika

Operator	Keterangan
&&	Dan
||	Atau

Setelah membahas tentang operator, lanjut ke statement kontrol.
Berikut beberapa contoh penulisan program dengan menggunakan bahasa pemrograman java.

Contoh 1.2 Program perulangan

Souce code	Hasil

public class Perulangan {
  public static void main(String[] args) {
  for (int i = 1; i <=5; i++) {
  System.out.println(i+"= Belajar Java itu mudah");
  }
  }
}
	run:
	1= Belajar Java itu mudah
	2= Belajar Java itu mudah
	3= Belajar Java itu mudah
	4= Belajar Java itu mudah
	5= Belajar Java itu mudah
	BUILD SUCCESSFUL (total time: 0 seconds)

Contoh 1.3 Program percabangan/ kondisi

Source code	Hasil

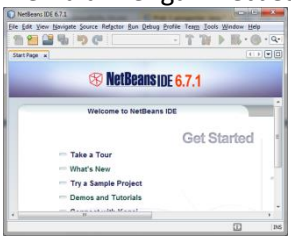
public class Percabangan {
  public static void main(String[] args) {
  int a = 2;
  if (a < 5) {
  System.out.println("Nilai a lebih kecil dari 5");
  } else {
  System.out.println("Nilai a lebih besar dari 5");
  }
  }
}
	run:
	Nilai a lebih kecil dari 5
	BUILD SUCCESSFUL (total time: 1 second)

B. Pre Test [Nmaks = 100]

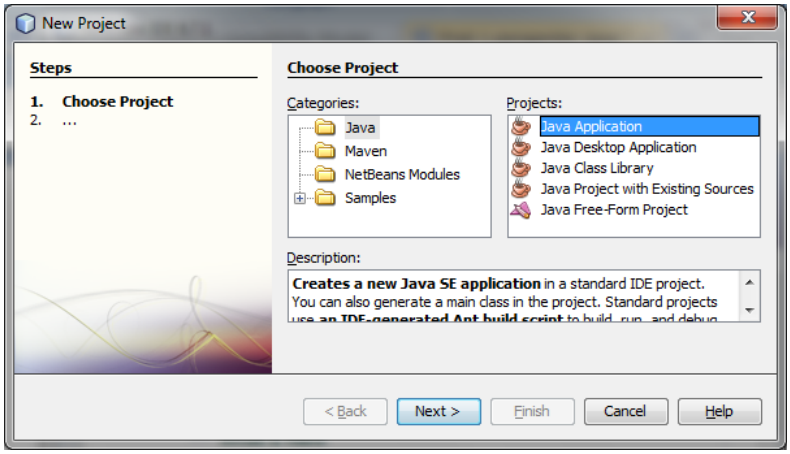
1. Tuliskan ciri-ciri bahasa pemrograman berorientasi object?
2. Tuliskan cara mendeklarasikan dan menginisialisasikan variabel?
3. Sebutkan dan berikan contoh beberapa operator? (min 3)
4. Tuliskan sintaks percabangan dan perulangan?

C. Langkah Praktikum

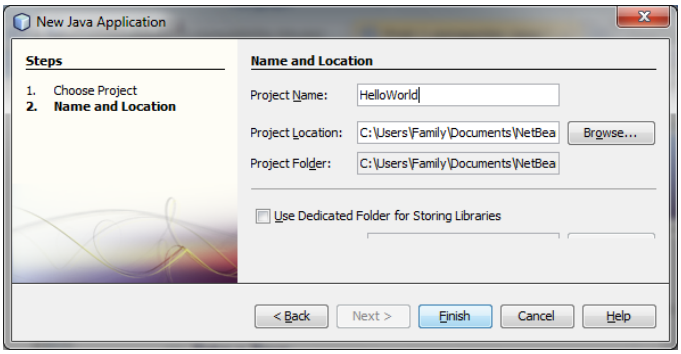
1. Memulai Dengan Netbeans klik 2x aplikasi netbeans



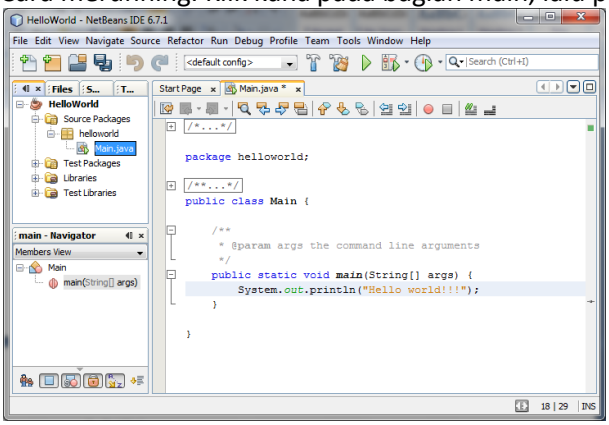
2. Pilih New Project, akan tampil cendela seperti di bawah ini.



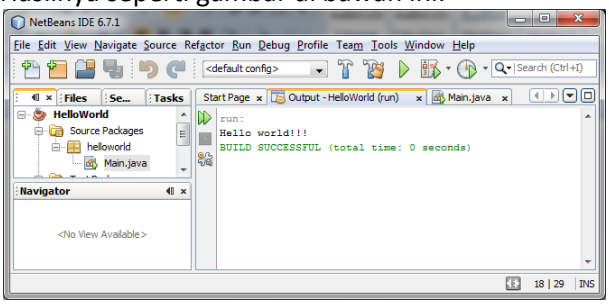
3. Plih java aplication, klik next. Ketikkan nama aplikasi yang mau dibuat. Misal HelloWorld.



4. Finish
5. Cara merunning. Klik kana pada bagian main, lalu pilih



6. Hasilnya seperti gambar di bawah ini.



D. Post Test [Nmaks = 100]

Buatlah program untuk

1. Menampilkan kalimat hello world sampai 100 baris.
2. Membaca jari-jari lingkaran kemudian menampilkan keliling dan luas lingkaran
3. Membaca tiga bilangan bulat kemudian menampilkan bilangan bulat terbesarnya dan terkecilnya
4. Membaca nilai angka kemudian menampilka nilai hurifnya (A:80-100, B:65-79, C:55-64, D:40-54, E :0-53)
5. Membaca sebuah bilangan bulat positif N, menamplkan banyaknya N bilangan genap pertama.

E. Daftar Pustaka

1. Raharjo, budi dkk.2007.”*Mudah Belajar Java*”.Bandung:informatika.
2. Bima,ifnu.2011.”Java Desktop”.Singapore.
3. H.M Deitel, Java How to Program, Sixth Edition. Prentice Hall, August 2004

Struktur Kendali dan Array

Pertemuan ke : 2
Alokasi Waktu : 1,5 Jam
Kompetensi Dasar : Struktur pencabangan dan perulangan dan penggunaan Array
Indikator : mampu membuat program dengan struktur kendali dan stuktur Array

A. Teori Pendukung

Strukur Percabangan pada Java

Sintaks if

Sintaks if-else, sebagai berikut :

```
if (boolean expression) {
    statement or block
} else if (boolean expression) {
    statement or block
} else {
    statement or block
}
```

Contohnya:

```
if (nilai >= 55){
    System.out.println("lulus "):
}
else{
    System.out.println("tidak lulus");
}
```

Sintaks switch

```
Syntax switch sebagai berikut :
switch (expression) {
    case constant1 :  statements;
                break;
    case constant2 :  statements;
                break;
    default :   statements;
          break;
    }
```

Contohnya:

```
switch(x){
case 1: System.out.println("Senin");
    break;
case 2: System.out.println("Selasa");
    break;
case 3: System.out.println("Rabu");
    break;
case 4: System.out.println("Kamis");
    break;
case 5: System.out.println("Jum'at");
    break;
case 6: System.out.println("Sabtu");
    break;
case 7: System.out.println("Minggu");
    break;
default: System.out.println("Salah input hari");
    break;
}
```

Ekspresi bersyarat

Digunakan untuk menggantikan sebuah bentuk if else. Sintaks adalah sebagai berikut :

```
exp1 ? exp2 : exp3
```

Dalam hal ini, jika exp1 adalah benar, exp2 dieksekusi.Jika exp2 adalah salah, exp3 dieksekusi.

Contoh :

```
int a = 50;
int b = 100;
int nilai = 3 > 2 ? a : b;
Outputnya :  nilai = 50
```

Perulangan pada Java

Perulangan for

Syntax for sebagai berikut :

```
for (init_expr;boolean testexpr;alter_expr) {
    statement or block
}
```

Contoh :

```
for (int i=0;i<5;i++) {
    System.out.println(i);
}
```

Perulangan while

Sintaks looping while sebagai berikut :

```
while (boolean testexpr) {
    statement or block
}
```

Contoh :

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

Perulangan do....while

Sintaks do/while loop, sebagai berikut

```
do {
    statement or block
} while (boolean testexpr)
```

Contoh :

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

Array

Mendeklarasikan Array

Array biasanya digunakan untuk mengelompokkan data atau obyek yang memiliki tipe yang sama. Array memungkinkan untuk mengacu ke sekumpulan obyek dengan nama yang sama.
Array dapat dideklarasikan dengan tipe apa saja, baik itu yang bertipe data primitif maupun obyek. Berikut contoh deklarasi Array :

```
char s[];
point p[]; // point adalah sebuah kelas
```

Dalam bahasa pemrograman Java, Array merupakan sebuah obyek meskipun ia terdiri dari elemen yang bertipe data primitif. Seperti halnya kelas yang lain, ketika mendeklarasikan Array belum dibentuk sebuah obyek Array. Deklarasi Array hanya membuat sebuah referensi yang dapat digunakan untuk mengacu ke sebuah obyek Array.

Besarnya memori yang digunakan oleh elemen-elemen Array akan dialokasikan secara dinamis dengan menggunakan pernyataan new atau dengan array initializer.
Contoh deklarasi Array di atas, dengan menggunakan kurung siku setelah nama variabel, merupakan standar penulisan array dalam C, C++ dan Java. Format tersebut agak sulit untuk dibaca. Oleh karena itu, bahasa Java menggunakan bentuk deklarasi Array alternatif dengan menggunakan kurung siku di sebelah kiri seperti dalam contoh berikut :

```
char [] s;
point [] p; // point adalah sebuah kelas
```

Sebuah deklarasi Array dapat ditulis sebagai pendeklarasian tipe Array di bagian kiri dan nama variabel di bagian kanan. Kedua bentuk deklarasi Array di atas dapat digunakan, tetapi sebaiknya konsisten dalam menggunakan satu bentuk saja. Dalam deklarasi Array, tidak ditentukan ukuran aktual dari Array. Ketika mendeklarasikan Array dengan kurung siku yang berada di sebelah kiri nama variabel, kurung siku tersebut berlaku bagi semua variabel yang berada di sebelah kanannya.

Membuat Array

Array dibuat dengan menggunakan keyword new. Contoh di bawah ini membuat sebuah Array dengan tipe primitif char:

```
s = new char[26];
```

Indeks Array dimulai dengan angka 0. Batasan legal untuk nilai indeks ini adalah dari nol hingga jumlah elemen Array – 1. Apabila program berusaha mengakses Array di luar batas yang legal, maka akan muncul runtime exception.
Untuk membuat Array dengan elemen obyek, gunakan cara penulisan berikut

```
p = new point[10];
```

Pernyataan tersebut tidak membuat 10 obyek Point. Untuk membuat obyek Point, gunakan pernyataan berikut :

```
p[0] = new point(0, 1);
p[1] = new point(1, 2);
```

Menginisialisasi Array

Java memiliki cara singkat untuk membuat sebuah obyek Array dengan elemen-elemennya memiliki nilai awal :

```
String names[] = {"Joko","Joni","Jujuk"};
Kode di atas adalah sama dengan kode berikut ini:
String names[];
names[0] = "Joko";
names[1] = "Joni";
names[2] = "Jujuk";
```

Cara singkat ini dapat digunakan untuk Array dengan berbagai jenis
tipe, seperti dalam contoh berikut :

```
MyDate dates[] = {
new MyDate(22, 7, 1964),
new MyDate(1, 1, 2000),
new MyDate(22, 12, 1964),
};
```

Array Multidimensi

Dalam Java, dapat dibuat Array dari Array sehingga dinamai Array multidimensi.

Contoh berikut ini memperlihatkan cara membuat Array dua dimensi :

```
int twoDim [][] = new int[4][];
twoDim[0] = new int[5];
twoDim[1] = new int[5];
```

Array dari Array yang bersifat non-rectangular dapat dibuat seperti dalam contoh berikut :

```
twoDim[0] = new int[2];
twoDim[1] = new int[4];
twoDim[2] = new int[6];
twoDim[3] = new int[8];
```

Untuk membuat Array dari Array yang rectangular dengan cara mudah, gunakan bentuk berikut ini :

```
int twoDim[][] = new int[4][5];
```

Batasan Array

Dalam bahasa Java, indeks Array dimulai dengan angka nol. Jumlah elemen di dalam Array disimpan sebagai bagian dari obyek Array di dalam atribut length. Atribut ini dapat digunakan untuk melakukan proses iterasi pada Array seperti dalam contoh berikut :

```
int list[] = new int[10];
for(int i = 0; i < list.length; i++) {
System.out.println(list[i]);
```

Dengan menggunakan atribut length, pemeliharaan kode program menjadi mudah karena program tidak perlu mengetahui jumlah eleman Array pada saat kompilasi.

Manipulasi Array

Mengubah Ukuran Array

Setelah membuat obyek Array, ukuran Array tersebut tidak dapat diubah. namun demikian, dapat digunakan variabel referensi yang sama untuk menunjuk ke sebuah obyek Array baru seperti dalam contoh di bawah ini :

```
int myArray[] = new int[6];
myArray = new int[10];
```

Dalam kode program ini, obyek Array yang pertama akan hilang kecuali ada variabel lain yang dibuat untuk merujuk ke obyek Array tersebut.

