MODUL PRAKTIKUM

Laboratorium Teknik Informatika

Java™

STRUKTUR BAHASA PEMROGRAMAN

(Java)

Ft.Umj

MODUL PRAKTIKUM STRUKTUR BAHASA PEMROGRAMAN

Tim Penyusun : Arsya Rifki
M. Nur Arifin
Dani Darmawan
Asmarita Dewi

Modul ini dicetak untuk panduan praktikum.
Dicetak Oktober, 2020 oleh Laboratorium Teknik Informatika FT UMJ

Laboratorium Teknik Informatika
Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muhammadiyah Jakarta
Jalan Cempaka Putih Tengah 27, Jakarta 10510
Telepon: (021)70329963
Website: http://if.umj.ac.id/laboratorium/
Email: lab.informatika@ftumj.ac.id

LEMBAR PENGESAHAN
MODUL PRAKTIKUM STRUKTUR BAHASA PEMROGRAMAN

Telah disetujui dan disahkan sebagai
Modul kuliah praktikum Struktur Bahasa Pemrograman

Disahkan di : Jakarta
Pada Tanggal : Oktober 2020

Menyetujui,
Kepala Laboratorium Program Studi Teknik Informatika
Fakultas Teknik Universitas Muhammadiyah Jakarta

Sitti Nurbaya Ambo, MMSI
NIDN: 0307067901

Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Teknik Universitas Muhammadiyah Jakarta

Popy Meilina, M.Kom
NIDN: 0305057901

KATA PENGANTAR

Assalamua'alaikum wr. wb.
Bismillahirrohmanirrohiim

Dengan segala doa dan harapan, modul praktikum Struktur Bahasa Pemrograman ini diharapkan dapat menjadi panduan yang baik dan jelas bagi siapapun yang membacanya. Modul ini akan digunakan dalam proses belajar praktikum Struktur Bahasa Pemrograman.

Tahun ini, Laboratorium Teknik Informatika FT-UMJ bekerja sama dengan dosen pengampu mata kuliah Struktur Bahasa Pemrograman berusaha untuk menyelaraskan pelaksanaan praktikum dengan pelaksanaan kuliah di kelas. Dengan usaha tersebut kami berharap penyerapan materi Struktur Bahasa Pemrograman oleh para mahasiswa dapat lebih efektif, sehingga pada akhir semester hasil evaluasi belajar mahasiswa dapat lebih meningkat dibandingkan dengan tahun-tahun yang lalu.

Sebagai penutup, saya ucapkan banyak-banyak terima kasih kepada semua pihak yang telah membantu terselesaikannya modul ini, dosen-dosen pengampu mata kuliah Struktur Bahasa Pemrograman, serta tim penyusun modul Laboratorium Teknik Informatika FT-UMJ atas seluruh dedikasi yang diberikan kepada Lab tercinta.

Terima kasih sebelumnya kami sampaikan kepada para pembaca modul ini, semoga ilmu yang telah dipelajari dapat bermanfaat dan semakin berkembang di tangan anda semua.

Wassalamu'alaikum wr. wb.
Jakarta, Oktober 2020
Laboratorium Teknik Informatika FT-UMJ,

Tim penyusun

DAFTAR ISI

MODUL 1 : PENGENALAN CLASS

Sejauh ini memang sebenarnya kita telah menggunakan *class* dalam setiap pembuatan contoh program yang kita tulis. Akan tetapi, *class* yang kita gunakan adalah *class* yang masih minim sekali, yang hanya ditunjukan untuk mendemonstrasikan sintaks-sintaks dalam pemograman java.

1.1 Pengertian *Class*

Class dapat didefenisikan sebagai cetak biru (*blueprint*) atau *prototype* / kerangka yang mendefenisikan *variable-variable* (data) dan *method-method* (prilaku) umum dari sebuah objek tertentu.

Dalam dunia pemograman, sebenarnya *class* tidak jauh beda dengan tipe data sederhana. Perbedaannya tipe data sederhana digunakan untuk mendeklarasikan variable normal, sedangkan *class* digunakan untuk mendeklarasikan variable yang berupa objek. Variable yang berupa objek sering juga disebut dengan referensi objek (oject reference).

Syntax :

```
Class nama_class {
        Statement1;
        Statement2;
        …
        Statement n;
}
```

Contoh Program 1.1 :

```
Class kotak{
        double panjang;
        double lebar;
        dauble tinggi;
}
Class demokotak{
Public static void main (string () args){
        double volume;
```

```
        kotak k = new kotak();
        k.panjang =3;
        k.lebar=3;
        k.tinggi =3;
        volume = k.panjang + k.tinggi + k.lebar;
        }

    }
```

Contoh program 1.2 :

@cobain1.java

```
public class cobain1{
    public static void main(String[]args){
        System.out.println("Hello kawan !:");
        cobain cobainbaru=new cobain();
        cobainbaru.coba();
    }
}
```

@cobain.java

```
public class cobain{
       void coba (){
              System.out.println (“ini program pertama”);
     }
}
```

1.2 Jenis Class

Java menizinkan kita untuk mendefenisikan suatu *class* yang bersifat generik. Selanjutnya, *class* tersebut dapat diturunkan lagi menjadi *class* baru dengan sifat yang lebih spesifik. Dalam terminology java, *class* induk yang diturunkan disebut dengan superclass. Adapun kelas baru hasil turunannya disebut dengan subclass. Pada proses penurunan *class* ini, *class* turunan akan mewarisi sifat-sifat yang terdapat pada *class*

induknya. Selanjutnya *class* turunan tersebut dapat memiliki sifat-sifat spesifik yang sebelumnya tidak dimiliki oleh class induknya.

Contoh program 1.3:

```
@DemoPewarisan.java
class A{
    private int a;
    public void getA(int nilai){
        a=nilai;
    }
    public int getA(){
return a;
    }
}
class B extend A{
    private int a;
    public void getA(int nilai){
        a=nilai;
    }
    public int getA(){
return a;
    }
}

class DemoPewarisan{
    public static void main(String args[]){
        B obj=new B();
        obj.setA(20);
        obj.setB(30);
        System.out.println("Nilai A = "+obj.getA());
        System.out.println("Nilai A = "+obj.getB()};

    }
}
```

Tugas Akhir Modul 1

1. Buatlah program superclass dan subclass utnuk menghitung luas dan keliling dari pilihan dibawah ini :
    - Trapezium sama kaki
    - Trapesium siku-siku
    - Laying-layang
    - Jajar genjang

MODUL 2 : Statement Method

Method adalah sarana bagi programmer untuk memodularisasi, artinya membreak atau memecah program kompleks menjadi bagian yang kecil-kecil sehingga nantinya akan dapat digunakan berulang-ulang, daripada harus menulis beberapa baris kode yang sama. Method dalam java mirip dengan fungsi atau procedure dalam bahasa pemograman yang lain. Pada konsep Object Oriented Programing, function lebih dikenal dengan istilah method dimana merupakan suatu bagian dari Object yang mendefenisikan apa yang biasa Object tersebut lakukan.

2.1 Pengertian Method

Jadi *method* adalah fungsi atau prosedur yang dibuat oleh sorang programmer didalam suatu *class*. *Method* dapat dibagi menjadi fungsi dan prosedur. Fungsi adalah bagian atau sub program yang mempunyai algoritma tertentu dalam menyelesaikan suatu masalah dengan mengembalikan hasil. Prosedur adalah bagian tau sub dari program yang mempunya algoritma tertentu dalam menyelesaikan suatu masalah tanpa mengembalikan suatu nilai hasil. Secara umum method menghasilkan suatu nilai dengan tipe data tertentu dengan Return Type. Apabila terdapat suatu fungsi yang tidak menghasilkan suatu nilai apapun maka bagian return type ini diganti dengan void.