Menyalin *Array*

Bahasa Java menyediakan method khusus untuk menyalin Array, yaitu dengan menggunakan method arrayCopy() dari kelas System seperti dalam contoh berikut :

```
int myArray[] = {1, 2, 3, 4, 5, 6};
// array engan elemen yang lebih banyak
int hold[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
//menyalin semua elemen myArray ke hold
// dimulai dengan indeks ke 0
System.arraycopy(myArray, 0, hold, 0,myArray.length);
```

Setelah proses pengkopian di atas, maka isi dari Array hold adalah 1, 2, 3, 4, 5, 6, 4, 3, 2, 1.

B. Pre Test [Nmaks = 100]

1. Jelaskan pengertian Array!
2. Tuliskan cara mendeklarasikan Array beserta contohnya!
3. Sebutkan dan jelaskan cara memanipulasi Array!

C. Post Test [Nmaks = 100]

Buatlah program untuk

1. Membaca suhu air kemudian menampilkan wujud air tersebut apakah beku, suhu <=0, cair suhu >0 dan suhu <=100 atau uap suhu >100).
2. Buatlah sebuah program dengan menggunakan bahasa Java untuk menampilkan bilangan prima antara 1 s.d. 100!
3. Membaca N buah data bilangan bulat positif N, kemudian menampilkan
   a. Jumlah total
   b. Rata-rata
   c. Nilai tertinggi
   d. Nilai terendah.

D. Daftar Pustaka

1. Raharjo, budi dkk.2007."*Mudah Belajar Java*".Bandung:informatika.
2. Bima,ifnu.2011."Java Desktop".Singapore.
3. H.M Deitel, Java How to Program, Sixth Edition. Prentice Hall, August 2004

Class dan Objek pada Java

Pertemuan ke : III
Alokasi Waktu : 1,5 Jam
Kompetensi Dasar : Class dan Objek
Indikator : mampu mengimplementasikan sebuah class dan objek di java

A. Teori Pendukung

Sebuah Class mendefinisikan struktur (structure) dan tingkah laku (behaviour) sebuah obyek atau sekumpulan obyek. Class juga merupakan prototype yang mendefinisikan variabel-variabel dan method-method secara umum. Di dalam java ada aturan untuk pemberian sebuah nama class. Sebuah nama class harus diawali dengan huruf besar. Hal ini untuk membedakan antara class dan objek. Class didefinisikan dengan kata kunci class. Contoh sederhana dari deklarasi sebuah class :

```
class Mahasiswa {
    String nim;                 //deklarasi variabel atau atribut
    String nama;                //deklarasi variabel atau atribut
}
```

Sebuah Obyek merupakan instansiasi dari suatu class. Kalau kita analogikan, class itu sebuah cetakan sedangkan object itu adalah barang dari hasil cetakan. Class juga bisa dikatakan sebagai kategori, sedangkan object adalah sesuatu yang memenuhi syarat-syarat yang harus dipenuhi agar masuk dalam kategori tersebut. Jadi bisa dibilang satu class bisa mempunyai banyak object, setiap object mempunyai sifat yang sama persis seperti yang didefnisikan dalam class tersebut. Untuk pemberian nama sebuah objek, diawali dengan huruf kecil. Pembuatan objek untuk class Mahasiswa adalah sebagai berikut:

```
Mahasiswa mahasiswa;                    //deklarasi objek
mahasiswa = new Mahasiswa();            //instansiasi dari kelas Mahasiswa
Mahasiswa mahasiswa= new Mahasiswa();   //dijadikan satu
```

Setiap obyek mempunyai identitas yang unik, seperti halnya setiap orang mempunyai identitas yang unik. Contoh : Mahasiswa mempunyai Nim dimana nim seorang mahasiswa berbeda dengan mahasiswa yang lain. Di dalam class terdapat tiga bagian utama dari sebuah kelas yang mendeklarasikan kode-kode program java.

1. Konstruktor : digunakan untuk instansiasi objek
2. Variabel : merupakan atribut yang menyatakan keadaan dari kelas dan objek.
3. Metode (method) : berupa fungsi atau procedure.

Variabel dan metode dapat memiliki salah satu sifat berikut :

1. Private, tidak dapat dipanggil dari luar class yang bersangkutan
2. Protected, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya
3. Public, dapat dipanggil oleh siapa saja.

Dalam java terdapat dua buah metode(method) yaitu :

1. Fungsi, merupakan metode yang memiliki nilai balik jika metode tersebut dipanggil, cara pembuatan sebuah fungsi adalah dengan cara menentukan nilai baliknya, lalu membuat nama metodenya.
2. Prosedur, merupakan metode yang tidak memiliki nilai balik, cara pembuatan prosedur sama dengan fungsi namun bedanya, nilai baliknya menggunakan kata kunci void.

Kelas POJO atau Java Bean merupakan kelas dimana sebuah kelas memiliki atribut dan

memiliki metode getter dan setter. Dimana atributnya bersifat private dan metode getter dan setter nya bersifat public. Metode getter digunakan untuk mendapatkan nilai atribut tersebut, sedangkan metode setter digunakan untuk mengubah nilai atribut.
Berikut contoh 2.1 kelas pojo sederhana.

```
public class Mahasiswa {
    private String nim;
    private String nama;
    public String getNama() {               // method brp Fungsi
        return nama;
    }
    public void setNama(String nama) {      // method brp procedure
        this.nama = nama;
    }
    public String getNim() {                // method brp Fungsi
        return nim;
    }
    public void setNim(String nim) {        // method brp procedure
        this.nim = nim;
    }
}
```

Constructor adalah suatu method yang pertama kali di jalankan pada saat pembuatan suatu objek. Contructor mempunyunyai ciri-ciri sebagai berikut :

- mempunyai nama yang sama persis dengan nama class
- tidak mempunyai tipe return
- digunakan untuk menginstansiasi object
- hanya mempunyai access modifer, tidak ada keyword lain yang diletakkan sebelum nama method pada deklarasi constructor.

contoh 2.2 Contructor

```
class Mahasiswa {
    String nim;                    //deklarasi variabel
    String nama;
    public Manusia(){}              //default constructor
    public Manusia(String nim, string nama){   //constructor
perparameter
        this.nim = nim;
        this.nama= nama;
    }
}
```

Setelah pembahasan mengenai deklarasi kelas, kelas pojo dan konstruktor, maka perlu dicoba penggunaannya dari kelas pojo tersebut. Setiap membuat project, pasti sudah terbuat fungsi main. Berikut contoh code untuk memanggil method yang terdapat dalam kelas pojo melalui main.java.
Contoh 2.3 Main.java

Source code	Hasil

`public class Prak2 {`
`  public static void main(String[] args) {`
`  Mahasiswa m= new Mahasiswa();`
`  m.setNim("08018222");`
`  m.setNama("Rudi");`
`  System.out.println("Nim :"+ m.getNim());`
`  System.out.println("Nama :"+ m.getNama());`
`  }`
`}`
	Nim :08018222
	Nama :Rudi
	BUILD SUCCESSFUL (total time: 1 second)

B. Pre Test [Nmaks =100]

1. Jelaskan perbedaan antara class dengan objek? Berikan contohnya.
2. Tuliskan bagaimana cara membuat objek? Berikan contohnya
3. Jelaskan maksud dari metode setter dan getter? Berikan contohnya

C. Langkah Praktikum

1. Buatlah project baru dengan nama prak2.
2. Buatlah kelas pojo dengan nama PersegiPanjang seperti pada contoh.

D. Post Test [Nmaks =100]

Buatlah program untuk

1. membuat kelas Titik dengan atribut absis dan ordinat, metode/operasi yang ada menentukan sebuah titik, menghitung jarak dua titik, menentukan titik tengah dua titik
2. membuat kelas Waktu dengan atribut jam,menit, detik , metode/operasi yang ada menentukan sebuah jam , menghitung lama waktu

E. Daftar Pustaka

a. Raharjo, budi dkk.2007."*Mudah Belajar Java*".Bandung:informatika
b. H.M Deitel, 2004, *Java How to Program, Sixth Edition*. Prentice Hall

Konstruktor, Enkapsulasi & Hidding Information

Pertemuan ke : IV
Alokasi Waktu : 1,5 Jam
Kompetensi Dasar : Konstruktor, enkapsulapsi dan information hidding
Indikator : mampu membuat program dengan Konstruktor, Enkapsulasi & Hidding Information

A. Teori Pendukung

Konstruktor merupakan sebuah pseudo-method yang dibuat pada sebuat objek. Pada bahasa pemrograman java, konstruktor adalah instance method yang memiliki nama method sama dengan nama kelasnya. Sebuah konstruktor dipanggil menggunakan kata kunci new.
Berikut contoh konstruktor:

```
public class Bicycle{
    public Bicycle() {
           gear = 1
           speed = 0;
    }
}
```

konstruktor ini akan dipanggil saat melakukan instansiasi seperti contoh berikut:

```
Bicycle myBike = new Bicycle();
```

`new Bicycle()` akan membuat space di memory untuk objek dan menginisialisasinya. Sebuah konstruktor dapat dilakukan overloading terhadap konstruktor lain. Overloading merupakan suatu cara untuk menggunakan satu identifier untuk merujuk kepada multiple items dalam scope yang sama. Pada bahasa pemrograman Java, sebuah method dapat di overload tetapi tidak dengan variable dan operator.
Berikut contoh overloading konstruktor:

```
public class Bicycle{
    public Bicycle() {
           gear = 1
           speed = 0;
    }
    public Bicycle(int startSpeed, int startGear) {
           gear = startGear;
           speed = startSpeed;
    }
}
```

Untuk instansiasi dari class Bicycle dapat dilakukan seperti berikut:

```
Bicycle myBike = new Bicycle(30, 8);
```

Aturan dalam pembuatan kontruktor didefinisikan dalam dua aturan berikut:

1. Nama konstruktor harus sama dengan nama kelasnya
2. Konstruktor harus tidak memiliki explisit return type.

Terdapat dua tipe dari konstruktor:

1. Default Constructor yaitu konstruktor yang tidak memiliki argument
2. Construktor berparameter yaitu konstruktor yang memiliki argument

Ada beberapa perbedaan antara contructor dan method, yaitu:

Java Constructor	Java Method
Constructor digunakan untuk menginiasialisasi state pada sebuah objek.	Method digunakan untuk menunjukkan behavior dari object.

Constructor harus tidak mempunyai return type.	Method dapat memiliki return type.
Constructor dipanggil secara implisit.	Method dipanggil secara explisit
Compiler Java menyediakan default constructor jika tidak mendefinisikan constructor.	Method tidak disediakan oleh compiler
Nama constructor harus sama dengan nama Class	Nama method boleh atau tidak sama dengan nama class.

Akses modifier di Java

Ada dua macam tipe modifiers di java yaitu access modifiers dan non-access modifiers.

Access modifiers di java dikhususkan pada accessibility (scope) dari data member, method, constructor atau class.

Terdapat 4 macam java access modifiers:

1. private
2. default
3. protected
4. public

Terdapat banyak non-access modifiers seperti static, abstract, synchronized, native, volatile, transient dan lain-lain.

Berikut contoh penggunaan access modifier:

```
class Students{
    private int data=40;
    private void msg(){
           System.out.println("Hello java");
    }
}

public class Tester{
    public static void main(String args[]){
       Students obj = new Students();
       System.out.println(obj.data);//Compile Time Error
       obj.msg();//Compile Time Error
       }
}
```

Penggunaan access modifier

Access Modifier	Class	Package	Subclass	Umum
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Encapsulation di Java

Encapsulation di java adalah sebuah proses membungkus kode dan data secara bersama-sama dalam unit tunggal, sebagai contoh kapsul dalam resep racikan.

Keuntungan encapsulation di java yaitu dengan menyediakan method setter dan getter dan dapat membuat class menjadi read-only atau write-only. Ini menghasilkan control yang jelas terhadap akses data.

Berikut contoh penggunaan encapsulation:

Contoh 4.1 Class dengan nama file Student.java

```
package com.uad;
```

```
public class Student{
    private int age=20;
    private String nama;
    public String getNama(){
       return name;
    }

    public void setName(String name){
       this.name=name
    }
}
```

Contoh 4.2 Class dengan nama file Test.java

```
package com.uad;
class Test{
   public static void main(String[] args){
     Student s=new Student();
     s.setName("Romeo");
     System.out.println(s.age);
     System.out.println(s.getNama());
   }
}
```

B. Pre Test (Nmaks =100)

1. Tulis contoh baris program membuat konstruktor berparameter.
2. Tulis perbedaan accessor dan mutator? Berikan contoh sederhana.
3. Tuliskan perbedaan object class dan driver class?

C. Langkah Praktikum

1. Buatlah project baru dengan nama prak4.
2. Buatlah seperti pada contoh 4.1 dan 4.2.
3. Modifikasi contoh object class (contoh 4.1) dengan menambahkan konstruktor default dan konstruktor berparameter. Deklarasikan variabel bernilai tertentu di konstruktor default dan simpan nilai parameter pada sebuah variabel di konstruktor berparameter.
4. Lakukan instansiasi dengan menggunakan konstruktor default, kemudian tampilkan dilayar isi variabel dari objek hasil instansiasi tersebut.
5. Lakukan instansiasi dengan menggunakan konstruktor berparameter, kemudian tampilkan dilayar isi variabel dari objek hasil instansiasi tersebut.

D. Post Test [Nmaks = 100]

1. Buatlah sebuah driver class dengan nama Tester.java dan object class dengan nama Karyawan.java.
2. Buatlah sebuah konstruktor default yang telah berisi variabel gaji tetap dan berikan nilai tertentu.
3. Buatlah sebuah konstruktor berparameter dengan sebuah parameter dan simpan parameter tersebut pada variabel gaji tetap.
4. Buatlah tiga buah variabel dengan nama variabel: nomor pegawai, nama pegawai dan gaji bonus.
5. Lakukan pembuatan method mutator dan accessor untuk tiga variabel tersebut.
6. Buat sebuah method menghitung gaji untuk karyawan baru yang berasal dari gaji tetap saja.
7. Buat sebuah method menghitung gaji untuk karyawan lama yang berasal dari gaji tetap ditambahkan bonus.
8. Tampilkan di layar berupa informasi nomor pegawai, nama pegawai dan hitungan gaji untuk objek karyawan baru dan karyawan lama.

E. Daftar Pustaka

1. Raharjo, budi dkk.2007.”*Mudah Belajar Java*”.Bandung:informatika.
2. Bima,ifnu.2011.”Java Desktop”.Singapore.
3. H.M Deitel, Java How to Program, Sixth Edition. Prentice Hall, August 2004

Pewarisan(Inheritance)

Pertemuan ke : V
Alokasi Waktu : 1,5 Jam
Kompetensi Dasar : Pewarisan
Indikator : mampu mengimplementasikan konsep pewarisan di java

A. Teori Pendukung

Inheritance (pewarisan) merupakan ciri utama dari pemrograman berorientasi objek dimana sifat-sifat yang terdapat pada kelas induk akan dimiliki oleh kelas turunannya. Kelas induk yang diturunkan disebut dengan superclass. Adapun kelas baru hasil turunan disebut subclass. Pada proses penurunan kelas ini, kelas turunan akan mewarisi sifat-sifat yang terdapat pada kelas induknya. Selanjutnya, kelas turunan tersebut dapat memiliki sifat-sifat spesifik yang sebelumnya tidak di miliki oleh kelas induk. Misal manusia mempunyai nama dan alamat. Mahasiswa mempunyai nim, nama dan alamat. Maka kelas manusia mewarisi sifatnya ke kelas mahasiswa. Karena sifatnya terdapat kesamaan sifat.Berikut contohnya :

Contoh 3.1 Kelas Pojo Induk (Manusia.java)

```
public class Manusia {
    private String nama;
    private String alamat;

    public Manusia(){}

    public Manusia(String nama, String alamat) {
        this.nama = nama;
        this.alamat = alamat;
    }
    public String getAlamat() {
        return alamat;
    }
    public void setAlamat(String alamat) {
        this.alamat = alamat;
    }
    public String getNama() {
        return nama;
    }
    public void setNama(String nama) {
        this.nama = nama;
    }
}
```

Contoh 3.2 Kelas Pojo Anak (Mahasiswa.java)

```
public class Mahasiswa {
    private String nim;
    private String nama;
    private String alamat;

    public Mahasiswa() {
    }
    public Mahasiswa(String nim, String nama, String alamat) {
        this.nim = nim;
        this.nama = nama;
        this.alamat = alamat;
    }
    public String getAlamat() {
        return alamat;
    }
    public void setAlamat(String alamat) {
        this.alamat = alamat;
    }
    public String getNama() {
        return nama;
    }
    public void setNama(String nama) {
        this.nama = nama;
```

```
    }
    public String getNim() {
        return nim;
    }
    public void setNim(String nim) {
        this.nim = nim;
    }

}
```

Kode diatas bisa dibandingkan pada bagian atribut dan method. beberapa atribut dan metode yang sama, yaitu nama, alamat, setNama(), getNama(), setAlamat() dan getAlamat(). Artinya banyak terjadi duplikasi kode, oleh karena itu lebih baik kelas tersebut digabungkan menggunakan pewarisan, yaitu Manusia diturunkan menjadi Mahasiswa, karena semua atribut dan metode Manusia ada di Mahasiswa namun tidak semua atribut dan metode Mahasiswa ada di kelas Manusia.

Untuk mengatakan bahwa kelas X turunan dari kelas Y kita dapat menggunakan kata kunci extends. Dengan begitu kita hanya perlu mengubah kelas Mahasiswa menjadi seperti berikut.

Contoh 3.3 Penurunan Kelas (Mahasiswa.java)

```
public class Mahasiswa extends Manusia{
    private String nim;

    public Mahasiswa() {
    }
    public Mahasiswa(String nim) {
        this.nim = nim;
    }
    public String getNim() {
        return nim;
    }
    public void setNim(String nim) {
        this.nim = nim;
    }
}
```

Walaupun kelas Mahasiswa tidak memiliki atribut dan metode untuk nama dan alamat, namun sebenarnya Mahasiswa tersebut memilikinya, karena Mahasiswa merupakan turunan dari Manusia, sehingga seluruh sifat dari Manusia ada pada Mahasiswa.

Contoh 3.4 Main.java

Source Code	Hasil

`public class Main {`
`  public static void main(String[] args) {`
`  Mahasiswa m=new Mahasiswa();`
`  m.setNim("06018251");`
`  m.setNama("Tejo");`
`  m.setAlamat("Sleman");`
`  System.out.println("Nim :"+m.getNim());`
`  System.out.println("Nama :"+m.getNama());`
`  System.out.println("Alamat :"+m.getAlamat());`
`  }`
	Nim :06018251
	Nama :Tejo
	Alamat :Sleman
	BUILD SUCCESSFUL (total time: 1 second)

B. Pree Test (Nmaks =100)

1. Jelaskan apa yang dimaksud inheritance? Dan apa manfaat penggunaannya?
2. Tuliskan sintaks penulisan inheritance!
3. Berikan contoh deklarasi kelas inheritance! (selain dari yang ada modul)

C. Langkah Praktikum

1. Buat project baru dan beri nama project dengan Praktikum 3.
2. Silahkan mencoba contoh 3.1, 3.3 dan 3.4.

D. Post Test (Nmaks =100)

Buatlah project seperti perintah langkah-langkah praktikum dengan studi kasus membuat kelas lingkaran, kelas persegi panjang disertai metodenya yang merupakan turunan dari kelas titik.

E. Daftar Pustaka

1. Bima,ifnu.2011."Java Desktop".Singapore.
2. H.M Deitel, Java How to Program, Sixth Edition. Prentice Hall, August 2004

Polymorphisme

Pertemuan ke : **VI**
Alokasi Waktu : **1,5 Jam**
Kompetensi Dasar : **Polymorphisme**
Indikator : **mampu mengimplementasikan konsep polymorphisme di java**

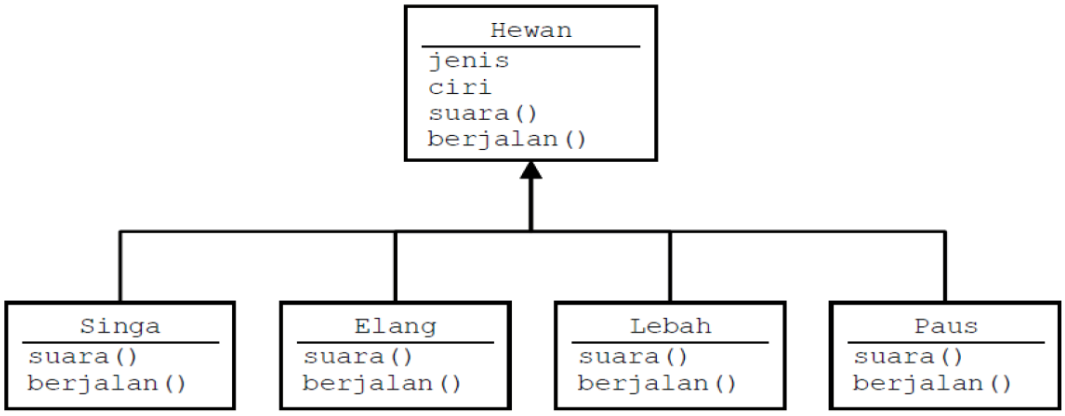
A. Teori Pendukung

Polimorphism berarti mempunyai banyak bentuk. Dua objek dikatakan sebagai polymorphic bila objek-objek itu mempunyai antarmuka-antarmuka yang identik namun mempunyai perilaku-perilaku berbeda. Polymorphism berupa satu nama tunggal yang menyatakan objek-objek kelas-kelas berbeda yang terhubung dengan suatu superkelas yang common di antara kelas-kelas itu. Dengan Polimorphism dapat dikenali dan dieksploitasi keserupaan-keserupaan diantara kelas-kelas berbeda

Ada 2 macam bentuk polymorphism, yaitu:

- Overloading: penggunaan satu nama untuk beberapa method yang berbeda dalam suatu class. Nama method yang sama dibedakan dari jumlah parameter dan tipe data parameternya.
- Overriding: mendeklarasikan sebuah method dengan nama dan parameter yang sama dari superclass-nya, kemudian dimodifikasi ulang (menumpuk/mendefinisi ulang).

Pada polymorphism, dikenal juga dynamic binding atau ikatan dinamis yang umumnya digunakan pada variabel bertipe class. Jika kita mempunyai variabel bertipe superclass, variabel ini dapat diisi dengan object superclass ataupun subclass-nya tanpa
melakukan perubahan tipe.



Contoh Polymorphisme

Pada gambar di atas suara superclass hewan bisa berbagai jenis terganting subclassnya.

Contoh pertama :

```
class EkspresiWajah{
    public String respons() {
        return("Perhatikan ekspresi wajah saya");
    }
}
class Gembira extends EkspresiWajah{
    public String respons() {
        return("ha ha ha…");
    }
}
class Sedih extends EkspresiWajah{
    public String respons() {
        return("hik hik ngeee ngeee ngeee");
    }
}
class Marah extends EkspresiWajah{
```

```
    public String respons() {
        return("Hai kurang ajar…!");
    }
}


class MainEkspresiWajah{
    public static void main(String args[]) {
       EkspresiWajah objEkspresi = new EkspresiWajah();
        Gembira objGembira = new Gembira();
        Sedih objSedih = new Sedih();
        Marah objMarah = new Marah();

        EkspresiWajah[] arrEkspresi = new EkspresiWajah[4];
        arrEkspresi[0] = objEkspresi;
        arrEkspresi[1] = objGembira;
        arrEkspresi[2] = objSedih;
        arrEkspresi[3] = objMarah;

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

Contoh kedua :

```
public class Employee {
    private String name;
    private double salary;
    protected Employee(String n, double s) {
        name = n;
        salary = s;
    }
    protected String getDetails() {
        return "Name : "+name+ "\nSalary : "+salary;
    }
    public void cetak() {
        System.out.println("coba di Employee");
    }
}
public class Manager extends Employee {
    private String department;
    public Manager(String nama, double salary, String dep) {
        super(nama, salary);
        department = dep;
    }
    public String getDepartment() {
        return department;
    }
    public String getDetails() {
        return super.getDetails()+ "\nDepartment : "+getDepartment();
    }
    public void cetak() {
        System.out.println("Coba di Manager");
    }
}

public class View {
    public static void main(String[] args) {
       Employee e = new Employee("Ali",1200000);
       Employee em = new Manager("Ali",1200000,"Informatika");
       System.out.println("Data employee :\n"+e.getDetails());
       System.out.println("Data manager :\n"+em.getDetails());
       em.cetak();
    }
}
```

Catatan :

Kalau method cetak() di kelas Employee dan kelas Manager ada, maka yang dijalankan adalah method milik kelas Manager. Prioritasnya adalah kelas Manager kemudian kelas Employee.

B. Pre Test [Nmaks =100]

1. Jelaskan apa yang dimaksud dengan polimorphisme dan jelaskan apa keuntungan menggunakannya?
2. Apa perbedaan antara overloading dan overriding dan berikan satu contoh masing-masing?

C. Langkah Praktikum

1. Buat project prak4 dengan menggunakan NetBeans
2. Kerjakan dua contoh diatas dan amati hasilnya
3. Buatlah sebuah kelas pojo dengan nama Mahasiswa seperti code dibawah ini.

```
public class Mahasiswa {
    private String nim;
    private String nama;

    public Mahasiswa() {
    }
    public Mahasiswa(String nim, String nama) {
        this.nim = nim;
        this.nama = nama;
    }
    public String getNama() {
        return nama;
    }
    public void setNama(String nama) {
        this.nama = nama;
    }
    public String getNim() {
        return nim;
    }
    public void setNim(String nim) {
        this.nim = nim;
    }
}
```

4. Buatlah sebuah kelas HM yang merupakan kelas turunan dari Mahasiswa dan buatlah metodhmetodhnya.
5. Buatlah sebuah kelas KelompokStudi yang merupakan kelas turunan dari Mahasiswa dan buatlah metodhmetodhnya.
6. Buatlah 1 objek untuk menginstantiasi kelas HM dan mengimplementasikan beberapa metodh yang telah Anda definisikan dalam kelas HM
7. Panggilllah objek yang telah Anda buat pada kelas KelompokStudi dan HM untuk bekerja sehingga tampil efek-efek polymorfisme

D. Post Test [Nmaks =100]

Kembangkan program dari ilustrasi gambar di atas dengan mengimplementasikam method-method yang ada kemudian buatlah testernya.

E. Daftar Pustaka

1. Khannedy, Eko K."Modul Pelatihan Java Dasar".Bandung
2. Anonim, 2009, Praktikum Pemrograman Berorientasi Obyek Politeknik Telkom Bandung

Implementasi Relasi Diagram Kelas

Pertemuan ke : **VII**
Alokasi Waktu : **1,5 Jam**
Kompetensi Dasar : **Implementasi dari relasi pada diagram kelas**
Indikator : **mengimplementasikan relasi asosiasi, agregasi dan komposisi pada class diagram**

A. Teori Pendukung

Pengantar tentang Array

Untuk mengawali hubungan antar kelas perlu mereview lagi tentang array (larik). Array digunakan untuk menggolongkan data atau obyek yang bertipe sejenis. Array dideklarasikan dengan tipe apa saja, baik itu yang bertipe data primitif maupun obyek. Berikut contoh deklarasi Array :

```
char c[];
titik t[]; // t adalah sebuah kelas
```

Dalam bahasa pemrograman Java, Array merupakan sebuah obyek meskipun ia terdiri dari elemen yang bertipe data primitif. Seperti halnya kelas yang lain, ketika mendeklarasikan Array belum dibentuk sebuah obyek Array. Deklarasi Array hanya membuat sebuah referensi yang dapat digunakan untuk mengacu ke sebuah obyek Array. Besarnya memori yang digunakan oleh elemen-elemen Array akan dialokasikan secara dinamis dengan menggunakan pernyataan new atau dengan array initializer.

Contoh deklarasi Array di atas, dengan menggunakan kurung siku setelah nama variabel, merupakan standar penulisan Array dalam C, C++ dan Java. Format demikian agak sulit untuk dibaca. Oleh karenanya, bahasa Java menggunakan bentuk deklarasi Array alternatif dengan menggunakan kurung siku di sebelah kiri seperti dalam contoh berikut :