Berikut adalah karakteristik dari *method*:

a. Dapat mengembalikan satu nilai atau tidak sama sekali
b. Dapat diterima beberapa parameter yang dibutuhkan atau tidak ada parameter sma sekali. Parameter bias juga juga disebut sebagai argumrnt dari fungsi
c. Setelah *method* telah selesai dieksekusi, dia akan kembali pada method yang memanggilnya.

.

2.2 *Method* Tanpa Parameter

Method tanpa parameter adalah *method* yang tidak memiliki parameter,namun *method* ini juga bisa mengakses perhitungan yang bisa mengembalikan nilai (return).

Syntax:

```
Tipe_Data Nama_Method(){
        Statement 1;
        Statement 2;
        …
        Statement n;
```

Contoh program 2.1:

```
@CetakBintang.java
public class CetakBintang{
    public static void bintang(){
        for(int i=8;i>=1;i--){
        for(int j=i;j>=1;j--){
            System.out.println("*");
        }
            System.out.println();
    }
    }

    public static void main(String[]args){
        System.out.println("Segitiga bintang bernilai 8");
        System.out.println();
        bintang();
    }
}
```

2.3 *Method* Berparameter

Meskipun java mengizinkan *method* tanpa parameter, namun pada kenyataanya sebagian besar *method* yang ditulis dalam program memiliki satu atau beberapa parameter. Dengan adanya parameter, sebuah method dapat bersifat dinamis dan general. Artinya method tersebut dapat mengembalikan nilai yang seragam sesuai dengan nilai parameter yang dilewatkanya.

Syntax :

```
Type_Data Nama_Method(daftar_parameter){
Statement 1;
Statement 2;
…
Statement n;
}
```

Contoh program 2.3:

```
@LuasKelilingLinkaran.java
class Lingkaran{
    double jarijari;
    double Luas(double j){
        jarijari=j;
        return(3.14*jarijari*jarijari);
    }
    double Keliling(double j){
        jarijari=j;
        return(2*3.14*jarijari)
    }
}

class LuasKelilingLingkaran{
    public static void main(String[]args){
        Lingkaran L;
```

```
        L =new Lingkaran();
        System.out.println("Luas Lingkaran : "+L.Luas(10));
        System.out.println("Keliling Lingkaran : "+L.Keliling(10));
    }
}
```

Tugas akhir modul 2

1. Buatlah program Method denganpengimplementasian Superclass dan Subclass untuk menghitung matriks 5 x5 dari pilihan dibawah ini:
   - Jumlah tiap baris
   - Jumlah tiap kolom
   - Nilai diagonal 1 dan nilai diagonal 2

MODUL 3 : PENGINPUTAN

3.1 *BufferedReader*

BufferedReader adalah class abstrak yang menanganu baca tulis kesuatu media. *Class* ini membutuhkan class lain sebagai pekerjaannya, yaitu InputStreanReader. Dan InputStreanReader membutuhkan media tempat baca tulis dilakukan, yaitu System.in. Fungsi BufferedReader ini adalah digunakan untuk menangkap inputan dari keyboard.

Contoh Program 3.1 :

```
@modul3_1.java
import java.io.*;

public class modul3 _1{
public static void main(String[]args) throws IOException{
BufferedReader nama = new BufferedReader( new InputStreamReader(Sytem.in));
    System.out.print("NAMA :");
    String a = nama.readLine();
    System.out.print("Hello"+a);
}
}
```

Contoh program 3.2:

```
@ifelse.java
Import java.io.*;
public class Ifelse {
    public static void main (String[]args) throws IOException{
        BufferReader dataAngka=new BufferReader(new Input Stream
Reader(system.in));
        System.out.println("Menghitung Grade Nilai");
        System.out.println("======================");
        System.out.print("Masukan Nilai Anda : ");
```

```
            String strl = dataAngka.readLine();
            int bill = Integer.parseInt(strl);
            if (bill >= 85)
                System.out.println("Grade : A");
            else if ((bill < 70) && (bill >=55))
                System.out.println("Garde : B");
            else if ((bill < 55) && (bill >=40))
                System.out.println("Garde : c");
            else if ((bill < 40) && (bill >=25))
                System.out.println("Garde : D");
            else if ((bill < 25) && (bill >=0))
                System.out.println("Garde : E");
        }
    }
```

3.2 *Scanner*

Selain dengan BufferedReader ada cara lain melakukan inputan dipemograman bahasa java, yaitu menggunakan Scanner yang terdapat dari java.util package.