```
char [] c;
titik [] t; // titik adalah sebuah kelas
```

Sebuah deklarasi Array dapat ditulis sebagai pendeklarasian tipe Array di bagian kiri dan nama variabel di bagian kanan. Kedua bentuk deklarasi Array di atas dapat digunakan, tetapi sebaiknya konsisten dalam menggunakan satu bentuk saja. Dalam deklarasi Array, tidak ditentukan ukuran aktual dari Array. Ketika mendeklarasikan Array dengan kurung siku yang berada di sebelah kiri nama variabel, kurung siku tersebut berlaku bagi semua variabel yang berada di sebelah kanannya.

Membuat Array

Array dibuat dengan menggunakan keyword new. Contoh di bawah ini membuat sebuah Array dengan tipe primitif char:

```
c = new char[26];
```

Indeks Array dimulai dengan angka 0. Batasan legal untuk nilai indeks ini adalah dari nol hingga jumlah elemen array-1. Apabila program berusaha mengakses Array di luar batas yang legal, maka akan muncul runtime exception.

Untuk membuat Array dengan elemen obyek, gunakan cara penulisan berikut

```
t = new titik[5];
```

Pernyataan tersebut tidak membuat 5 obyek Titik. Untuk membuat obyek Titik, gunakan pernyataan berikut :

```
t[0] = new titik(1, 5);
t[1] = new titik (2, 4);
```

Menginisialisasi Array

Java memiliki cara singkat untuk membuat sebuah obyek Array dengan elemen-elemennya memiliki nilai awal :

```
String names[] = {"Anni","Syauqi","Savitri"};
```

Kode di atas adalah sama dengan kode berikut ini:

```
String names[];
names[0] = "Anni";
names[1] = "Syauqi";
```

```
names[2] = "Savitri";
```

Array Multidimensi

Dalam Java, dapat dibuat Array dari Array sehingga dinamai Array multidimensi. Contoh berikut ini memperlihatkan cara membuat Array dua dimensi :

```
int twoDim [][] = new int[4][];
twoDim[0] = new int[5];
twoDim[1] = new int[5];
```

Batasan Array

Dalam bahasa Java, indeks Array dimulai dengan angka nol. Jumlah elemen di dalam Array disimpan sebagai bagian dari obyek Array di dalam atribut length. Atribut ini dapat digunakan untuk melakukan proses iterasi pada Array seperti dalam contoh berikut :

```
int list[] = new int[10];
for(int i = 0; i < list.length; i++) {
System.out.println(list[i]);
}
```

Dengan menggunakan atribut length, pemeliharaan kode program menjadi mudah karena program tidak perlu mengetahui jumlah eleman Array pada saat kompilasi.

Manipulasi Array

Mengubah Ukuran Array

Setelah membuat obyek Array, ukuran Array tersebut tidak dapat diubah. namun demikian, dapat digunakan variabel referensi yang sama untuk menunjuk ke sebuah obyek Array baru seperti dalam contoh di bawah ini :

```
int myArray[] = new int[6];
myArray = new int[10];
```

Menyalin Array

Bahasa Java menyediakan method khusus untuk menyalin Array, yaitu dengan menggunakan method arrayCopy() dari kelas System seperti dalam contoh berikut :

```
// array semula
int myArray[] = {1, 2, 3, 4, 5, 6};
// array engan elemen yang lebih banyak
int hold[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
//menyalin semua elemen myArray ke hold
// dimulai dengan indeks ke 0
System.arraycopy(myArray, 0, hold, 0,myArray.length);
```

Setelah proses pengkopian di atas, maka isi dari Array hold adalah 1, 2, 3, 4, 5, 6, 4, 3, 2, 1.

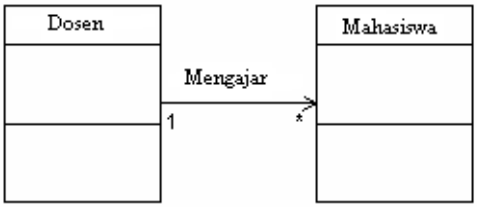
Hubungan Antar Kelas

Dalam Object Oriented Programming, kelas-kelas yang terbentuk dapat memiliki hubungan satu dengan yang lainnya, sesuai dengan kondisi dari kelas-kelas yang bersangkutan. Ada beberapa jenis hubungan yang dapat terjadi antara kelas yang satu dengan kelas yang lainnya, antara lain:

1. Asosiasi
2. Agregasi
3. Komposisi
4. Generalisasi (terkait dengan pewarisan)
5. Spesialisasi (terkait dengan pewarisan)

1. Asosiasi

Asosiasi merupakan hubungan antara dua kelas di yang merupakan hubungan struktural yang menggambarkan himpunan link antar obyek. Contoh dari hubungan asosiasi ini adalah:



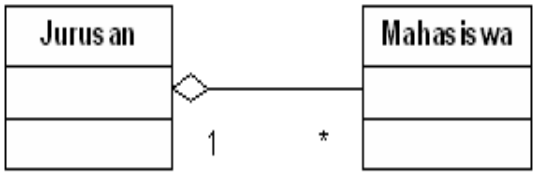
Bentuk implementasi dari diagram kelas tersebut di Java adalah sebagai berikut:

```
class Mahasiswa {
	private String nim;
	private String nama;

	public Mahasiswa(String nim, String nama)
	{
		this.nim=nim;
		this.nama=nama;
	}
	public void setnama (String nama) {
	this.nama = nama;
	}
	public void setnim (String nim) {
	this.nim = nim;
	}
	public String getNim() {
	return this.nim;
	}
	public String getNama () {
	return this.nama;
	}
}

class Dosen {
	private String Kddosen;
	private String[] nimMHS=new String[5];
	private int jmlMahasiswa=0;

	public Dosen(String kode)
	{
		this.Kddosen=kode;
	}
	public void setKddosen (String Kddosen) {
		this.Kddosen = Kddosen;
	}
	public void setNimMahasiswa (String nim) {
		nimMHS[jmlMahasiswa]=nim;
		jmlMahasiswa++;
	}
	public int getJmlMahasiswa () {
	return this.jmlMahasiswa;
	}
	public String getKddosen () {
	return this.Kddosen;
	}
	public void daftarMahasiswa() {
		System.out.println("Kode Dosen "+Kddosen);
		System.out.println("Daftar Mahasiswa");
		for (int i=0;i<jmlMahasiswa;i++)
		{
			System.out.println(nimMHS[i]);
		}

	}
}

//kelas DriverMahasiswaDosen
```

```
class DriverMahasiswaDosen
{
	public static void main(String[] args)
	{
/*kode dibawah ini merupakan implementasi dari hubungan asosiasi antar kelas*/

	Mahasiswa mhs1 = new Mahasiswa("30107998","Abdul Kadir");
	Mahasiswa mhs2 = new Mahasiswa("30107999","Asep Sumarta");
		Dosen dsn = new Dosen("SKS");
		dsn.setNimMahasiswa(mhs1.getNim());
		dsn.setNimMahasiswa(mhs2.getNim());
		dsn.daftarMahasiswa();
	}
}
```

2. Agregasi

Agregasi merupakan hubungan antara dua kelas di mana kelas yang satu merupakan bagian dari kelas yang lain namun kedua kelas ini dapat berdiri sendiri-sendiri. Agregasi sering juga disebut relasi “part of” atau relasi “whole-part”. Contoh hubungan agregasi ini adalah:



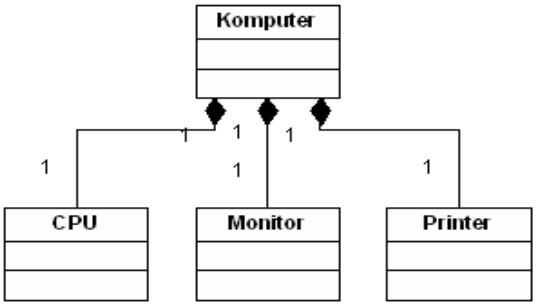
Implementasi dari diagram kelas tersebut dalam Java adalah sebagai berikut:

```
//mahasiswa.java
public class mahasiswa {
	private String NIM, Nama;
public mahasiswa(String no, String nm) {
		this.NIM = no;
		this.Nama = nm;
	}
public String GetNIM() {
		return (NIM);
	}
	public String GetNama() {
		return (Nama);
	}
}

//jurusan.java
public class Jurusan {
     private String KodeJurusan, NamaJurusan;
     /*baris dibawah ini merupakan implementasi dari hubungan agregasi
antar kelas*/
     private Mahasiswa[] Daftar = new Mahasiswa[10];
public Jurusan(String kode, String nama) {
		this.KodeJurusan = kode;
		this.NamaJurusan = nama;
	}
	private static int JmlMhs = 0;
	public void AddMahasiswa(Mahasiswa m) {
		this.Daftar[JmlMhs] = m;
		this.JmlMhs++;
	}
public void DisplayMahasiswa() {
	int i;
	System.out.println("Kode Jurusan : "+this.KodeJurusan);
	System.out.println("Nama Jurusan : "+this.NamaJurusan);
	System.out.println("Daftar Mahasiswa :");
	for (i=0;i<JmlMhs;i++)
System.out.println(Daftar[i].GetNIM()+" "+Daftar[i].GetNama());
	}
}
```

3. Komposisi

Komposisi merupakan bentuk khusus dari agregasi di mana kelas yang menjadi part (bagian)

baru dapat diciptakan setelah kelas yang menjadi whole (seluruhnya) dibuat dan ketika kelas yang menjadi whole dimusnahkan, maka kelas yang menjadi part ikut musnah. Contoh hubungan komposisi adalah sebagai berikut:



Implementasi dari diagram kelas tersebut dalam Java adalah sebagai berikut:

```
//CPU.java
public class CPU {
       private String Merk;
       private int Kecepatan;
public CPU(String m, int k) {
       this.Merk = m;
       this.Kecepatan = k;
}
public void DisplaySpecCPU() {
       System.out.println(this.Merk + ", " + this.Kecepatan);
              }
       }

//Monitor.java
public class Monitor {
       private String Merk;
public Monitor(String m) {
       this.Merk = m;
              }
public void DisplaySpecMonitor() {
       System.out.println(this.Merk);
              }
       }

//Printer.java
public class Printer {
       private String Merk, Type;
public Printer(String m, String t) {
       this.Merk = m;
       this.Type = t;
              }
public void DisplaySpecPrinter() {
       System.out.println(this.Merk + ", " + this.Type);
              }
       }

//Komputer.java
public class Komputer {
       private String Kode;
       private long Harga;
       private CPU Proc;
       private Monitor Mon;
       private Printer Prn ;

public Komputer(String k, long h) {
       this.Kode = k;
       this.Harga = h;
              if (k == "PC-01") {
/*baris dibawah ini merupakan implementasi dari hubungan komposisi antar kelas*/
       Proc = new CPU("Pentium IV", 500);
       Mon = new Monitor("Sony Multiscan 15sf");
       Prn = new Printer("Canon BJC-210SP", "Color");
            }
       }
public void DisplaySpec() {
```

```
System.out.println("Kode      : " + this.Kode);
        System.out.print("Processor: ");
        Proc.DisplaySpecCPU();
        System.out.print("Monitor   : ");
        Mon.DisplaySpecMonitor();
System.out.print("Printer   : ");
        Prn.DisplaySpecPrinter();
        System.out.println("Harga : Rp. " + this.Harga);
        }
}
```

B. Pre Test (Nmaks =100)

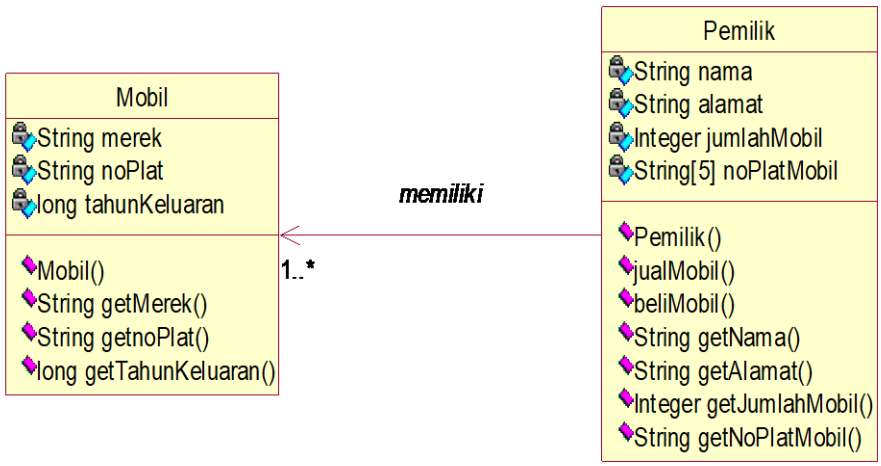
1. Jelaskan apa yang dimaksud dengan array ?
2. Jelaskan apa yang dimaksud dengan istilah berikut dan beri contoh yang lain selain yang ada di modul :
   a. asosiasi
   b. agregasi
   c. komposisi

C. Langkah Praktikum

Buatlah Tester untuk contoh-contoh yang diberikan dalam modul ini

D. Post Test (Nmaks =100)

1. Asosiasi



Diketahui class diagram di atas menggambarkan hubungan antara kelas Pemilik dan kelas Mobil.
Pada kelas mobil:

- Terdapat atribut merek untuk menyimpan merek mobil, noPlat untuk menyimpan nomor plat mobil, dan tahunKeluaran untuk menyimpan tahun keluaran mobil
- Terdapat konstruktor mobil untuk mengeset nilai merek, noPlat, dan tahunKeluaran mobil

Pada kelas pemilik:

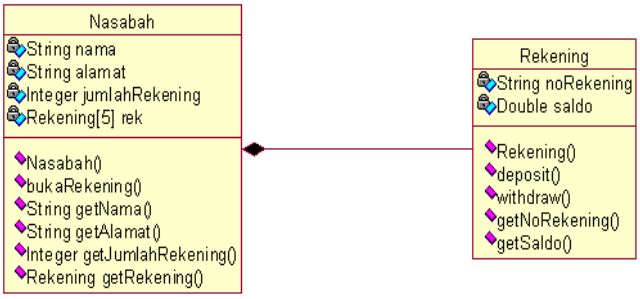
- Terdapat atribut nama untuk menyimpan nama pemilik, alamat untuk menyimpan alamat pemilik, jumlahMobil untuk menyimpan jumlah mobil pemilik, dan noPlatMobil untuk menyimpan nomor plat mobil pemilik
- Seorang pemilik maksimal memiliki 5 mobil
- Terdapat konstruktor pemilik untuk mengeset nama dan alamat pemilik
- Terdapat method beliMobil untuk menambahkan nomor plat mobil baru seorang pemilik
- Terdapat method jualMobil untuk mengurangi nomor plat mobil seorang pemilik

Tugas:

- Buatlah kode Java untuk mengimplementasikan class diagram di atas!

- Buatlah class tester untuk menampilkan data seorang pemilik dan mobilnya dengan jumlah mobilnya 5! (data diasumsikan sendiri)

2. Agregasi



Diketahui class diagram di atas menggambarkan hubungan antara kelas nasabah dan kelas rekening. Seorang nasabah bisa membuat maksimal 5 rekening pada sebuah bank.
Pada kelas nasabah:

- Terdapat atribut nama untuk menyimpan nama nasabah, alamat untuk menyimpan alamat nasabah, jumlahRekening untuk menyimpan jumlah rekening nasabah, dan rek untuk menyimpan data rekening nasabah
- Terdapat konstruktor nasabah untuk mengeset nama dan alamat nasabah
- Terdapat method bukaRekening untuk mengeset rekening baru

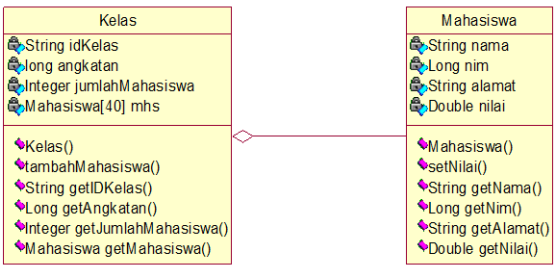
Pada kelas rekening:

- Terdapat atribut noRekening untuk menyimpan nomor rekening dan saldo untuk menyimpan nilai saldo
- Terdapat method deposit untuk menambahkan nilai saldo
- Terdapat method withdraw untuk mengurangi nilai saldo

Tugas:

- Buatlah code Java untuk mengimplementasikan class diagram di atas!
- Buatlah class tester untuk menampilkan data seorang nasabah dan rekeningnya! (data diasumsikan sendiri)

3. Komposisi



Diketahui class diagram di atas menggambarkan hubungan antara class Kelas dan class Mahasiswa. Satu kelas maksimal memiliki 40 orang mahasiswa.
Pada class Kelas:

- Terdapat atribut idKelas untuk menyimpan ID Kelas, angkatan untuk menyimpan angkatan, jumlahMahasiswa untuk menyimpan jumlah mahasiswa pada kelas tersebut, dan juga mhs untuk menyimpan data mahasiswa sebuah kelas
- Terdapat konstruktor kelas untuk mengeset idKelas dan angkatan
- Terdapat method tambahMahasiswa untuk menambahkan data mahasiswa suatu kelas

Tugas:

- Buatlah code Java untuk mengimplementasikan class diagram dan buatlah class driver untuk menampilkan data kelas dan mahasiswanya! (data diasumsikan sendiri

E. Daftar Pustaka

1. Khannedy, Eko K.”Modul Pelatihan Java Dasar”.Bandung
2. Anonim, 2009, Praktikum Pemrograman Berorientasi Obyek Politeknik Telkom Bandung

Abstract Class

Pertemuan ke : VIII
Alokasi Waktu : 1,5 Jam
Kompetensi Dasar : Abstract Class
Indikator : mampu mengimplementasikan konsep abstract Class di java

A. Teori Pendukung

Abstract Class merupakan kelas yang berada pada posisi tertinggi dalam sebuah hierarki kelas. Sesuai dengan namanya, abstract class dapat didefinisikan pada class itu sendiri. Berikut adalah cara mendeklarasikan abstract class:

```
abstract class NamaKelas{
       //isi abstract class
}
Contoh
abstract class Manusia{
       //isi abstract class
}
```

Didalam abstract class dapat juga diberi abstract method (Optional). Penggunaan abstract method tidak diperlukan statement dalam method tersebut. Berikut cara mendeklarasikan abstract method pada abstract class:

```
abstract class NamaKelas{
       //untuk method berjenis prosedur
[access_modifier] abstract void [namaMethod]();

       //untuk method berjenis fungsi
[access_modifier] abstract [tipe_data] [namaMethod]();
}
Contoh:
abstract class Manusia{
       //untuk method berjenis prosedur
public abstract void setNama();

       //untuk method berjenis fungsi
public abstract String getNama();
}
```

Catatan:

1. Apabila dalam abstract class terdapat abstract method dan kelas tersebut diturunkan ke kelas turunannya, maka method tersebut harus dideklarasikan ulang (overriding method) dengan diberi statement pada isi methodnya.
2. Apabila class tersebut merupakan abstract class, maka class tersebut bisa terdapat abstract method atau tidak (optional). Sedangkan apabila kelas tersebut, terdapat abstract method, maka kelas tersebut wajib berbentuk abstract class.

Keyword "final"

Keyword "final" digunakan untuk mencegah suatu class diturunkan atau suatu method dilakukan pendeklarasian ulang (overriding method). Berikut adalah cara mendeklarasikan *final* dalam class:

```
final class NamaKelas{
//isi class
}
```

Contoh

```
final class Manusia{
       //isi class
}
```

Sedangkan cara mendeklarasikan “final” pada method adalah sebagai berikut:

```
final class NamaKelas{
//untuk method berjenis prosedur
}
```

Contoh

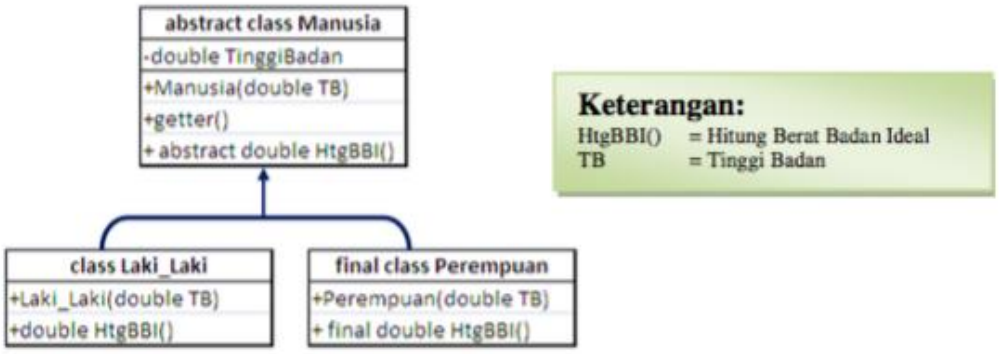
```
final class Manusia{
       //method berjenis prosedur
       public final void cetakBeratBadanIdeal(){
              //isi method
       }
       //method berjenis prosedur
       public final double hitungBeratBadanIdeal(){
              //isi method
              return 0;
       }

}
```

B. Pre Test [Nmaks =100]

1. Jelaskan apa yang dimaksud dengan abstract class dan jelaskan apa keuntungan menggunakannya?
2. Jelaskan apa yang dimaksud dengan keyword final dan jelaskan cara menggunakannya?

C. Langkah Praktikum

1. Buat project prak8 dengan menggunakan netbeans
2. Perhatikan class diagram berikut:



Untuk menghitung berat badan ideal sesuai dengan class adalah:

a. Laki-laki = (tinggi badan(cm)-100) kg X 90%
b. Perempuan = (tinggi badan(cm)-100) kg X 80%

3. Buatlah sebuah kelas pojo dengan nama Manusia seperti code dibawah ini.

```
public class Manusia {
    private double tinggiBadan;

    public Manusia () {
    }
    public Manusia (double tb) {
        this.tinggiBadan = tb;
    }
    public double getTinggi() {
        return tinggiBadan;
    }
    public abstract doule hitungBeratBadanIdeal();
}
```

4. Buatlah sebuah kelas Lakilaki seperti code dibawah ini.

```
public class Lakilaki extends Manusia{

    public Lakilaki (double tinggiBadan) {
        super(tinggiBadan);
    }
    public doule hitungBeratBadanIdeal(){
        return (super.getTinggiBadan()-100)*0.9;
    }
}
```

5. Buatlah sebuah kelas Perempuan seperti code dibawah ini.

```
final public class Perempuan extends Manusia {
    public Perempuan (double tinggiBadan) {
        super(tinggiBadan);
    }
    public final double hitungBeratBadanIdeal(){
        return (super.getTinggiBadan()-100)*0.8;
}
```

Keyword “final” pada kelas perempuan digunakan untuk mencegah pembuatan kelas baru dari kelas turunan perempuan. Sedangkan keyword “final” pada method digunakan untuk mencegah pendeklarasian (ulang method) pada kelas turunannya.

D. Post Test [Nmaks =100]

1. Buatlah kelas tester dari contoh diatas.
2. Lakukan instansiasi objek untuk objek manusia kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek manusia tersebut.
3. Lakukan instansiasi objek untuk objek laki-laki kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek laki-laki tersebut.
4. Lakukan instansiasi objek untuk objek perempuan kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek perempuan tersebut.
5. Buat kelas baru dengan nama PerempuanDewasa sebagai turunan dari kelas perempuan dan lakukan overriding pada method hitungBeratBadanIdeal dengan rumus menghitung berat badan ideal adalah (tinggi badan(cm)-100) kg X 85%
6. Lakukan instansiasi objek untuk objek perempuandewasa kemudian berikan nilai untuk tinggi badannya dan hitung tinggi badan ideal objek perempuandewasa tersebut.

E. Daftar Pustaka

1. Khannedy, Eko K.”Modul Pelatihan Java Dasar”.Bandung
2. Anonim, 2009, Praktikum Pemrograman Berorientasi Obyek Politeknik Telkom Bandung

Interface

Pertemuan ke : **IX**
Alokasi Waktu : **1,5 Jam**
Kompetensi Dasar : **Interface**
Indikator : **mampu mengimplementasikan konsep polymorphisme di java**

A. Teori Pendukung

Interface adalah kumpulan method yang hanya memuat deklarasi dan struktur method tanpa detail implementasinya. Cara mendeklarasikan interface adalah sebagai berikut:.

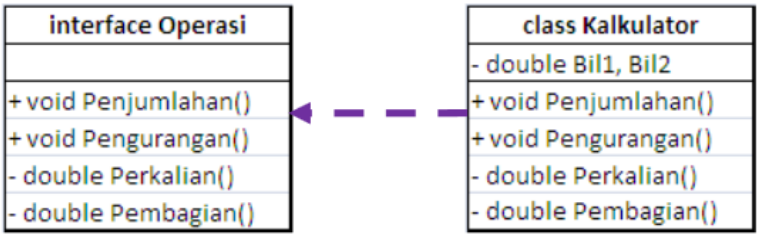
```
interface Nama_Interface
{
    //deklarasi variabel dan/atau method
}
```

Contoh:

```
interface Operasi
{
    //deklarasi variabel dan/atau method
    public void Penjumlahan();
    public void Pengurangan();
    public double Perkalian();
    public double Pembagian();
}
```

Pada contoh di atas, method yang dideklarasikan pada interface Operasi tidak terdapat statement apapun, baik itu rumus atau hanya sebuah nilai balik di dalamnya. Hal ini dikarenakan interface hanyalah sebuah berisi kumpulan konstanta maupun method tanpa menyertakan/menuliskan body methodnya. Perlu diketahui pula, bahwa an interface is not a class and classes can only implement interfaces (sebuah interface bukanlah sebuah kelas dan kelas hanya bisa mengimplementasi interface). Sehingga jangan anda menganggap bahwa interface adalah super class dimana memiliki kelas turunan.

Penggunaan (implementasi) interface dalam sebuah kelas dapat anda lihat melalui skema OOP di bawah ini:



Anak panah putus-putus merupakan gambaran bahwa class Kalkulator merupakan implementasi dari interfaces Operasi, dimana method-method yang terdapat pada interface Operasi harus dideklarasi ulang (overriding method) pada kelas Kalkulator. Interface dilambangkan dengan anak panah dengan garis putus-putus, sedangkan inheritance dilambangkan dengan anak panah dengan garis lurus (→).

Dalam pemrograman OOP, implementasi interfaces menggunakan keyword implements. Berikut adalah cara mengimplementasikan interface ke dalam pemrograman Java:

```
class Nama_Kelas implements Nama_Interface
{
    //isi kelas
}
```

Contoh implementasi:

```
class Kalkulator implements Operasi
{
    public void Penjumlahan()
    {
        //isi method Penjumlahan()
    };

    public void Pengurangan()
    {
        //isi method Pengurangan()
    };

    public double Perkalian()
    {
        //isi method Perkalian()
        return 0;
    };

    public double Pembagian()
    {
        //isi method Pembagian()
        return 0;
    };
};
```

B. Pre Test [Nmaks =100]

1. Jelaskan apa yang dimaksud dengan class interface dan apa kegunaannya!
2. Apa perbedaan method yang ada pada kelas interface dengan method-method lainnya seperti method setter dan getter?

C. Langkah Praktikum

1. Buat project prak9 dengan menggunakan NetBeans
2. Kerjakan contoh di bawah ini dan amati hasilnya!