Contoh program 3.2:

```
@ProgramScanner.java
import java.until.Scanner;

public class ProgramScanner{
    public static void main(String[]args){
    Scanner masukan = new Scanner(System.in);
        System.out.println("Masukan suatu integer : ");
        int angka = masukan nextInt();

        if(angka % 5 = 0)
            System.out.println(angka+"Adalah Kelipatan Lima");
        if(angka % 2 = 0)
```

```
            System.out.println(angka+"Adalah angka genap");
        }
    }
```

3.3 *JOptionPane*

Ada sebuah cara lagi melakukan input dipemograman bahasa java, yaitu menggunakan JOptionPane yang terdapat dari jvax.swing package. JOptionPane memudahkan munculnya dialog box standard.

Contoh program 3.3:

```
@InputNama.java
import javax.swing.JOptionPane;
public class InputNama{
    public static void main(String[]args){
        String nama = "";
        nama = JOptionPane.showInputDialog("Masukan Nama Anda");
        String msg="Hello" +nama+ "!!";
        JOpionPane.showMassageDialog(null,msg);
    }
}
```

Tugas Akhir Modul 3:

1. Buatlah rangkuman modul 4
2. Buatlah program pengimplementasian penginputan menggunakan BufferedReader dan mehod mengenai pilihan dibawah ini:
   - Pertambahan
   - Pengurangan
   - Perkalian
   - Pembagian
   - Modularisasi

MODUL 4 : INHERITANCE, ENCAPSULATION, DAN POLYMORPHISM

4.1 Inheritance

Inheritance atau pewarisan adalah pendeklarasian yang dilakukan dengan cara menurunkan atribut beserta method (dengan catatan selain private) dari superclass ke subclass. Pengertian dari superclass adalah class yang mewariskan atribut dan method-nya ke subclass sedangkan subclass merupakan class yang diwarisi sifat-sifat dari superclass. Dan nantinya class yang baru yang telah diwarisi sifat dari superclass, akan mempunyai sifat-sifat yang sama dengan superclass.

Ada juga beberapa jenis dari inheritance yaitu single inheritance, multiple inheritance, dan multilevel inheritance. Single inheritance merupakan suatu class yang hanya mempunyai satu superclass, sedangkan multiple inheritance adalah class yang mempunyai lebih dari satu superclass-nya pada variable dan method yang telah diwariskan merupakan hasil variasi dari superclass-nya. Dan multilevel inheritance adalah subclass yang menjadi superclass bagi class yang lain.

Dalam pewarisan terdapat method overriding dan overloading. Pengertian dari method overriding merupakan pendefinisian kembali method yang sama. Contohnya kasusnya misalnya penulisan nama method yang sama atau pada parameter (signature) yang ada pada subclass. Lalu overloading merupakan pendefinisian pada method yang sama, tetapi parameter yang beda dalam definisi class yang sama (masih dalam satu class).