```
/*interface berisi abstract method atau method header*/
interface MyComparable {
boolean greaterThan(Object obj); boolean lessThan(Object obj);
boolean equal(Object obj);
}
/*interface berisi konstanta*/
interface Constants {
int min = 1000;
int max = 9999;
}
/*class mengimplementasikan dua interface*/
class FourDigitsNumber implements Constants, MyComparable {
private int value;
public FourDigitsNumber(int initValue) {
/*latihan 1: atur agar nilai yang diinputkan dalam constructor hanya
berada di antara min – max
*/
}
/*latihan 2: tambahkan method get untuk mengakses value*/
/*overriding method greaterThan*/
public boolean greaterThan(Object obj) {
/*casting from superclass to subclass*/ FourDigitsNumber number =
(FourDigitsNumber)obj; return ( value > number.getValue() );
}
```

```
/*latihan 3: lengkapi overriding method interface*/
}
/*Contoh penggunaan class*/
class ExInterface {
public static void main(String [] args) { FourDigitsNumber number1 =
new FourDigitsNumber(700); FourDigitsNumber number2 = new
FourDigitsNumber(1700); FourDigitsNumber number3 = new
FourDigitsNumber(70000); System.out.println(number1.getValue());
System.out.println(number2.getValue());
System.out.println(number3.getValue());
System.out.println(number1.greaterThan(number2));
System.out.println(number1.lessThan(number2));
System.out.println(number1.equal(number2));
}}
```

3. Kerjakan soal latihan yang ada dalam contoh di atas!

D. Post Test [Nmaks =100]

Buatlah program baru yang menggambarkan implementasi interface Operasi di bawah ini, ke dalam class Calculator agar dapat mengoperasikan 2 bilangan Bil1 dan Bil2 menggunakan method yang ada dalam interface Operasi!

```
interface Operasi
{
    //deklarasi variabel dan/atau method
    public void Penjumlahan();
    public void Pengurangan();
    public double Perkalian();
    public double Pembagian();
}
```

Input Output dan Exception Handling

Pertemuan ke : **X**
Alokasi Waktu : **1,5 Jam**
Kompetensi Dasar : **Input Output dan Exception Handling**
Indikator : **mampu mengimplementasikan konsep Input Output dan Exception Handling di java**

A. Teori Pendukung

Pada program-program yang membutuhkan data-data eksternal, maka diperlukan suatu proses input dan ouput (I/O), dimana pada Java dukungan proses I/O ini sudah disediakan dalam paket java.io. Di dalam paket tersebut tersimpan banyak kelas dan interface siap pakai yang akan memudahkan programmer dalam pengambilan dan penyimpanan informasi dari/ke media lain (misalnya file).

Program Java melakukan proses I/O melalui stream, yaitu sebuah abstraksi yang dapat memberikan atau mendapatkan informasi. Stream dapat dihubungkan dengan peralatan fisik yang terdapat dalam sistem I/O Java, seperti keyboard, file, layar console, soket jaringan, dan lainnya. Walaupun dihubungkan dengan peralatan fisik yang berbeda, cara kerja stream selalu sama, sehingga kode program yang ditulis juga sama untuk masing-masing peralatan fisik. Misalnya, untuk melakukan penulisan sebuah teks ke layar console maupun ke dalam file, maka dapat digunakan kelas dan method yang sama.

Stream ada dua jenis, yaitu stream byte dan stream karakter. Stream byte digunakan untuk memberikan atau menyimpan informasi data dalam bentuk byte, misalnya untuk menulis dan membaca file biner. Sedangkan stream karakter pada proses I/O yang melibatkan data-data berbentuk karakter, misalnya proses baca/tulis ke suatu file teks, dengan menggunakan karakter Unicode.

Pendefinisian stream dilakukan dengan menggunakan empat kelas abstrak, yaitu InputStream dan OutputStream, sebagai superclass untuk kelas-kelas dalam kategori stream byte, dan kelas abstrak Reader dan Writer untuk kategori stream karakter. Melalui proses pewarisan (inheritance), semua kelas yang diturunkan dari InputStream maupun Reader akan memiliki method read(), yang digunakan dalam proses pembacaan data. Adapun untuk proses penulisan data digunakan method write() dalam semua kelas yang diturunkan dari OutputStream maupun Writer.

Exception merupakan runtine error. Exception terjadi jika program berjalan tidak sesuai dengan yang seharusnya. Artinya, alur program berubah dan aplikasi berhenti dalam keadaan error. Eksepsi dapat terjadi karena tidak ada antisipasi terhadap kesalahan penggunaan program.

Misalnya program didesain untu menghitung angka yang dimasukkan. Jika pengguna memasukkan karakter yang tidak dapat dihitung, misalnya huruf, maka akan terjadi exception

Terdapat dua jenis exception dalam java, yaitu:

1. Checked Exception.
   Merupakan eksepsi yang harus ditangkap atau dilempat secara eksplisit. Jika tidak ada metode yang dikonfigurasi untuk menangkap eksepsi ini, maka kode program tidak dapat dikompilasi
2. Unchecked Exception
   Merupakan eksepsi yang dapat dibuat agar program benar-benar dapat diandalkan. Namun tidak harus ada

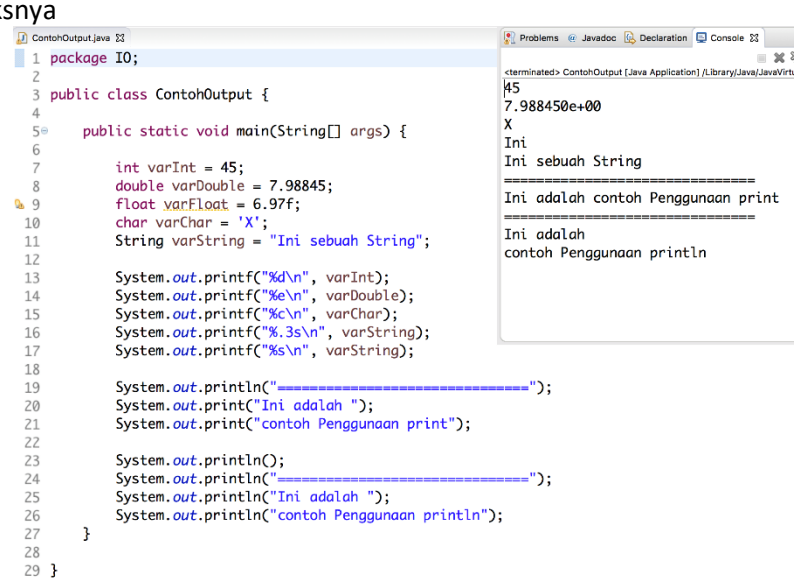
B. Pre Test [Nmaks =100]

1. Jelaskan kegunaan variabel standar stream yaitu: System.id, System.out, dan System.err?
2. Jelaskan perbedaan penggunaan metode menampilkan output dengan print(), println() dan printf()?
3. Tulis Sintaks exception handling?

C. Langkah Praktikum

1. Buat project prak10 dengan menggunakan netbeans

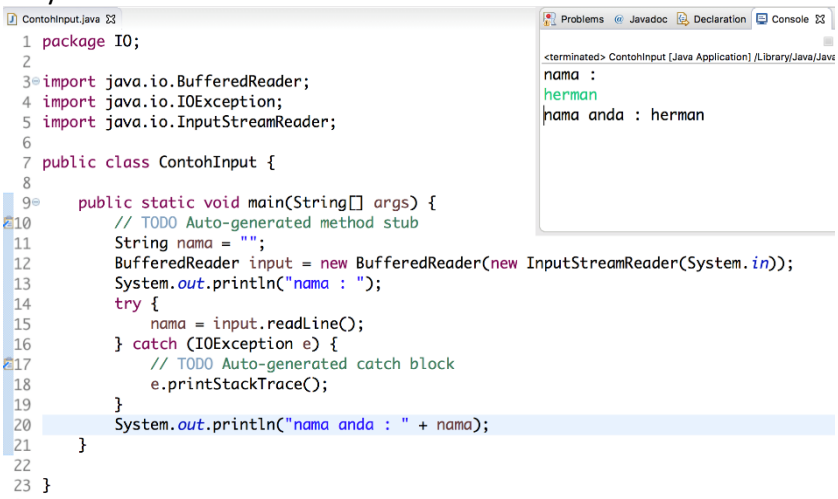
2. Buatlah sebuah kelas pojo dengan nama ClassOutput seperti code dibawah ini dan amati manfaat sintaksnya



```
ContohOutput.java
1  package IO;
2
3  public class ContohOutput {
4
5      public static void main(String[] args) {
6
7          int varInt = 45;
8          double varDouble = 7.98845;
9          float varFloat = 6.97f;
10         char varChar = 'X';
11         String varString = "Ini sebuah String";
12
13         System.out.printf("%d\n", varInt);
14         System.out.printf("%e\n", varDouble);
15         System.out.printf("%c\n", varChar);
16         System.out.printf("%.3s\n", varString);
17         System.out.printf("%s\n", varString);
18
19         System.out.println("==============================");
20         System.out.print("Ini adalah ");
21         System.out.print("contoh Penggunaan print");
22
23         System.out.println();
24         System.out.println("==============================");
25         System.out.println("Ini adalah ");
26         System.out.println("contoh Penggunaan println");
27     }
28
29 }
```

```
Problems  Javadoc  Declaration  Console
<terminated> ContohOutput [Java Application] /Library/Java/JavaVirtu
45
7.988450e+00
X
Ini
Ini sebuah String
==============================
Ini adalah contoh Penggunaan print
==============================
Ini adalah
contoh Penggunaan println
```

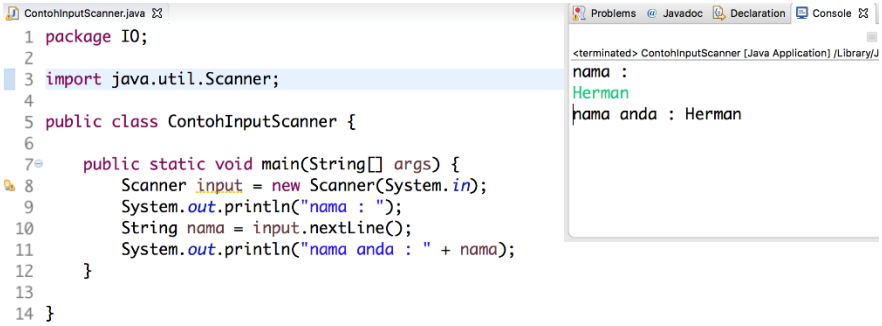
3. Buatlah sebuah kelas pojo dengan nama ClassInput seperti code dibawah ini dan amati manfaat sintaksnya



```
ContohInput.java
1  package IO;
2
3  import java.io.BufferedReader;
4  import java.io.IOException;
5  import java.io.InputStreamReader;
6
7  public class ContohInput {
8
9      public static void main(String[] args) {
10         // TODO Auto-generated method stub
11         String nama = "";
12         BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
13         System.out.println("nama : ");
14         try {
15             nama = input.readLine();
16         } catch (IOException e) {
17             // TODO Auto-generated catch block
18             e.printStackTrace();
19         }
20         System.out.println("nama anda : " + nama);
21     }
22
23 }
```

```
Problems  Javadoc  Declaration  Console
<terminated> ContohInput [Java Application] /Library/Java/Java
nama :
herman
nama anda : herman
```

4. Buatlah sebuah kelas pojo dengan nama ClassInputScanner seperti code dibawah ini dan amati manfaat sintaksnya



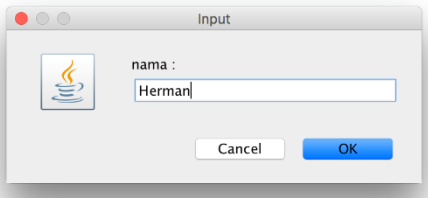
```
ContohInputScanner.java
1  package IO;
2
3  import java.util.Scanner;
4
5  public class ContohInputScanner {
6
7      public static void main(String[] args) {
8          Scanner input = new Scanner(System.in);
9          System.out.println("nama : ");
10         String nama = input.nextLine();
11         System.out.println("nama anda : " + nama);
12     }
13
14 }
```

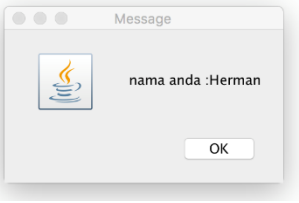
```
Problems  Javadoc  Declaration  Console
<terminated> ContohInputScanner [Java Application] /Library/J
nama :
Herman
nama anda : Herman
```

5. Buatlah sebuah kelas pojo dengan nama ClassInputJOptionPane seperti code dibawah ini dan amati manfaat sintaksnya

ContohInputJOptionPane.java

```
1  package IO;
2
3  import javax.swing.JOptionPane;
4
5  public class ContohInputJOptionPane {
6
7      public static void main(String[] args) {
8          String nama = JOptionPane.showInputDialog(null, "nama : ");
9          JOptionPane.showMessageDialog(null, "nama anda :" + nama);
10     }
11
12 }
13
```





6. Buatlah sebuah kelas pojo dengan nama ContohClassFile seperti code dibawah ini dan amati manfaat sintaksnya

ContohClassFile.java

```
1  package IO;
2  public class ContohClassFile {
3      public static void main(String[] args) {
4          // Pembuatan sebuah file
5          java.io.File file = new java.io.File("teksku.txt");
6          try {
7              if(file.createNewFile())
8                  System.out.println("File Berhasil Dibuat");
9              else
10                 System.out.println("File Gagal Dibuat");
11         } catch (Exception e) {
12             System.out.println("Error");
13         }
14
15         //Pembacaan sebuah file
16         System.out.println("Apakah fIle ada? " +file.exists());
17         System.out.println("Apakah bisa dibaca? " +file.canRead());
18         System.out.println("Apakah bisa ditulis? " +file.canWrite());
19         System.out.println("Apakah berupa direktori? " +file.isDirectory());
20         System.out.println("Apakah berupa file? " +file.isFile());
21         System.out.println("Apakah namanya? " +file.getName());
22         System.out.println("Dimanakah Pathnya? " +file.getPath());
23     }
24 }
```

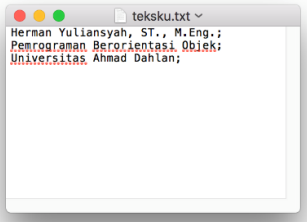
Problems Javadoc Declaration Console

```
<terminated> ContohClassFile [Java Application] /Library/Java/
File Berhasil Dibuat
Apakah fIle ada? true
Apakah bisa dibaca? true
Apakah bisa ditulis? true
Apakah berupa direktori? false
Apakah berupa file? true
Apakah namanya? teksku.txt
Dimanakah Pathnya? teksku.txt
```

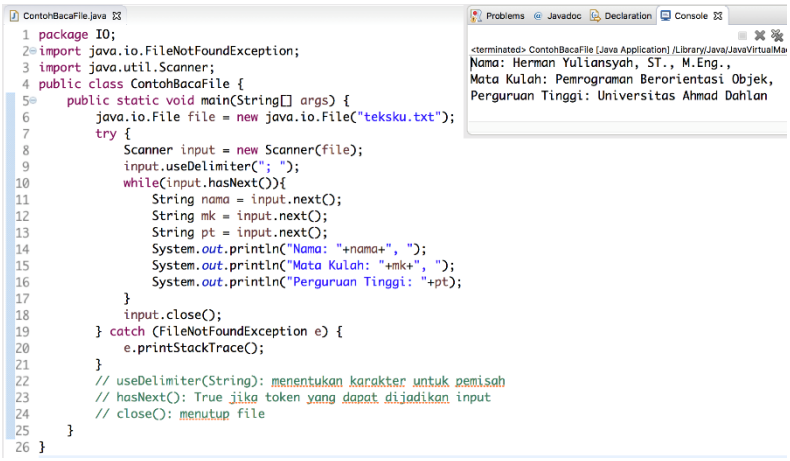
7. Buatlah sebuah kelas pojo dengan nama ContohTulisFile seperti code dibawah ini dan amati manfaat sintaksnya

ContohTulisFile.java

```
1  package IO;
2  import java.io.FileNotFoundException;
3  public class ContohTulisFile {
4      public static void main(String[] args) {
5          java.io.File file = new java.io.File("teksku.txt");
6          try {
7              java.io.PrintWriter output = new java.io.PrintWriter(file);
8              output.print("Herman Yuliansyah, ST., M.Eng.; ");
9              output.print("Pemrograman Berorientasi Objek; ");
10             output.print("Universitas Ahmad Dahlan; ");
11             output.close();
12         } catch (FileNotFoundException e) {
13             e.printStackTrace();
14         }
15     }
16 }
```



8. Buatlah sebuah kelas pojo dengan nama ContohBacaFile seperti code dibawah ini dan amati manfaat sintaksnya



```
ContohBacaFile.java
1  package IO;
2  import java.io.FileNotFoundException;
3  import java.util.Scanner;
4  public class ContohBacaFile {
5      public static void main(String[] args) {
6          java.io.File file = new java.io.File("teksku.txt");
7          try {
8              Scanner input = new Scanner(file);
9              input.useDelimiter("; ");
10             while(input.hasNext()){
11                 String nama = input.next();
12                 String mk = input.next();
13                 String pt = input.next();
14                 System.out.println("Nama: "+nama+", ");
15                 System.out.println("Mata Kulah: "+mk+", ");
16                 System.out.println("Perguruan Tinggi: "+pt);
17             }
18             input.close();
19         } catch (FileNotFoundException e) {
20             e.printStackTrace();
21         }
22         // useDelimiter(String): menentukan karakter untuk pemisah
23         // hasNext(): True jika token yang dapat dijadikan input
24         // close(): menutup file
25     }
26 }
```

```
Problems  Javadoc  Declaration  Console
<terminated> ContohBacaFile [Java Application] /Library/Java/JavaVirtualMa
Nama: Herman Yuliansyah, ST., M.Eng.,
Mata Kulah: Pemrograman Berorientasi Objek,
Perguruan Tinggi: Universitas Ahmad Dahlan
```

D. Post Test [Nmaks =100]

1. Buatlah sebuah program untuk menghitung luas persegi, dimana nilai sisi persegi diperoleh dari input pengguna dan hasil perhitungannya ditampilkan dilayar dan juga disimpan dalam sebuah file.
2. Berikan exception handling untuk masukan supaya melakukan pengecekan terhadap kesalahan pengisian nilai sisi persegi.

E. Daftar Pustaka

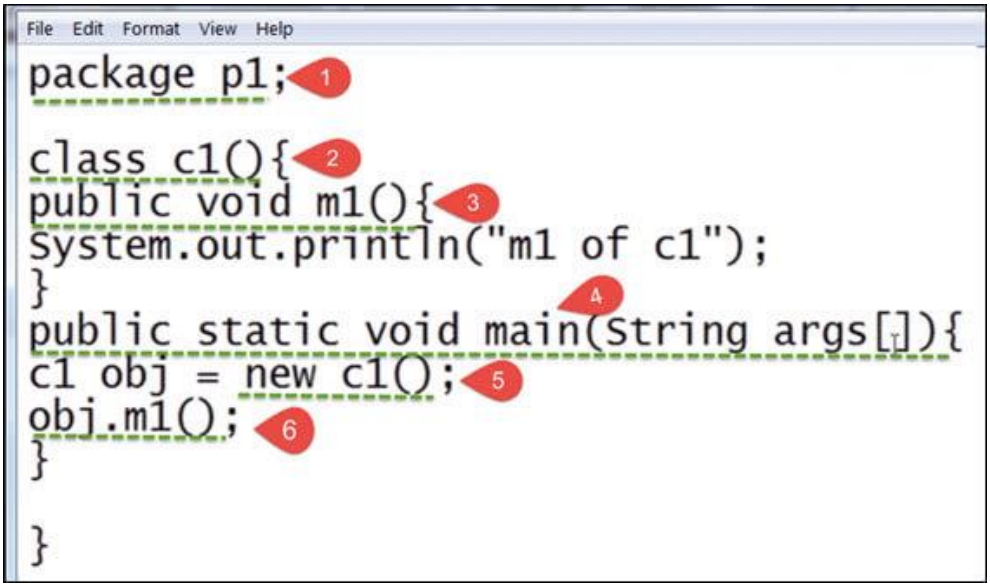
1. Khannedy, Eko K.”Modul Pelatihan Java Dasar”.Bandung
2. Anonim, 2009, Praktikum Pemrograman Berorientasi Obyek Politeknik Telkom Bandung

Package

Pertemuan ke : **XI**
Alokasi Waktu : **1,5 Jam**
Kompetensi Dasar : **Package di Java**
Indikator : **Mampu mengimplementasikan konsep Package di java**

A. Teori Pendukung

Sebuah package adalah sebuah koleksi dari kelas-kelas yang berhubungan. Package membantu untuk mengorganisasi kelas kedalam struktur direktori tertentu dan membuatnya menjadi mudah untuk dilokalisir ketika menggunakannya. Lebih penting lagi, package membantu untuk meningkatkan kerja re-usability (penggunaan ulang).
Setiap package dijava memiliki nama unik dan mengorganisasi kelas dan interface nya kedalam namespace atau nama grup yang terpisah.



Keterangan

1. Untuk meletakkan sebuah kelas dalam sebuah package, pada baris pertama dari code mendefinisikan package p1
2. Membuat sebuah class dengan nama c1
3. Mendefinisikan sebuah method dengan nama m1 yang berisikan sintaks print sebuah baris.
4. Mendefinisikan main method
5. Membuat sebuah object dari class c1
6. Memanggil method m1

Terdapat dua macam package yaitu:

1. package-package default Java
2. package-package yang dibuat sendiri,

Untuk menggunakan suatu package kedalam kelas yang berbeda package dapat digunakan sintaks import. Proses import package dapat hanya memanggil hanya nama package atau bisa memanggil khusus pada satu class yang dituju.
Kesepakatan Penamaan Package
Sebuah nama package dibuat dengan reverse dari Internet Domain Name (untuk meyakinkan keunikan) ditambah dengan tambahan sesuatu yang unik dengan memisahkan dengan dot (.). Nama package menggunakan lowercase sebagai contoh: Internet Domain Name-nya adalah

"zzz.com", sehingga penamaan package dapat dengan "com.zzz.project1".

Beberapa nama package yang sudah dimiliki oleh java secara default yaitu:

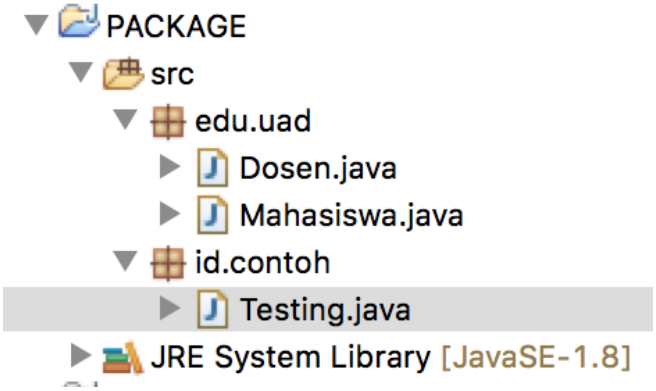
java.lang	— basic language functionality and fundamental types
java.util	— collection data structure classes
java.io	— file operations
java.math	— multiprecision arithmetics
java.nio	— the Non-blocking I/O framework for Java
java.net	— networking operations, sockets, DNS lookups, ...
java.security	— key generation, encryption and decryption
java.sql	— Java Database Connectivity (JDBC) to access databases
java.awt	— basic hierarchy of packages for native GUI components
javax.swing	— hierarchy of packages for platform-independent rich GUI components
java.applet	— classes for creating an applet
java.beans	— Contains classes related to developing beans -- components based on the JavaBean architecture.
java.text	— Provides classes and interfaces for handling text, dates, numbers, and messages in a manner independent of natural languages.
java.rmi	— Provides the RMI package.
java.time	— The main API for dates, times, instants, and durations.

B. Pre Test [Nmaks =100]

1. Sebutkan jenis-jenis package di Java?
2. Tulis sintaks cara pemanggilan Package di Java?
3. Sebutkan penggunaan access modifier yang dapat digunakan antar package?

C. Langkah Praktikum

1. Buat project prak11 dengan menggunakan netbeans
2. Buatlah dua buah class dan tiga buah class sehingga project anda akan tampil seperti ini:



3. Buatlah sebuah kelas pojo dengan nama Mahasiswa seperti code dibawah ini dan amati manfaat sintaksnya

Mahasiswa.java

```
1 package edu.uad;
2
3 public class Mahasiswa {
4
5     public void infoMahasiswa(){
6         System.out.println("Ini method dari class mahasiswa");
7     }
8 }
9
```

4. Buatlah sebuah kelas pojo dengan nama Dosen seperti code dibawah ini dan amati manfaat sintaksnya

Dosen.java

```java
package edu.uad;

public class Dosen {

    public void infoDosen(){
        System.out.println("Ini method dari class dosen");
    }
}

```

5. Buatlah sebuah kelas pojo dengan nama Testing seperti code dibawah ini dan amati manfaat sintaksnya

Testing.java

```java
package id.contoh;

import edu.uad.*;
public class Testing {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Mahasiswa mhs = new Mahasiswa();
        mhs.infoMahasiswa();