Contoh Program 4.1:

```
package pewarisan;

class pegawai
{
    public int nip,npp;
    public void  data(int nip,int npp)
    {
```

```
        this.nip=nip;
        this.npp=npp;
    }

    int getNip()
    {
        return nip;
    }

    int getNpp()
    {
        return npp;
    }

    public void tampil()
    {
        System.out.println("  Data NIP : "+nip);
        System.out.println("  Data NPP : "+npp);
    }
}

class manager extends pegawai
{
    protected String nama_manager,alamat;
    protected int umur;

    public void   datam(String nama_manager,String alamat,int umur,int n1,int
n2)
    {
        this.alamat=alamat;
        this.nama_manager=nama_manager;
        this.umur=umur;
        super.data(n1, n2);
    }

    String getNama_manager()
    {
        return nama_manager;
```

```
    }

    String getAlamat()
    {
        return alamat;
    }
    int getUmur()
    {
        return umur;
    }

    public void tampil()
    {
        System.out.println(" 1. No Induk Pegawai : "+nip);
        System.out.println(" 2. No Pokok Pegawai : "+npp);
    }

}

class supervisor extends manager
{
        String nama_supervisor;

        public void datas(String nama_supervisor , String alamat, int umur,int
n1,int n2  )
        {
            this.nama_supervisor=nama_supervisor;
            super.datam(nama_manager, alamat, umur,n1,n2);
            super.data(n1, n2);
        }

        String getNama_supervisor()
                {
                        return nama_supervisor;
        }
        public void tampil()
        {
            System.out.println(" Data Supervisor ");
```

```
            System.out.println("------------------");
            super.tampil();
            System.out.println(" 3. Nama             : "+nama_supervisor);
            System.out.println(" 4. Alamat           : "+alamat);
            System.out.println(" 5. Umur             : "+umur);
        }
}

public class Pewarisan {
    public static void main(String[] args) {
        // TODO code application logic here
            pegawai baru=new pegawai();
            manager m1=new manager();
            supervisor s1=new supervisor();
            //inisiasi nilai
            baru.nip=2013131;
            baru.npp=1313;
            //input data
            m1.datam(" Van Travis  "," Pamulang ", 20,baru.nip,baru.npp);
            s1.datas(" Sinta Indah  "," Bogor ", 18,baru.nip,baru.npp);
        System.out.println(" Data Manager ");
        System.out.println("------------------");
        m1.tampil();
        System.out.println(" 3. Nama             : "+m1.nama_manager);
        System.out.println(" 4. Alamat           : "+m1.alamat);
        System.out.println(" 5. Umur             : "+m1.umur);
        s1.tampil();
    }

}
```

4.2 *Encapsulation*

Enkapsulasi adalah pembungkus, pembungkus disini dimaksudkan untuk menjaga suatu proses program agar tidak dapat diakses secara sembarangan atau di intervensi oleh

program lain. Konsep enkapsulasi sangat penting dilakukan untuk menjaga kebutuhan program agar dapat diakses sewaktu-waktu, sekaligus menjaga program tersebut.

Dalam kehidupan sehari hari enkapsulasi dapat dimisalkan sebagai arus listrik pada generator, dan system perputaran generator untuk menghasilkan arus listrik. Kerja arus listrik tidak mempengaruhi kerja dari system perputaran generator, begitu pula sebaliknya. Karena didalam arus listrik tersebut, kita tidak perlu mengetahui bagaimana kinerja system perputaran generator, apakah generator berputar kebelakang atau kedepan atau bahkan serong. Begitu pula dalam system perputaran generator, kita tidak perlu tahu bagaimana arus listrik, apakah menyala atau tidak.

Begitulah konsep kerja dari enkapsulasi, dia akan melindungi sebuah program dari akses atau pun intervensi dari program lain yang mempengaruhinya. Hal ini sangat menjaga keutuhan program yang telah dibuat dengan konsep dan rencana yang sudah ditentukan dari awal.

Contoh Program 4.2:

```
package cv;
public class Cv {
    private int nim;
    private double alas,tinggi;
    public void setNim(int n)
    {
        this.nim=n;
    }
    public int getNim()
    {
        return nim;
    }
    public void input (double a,double b)
    {
        this.alas=a;
```

```
        this.tinggi=b;
    }
    public double hitung()
    {
        return alas*tinggi/2;
    }

public static void main(String[] args) {
    Cv c=new Cv();
    c.setNim(4234242);
    c.input(4.3, 90);
    System.out.println(" Nim Anda : "+c.getNim());
    System.out.println(" \n");
    System.out.println(" Luas Segitiga : "+c.hitung());
}
}
```

4.3 *Polymorphism*

Polymorphism merupakan sebuah pemikiran yang intinya suatu hal yang sama dapat menghasilkan bentuk dan perilaku yang berbeda-beda. *Polymorphism* juga berarti terdapat operasi yang sama pada *class* yang berbeda.