        Dosen dsn = new Dosen();
        dsn.infoDosen();
    }

}
```

D. Post Test [Nmaks =100]

Buatlah sebuah program untuk memanfaatkan package default dari java.

E. Daftar Pustaka

1. Khannedy, Eko K."Modul Pelatihan Java Dasar".Bandung
2. Anonim, 2009, Praktikum Pemrograman Berorientasi Obyek Politeknik Telkom Bandung

Aplikasi Games

Pertemuan ke : XII
Alokasi Waktu : 1,5 Jam
Kompetensi Dasar : Games 2D dengan Java
Indikator : Mampu mengimplementasikan konsep java pada Games 2D

A. **Teori Pendukung**

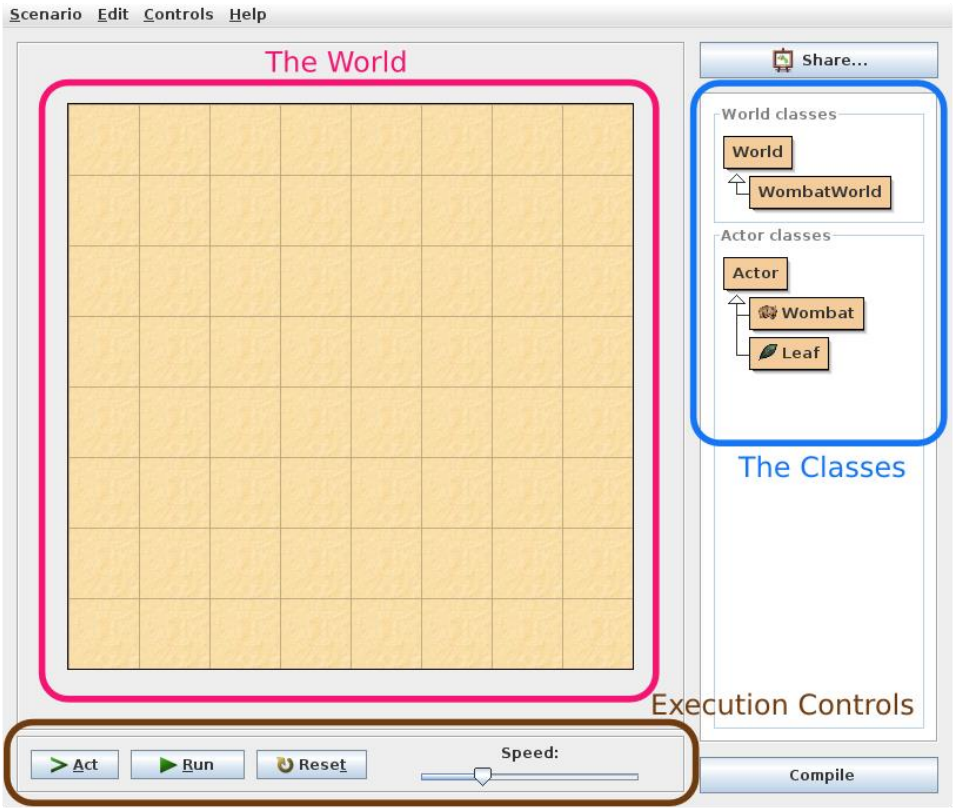
Greenfoot merupakan lingkungan pengembangan interaktif Java yang dirancang terutama untuk tujuan pendidikan di sekolah tinggi dan tingkat sarjana. Hal ini memungkinkan pengembangan yang mudah dari aplikasi grafis dua dimensi, seperti simulasi dan permainan interaktif.

Greenfoot sedang dikembangkan dan dipelihara di University of Kent, dengan dukungan dari Oracle. Ini adalah perangkat lunak bebas, dirilis di bawah lisensi GPL. Greenfoot tersedia untuk Windows, OS X, Linux, Solaris, dan setiap JVM baru-baru ini. Proyek Greenfoot ini diprakarsai oleh Michael Kölling pada tahun 2003, dan prototipe pertama dibangun oleh Poul Henriksen (mahasiswa master) dan Michael Kölling (supervisor) pada tahun 2003/2004. [1] Dari tahun 2005 pembangunan dilanjutkan melibatkan anggota lain dari Grup BlueJ di University of Kent dan Deakin University. [2]

Rilis penuh pertama, Greenfoot versi 1.0, diterbitkan pada tanggal 31 Mei 2006, dengan rilis lanjut berikut sesekali setelahnya[3]. Pada bulan Mei 2007, proyek Greenfoot dianugerahi "Duke Choice Award" untuk kategori "Teknologi Java dalam Pendidikan", dan pada tahun 2010 memenangkan "Premier Award untuk Keunggulan dalam Pendidikan Teknik Courseware". Pada bulan Maret 2009, proyek Greenfoot menjadi bebas dan perangkat lunak open source, dan berlisensi di bawah GNU GPL dengan pengecualian Classpath. Pada bulan Agustus 2009, buku teks [4] diterbitkan yang mengajarkan pemrograman dengan Greenfoot.

Model pemrograman Greenfoot terdiri dari kelas World (diwakili oleh area layar persegi panjang) dan sejumlah objek Actor yang hadir di World dan dapat diprogram untuk bertindak independen. World dan Actor yang diwakili oleh objek Java dan didefinisikan oleh kelas Java. Greenfoot menawarkan metode untuk dengan mudah memprogram para Actor ini, termasuk metode untuk gerakan, rotasi, perubahan penampilan, tabrakan, dll

Pemrograman di Greenfoot di dasar yang paling terdiri dari subclassing dua built-in kelas, World dan Actor seperti pada Gambar 1. Sebuah contoh dari subclass dunia mewakili World di mana eksekusi Greenfoot akan terjadi. subclass Actor adalah objek yang bisa eksis dan bertindak di World. Sebuah contoh dari subclass dunia secara otomatis dibuat oleh lingkungan.



Gambar 1. User interface Greenfoot

Eksekusi di Greenfoot terdiri dari loop utama built-in yang berulang kali memanggil metode tindakan masing-masing Actor. Pemrograman skenario, oleh karena itu, sebagian besar terdiri dari menerapkan metode tindakan bagi Actor skenario ini. Implementasi dilakukan di Java standar. Greenfoot menawarkan metode API untuk berbagai tugas umum, seperti animasi, suara, pengacakan, dan manipulasi gambar. Semua Java perpustakaan standar dapat digunakan juga, dan fungsi canggih dapat dicapai.

Greenfoot bertujuan untuk memotivasi peserta didik cepat dengan menyediakan akses mudah ke grafis animasi, suara dan interaksi. Lingkungan sangat interaktif dan mendorong eksplorasi dan eksperimen. Pedagogis, desain didasarkan pada pendekatan konstruktivis dan magang.

Kedua, lingkungan dirancang untuk menggambarkan dan menekankan abstraksi penting dan konsep pemrograman berorientasi objek. Konsep seperti hubungan kelas/objek, metode, parameter, dan interaksi objek yang disampaikan melalui visualisasi dan interaksi dipandu. Tujuannya adalah untuk membangun dan mendukung model mental yang benar merupakan sistem pemrograman berorientasi objek modern.

Bagi pengguna Greenfoot, sebuah situs komunitas disebut Gallery Greenfoot [5] menyediakan platform untuk mempublikasikan dan mendiskusikan proyek-proyek mereka. Siapapun dapat membuat account di Gallery Greenfoot dan mempublikasikan karya mereka. Bila dipublikasikan, skenario menjalankan hidup di web browser, dan langsung dimainkan oleh siapa saja di seluruh dunia. Kemampuan untuk dengan mudah mempublikasikan proyek pemrograman ke internet dipandang sebagai motivator signifikan bagi pelajar muda.

Untuk pendidik, proyek menyediakan Greenroom, [6] sebuah situs komunitas untuk membahas strategi pengajaran, bertukar pengalaman dan materi saham. Selain forum diskusi, Greenroom menyediakan repositori bersama dari sumber-sumber pengajaran, termasuk banyak lembar kerja, ide proyek, geser set dan alat peraga lainnya.

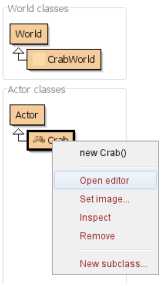
Dalam versi 2.0, editor dan API menerima beberapa perubahan besar dari versi sebelumnya 1.5.6. editor menambahkan dukungan untuk penyelesaian kode, berwarna lingkup menyoroti, ditingkatkan mencari dan mengganti, dan pandangan navigasi baru. [7] Sebuah built-in perekam suara ditambahkan, serta dukungan yang lebih baik untuk suara dalam skenario melalui tingkat tinggi kelas suara baru yang disebut GreenfootSound. Ia menambahkan kemampuan untuk berhenti sejenak dan suara lingkaran, serta menambahkan dukungan MP3.

B. Pre Test (Nmaks =100)

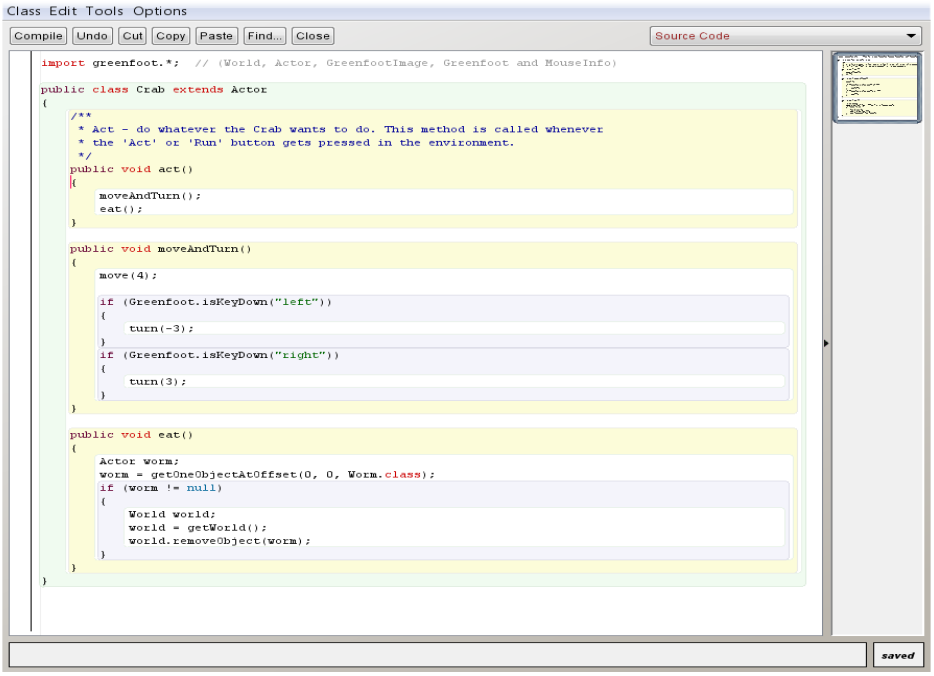
1. Jelaskan apakah yang dimaksud dengan Actor di greenfoot?
2. Jelaskan apakah yang dimaksud dengan World di greenfoot?

C. Langkah Praktikum

1. Download contoh scenario greenfoot di url: http://www.greenfoot.org/tutorial-files/modern-crab.zip
2. Buka Aktor Crab dengan klik kanan pilih open editor.

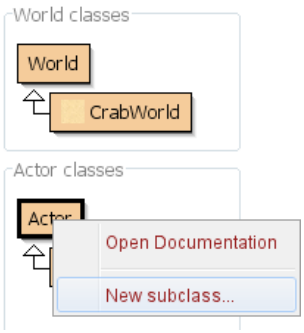


3. Lengkapi class crab menjadi seperti berikut:

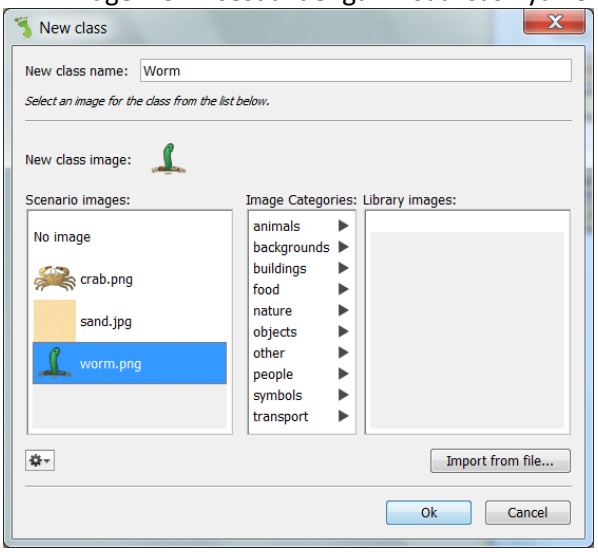



```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

public class Crab extends Actor
{
    /**
     * Act - do whatever the Crab wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        moveAndTurn();
        eat();
    }

    public void moveAndTurn()
    {
        move(4);

        if (Greenfoot.isKeyDown("left"))
        {
            turn(-3);
        }
        if (Greenfoot.isKeyDown("right"))
        {
            turn(3);
        }
    }

    public void eat()
    {
        Actor worm;
        worm = getOneObjectAtOffset(0, 0, Worm.class);
        if (worm != null)
        {
            World world;
            world = getWorld();
            world.removeObject(worm);
        }
    }
}
```

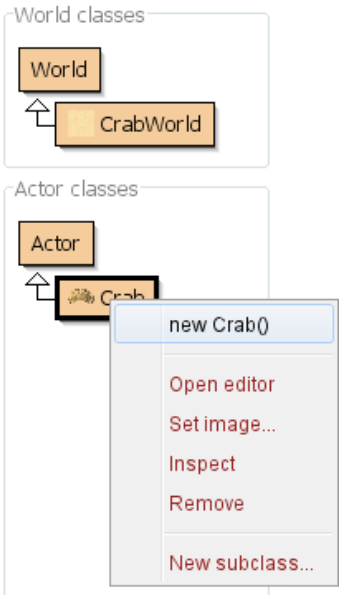
4. Buat sebuah class dengan nama Worm dengan cara klik kanan pada kelas actor dan pilih New subclass.



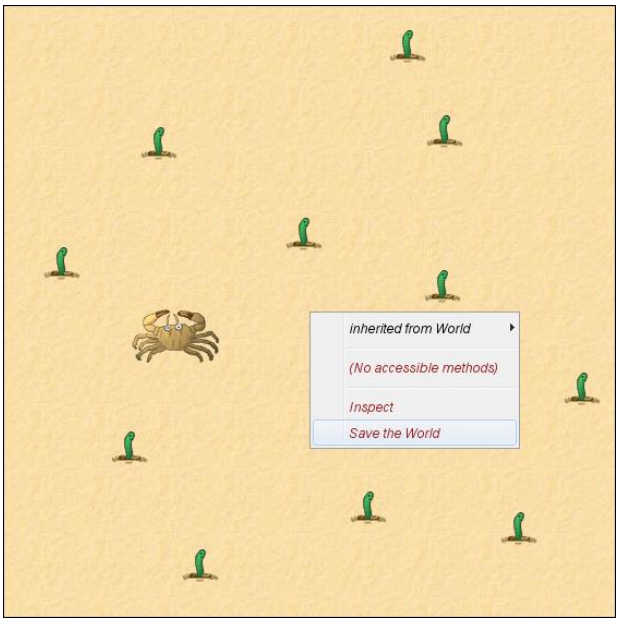
5. Pilih image worm sesuai dengan visualisasinya kemudian pilih tombol Ok.



6. Buatlah beberapa object Worm dan satu object Crab kemudian letakkan di World dengan cara klik kanan pada class Worm dan Class Brab, pilih new Crab() untuk membuat objek Crab dan pilih new Worm() untuk membuat objek Worm serta letakkan keduanya di World.



7. Letakkan beberapa Worm dan satu Crab sehingga akan tertampil seperti berikut:



8. Klik kanan pada World kemudian pilih Save the World untuk menyimpan sebaran objek yang telah dibuat.
9. Pilih Tombol Run untuk menjalankan greenfoot dan gunakan panah pada keyboard untuk mengatur gerak dari Crab.

D. Post Test (Nmaks =100)

1. Jelaskan kegunaan dari perintah berikut:
    a. Public void act()
    b. move();
    c. turn();
    d. Greenfoot.isKeyDown("left");
    e. getOneObjectAtOffset();
    f. removeObject();
    g. Greenfoot.playSound("eating.wav");
2. Buatlah ringkasan dari isi dokumentasi greenfoot berikut (**Tugas Rumah**):
    a. http://www.greenfoot.org/files/javadoc/
    b. http://www.greenfoot.org/files/javadoc/greenfoot/Greenfoot.html
    c. http://www.greenfoot.org/files/javadoc/greenfoot/World.html
    d. http://www.greenfoot.org/files/javadoc/greenfoot/Actor.html

E. Daftar Pustaka

1. P. Henriksen Masters thesis: A Direct Interaction Tool for Object-Oriented Programming Education. http://www.greenfoot.org/papers/polleMasterThesis.zip
2. Greenfoot contributors. http://www.greenfoot.org/about/contributors.html
3. Version history. http://www.greenfoot.org/doc/version-history.html
4. Textbook: Introduction to Programming with Greenfoot. http://www.greenfoot.org/book/
5. Greenfoot Gallery. http://www.greenfoot.org/home
6. Greenroom. http://greenroom.greenfoot.org/
7. "Greenfoot 2.0 – The new features – mik's blog". Blogs.kent.ac.uk. 13 October 2010. Retrieved 2013-07-05. http://blogs.kent.ac.uk/mik/2010/10/05/greenfoot-2-0-the-new-features/