Dalam pembuatan *Polymorphism* terdapat beberapa aspek yang harus diperhatikan :

a. *Method access attribute* yang ada pada *subclass* harus lebih spesifik dibandingan dengan yang ada di *superclass.*
b. *Method-method* yang akan dipanggil, cara pemanggilannya harus melalui variable-variabel yang ada pada *superclass*.
c. *Method-method* yang akan dipanggil harus menjadi *method* juga di *superclass.*
d. Parameter yang ada di method harus sama antara yang ada di *subclass* dan yang ada di *superclass*

Contoh Program 4.3 :

```
package polymorphism;
class kendaraan
{
    private int posisi;

    public kendaraan(int posisi)
    {
        this.posisi=posisi;
    }
    public void jalan()
    {
        return;
    }
    public void setPosisi(int posisi)
    {
        this.posisi=posisi;
    }
    public int getPosisi()
    {
        return posisi;
    }
}

class mobil extends kendaraan
{
    public mobil(int posisi)
    {
        super(posisi);
    }

    @Override
```

```
    public void jalan()
    {
        setPosisi(getPosisi() + 45);
    }

}

class motor extends kendaraan
{
    public motor(int posisi)
    {
        super(posisi);
    }
    @Override
    public void jalan()
    {
        setPosisi(90);
    }
}

class pesawat extends kendaraan
{
    public pesawat(int posisi)
    {
        super(posisi);
    }
    @Override
    public void jalan()
    {
        setPosisi(getPosisi() + 300);
    }
}

public class Polymorphism {
```

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


        kendaraan buatKendaraan[]=new kendaraan[3];
       int posisi_awal=10;
      buatKendaraan[0]=new mobil(posisi_awal);
      buatKendaraan[1]=new motor(posisi_awal);
      buatKendaraan[2]=new pesawat(posisi_awal);
        System.out.println("  Posisi Kendaraan Sebelum Di Jalankan ");
        System.out.println(" ---------------------------------------");
        System.out.println("  1. Posisi Mobil "+ posisi_awal);
        System.out.println("  2. Posisi Motor "+ posisi_awal);
        System.out.println("  3. Posisi Pesawat "+ posisi_awal);
        System.out.println("\n\n");
        System.out.println("  Posisi Kendaraan Setelah Di Jalankan ");
        System.out.println(" ---------------------------------------");
        for(int i=0;i<3;i++)
        {
            System.out.println("Posisi awal");
            System.out.println(" "+i+". "+buatKendaraan[i].getPosisi());
            buatKendaraan[i].jalan();
            System.out.println("Posisi setelah  dipanggil method jalan() ");
            System.out.println(" "+buatKendaraan[i].getPosisi());
        }
    }
}
```

Tugas Akhir Modul 4

a. Buatlah rangkuman perbedaan antara *INHERITANCE, ENKAPSULASI, DAN POLIMORFISME*
b. Buatlah program pengimplementasian *INHERITANCE* dan penginputan menggunakan *JOptionPane*
c. Buatlah program pengimplementasian *ENKAPSULASI* dan penginputan menggunakan *BufferedReader*
d. Buatlah program pengimplementasian *POLIMORFISME* dan penginputan menggunakan *Scanner*

DAFTAR PUSTAKA

Irawan. (2011). *Java Untuk Orang Awam.* Palembang: Maxikom.

Liang, Y. D. (2013). *INTRODUCTION TO JAVA PROGRAMMING COMPREHENSIVE VERSION.* USA: Prentice Hall.