**POKOK BAHASAN 3**

**CLASS**

**PENDAHULUAN**

Pada pokok bahasan ini akan dibahas mengenai objek dan class, serta penggunaannya dalam sebuah program java sederhana. Setelah memepelajari materi ini diharapkan mahasiswa mampu untuk :

1. Mengetahui pengertian dari objek & class
2. Dapat membuat program sederhana dari java dengan menggunakan objek dan class.

**PENYAJIAN (TUTORIAL)**

Class merupakan bagian pokok dari Pemrograman Berorientasi Obyek (Object Oriented Programming). Class merupakan cara merakit data-data dan menentukan semua metode yang diperlukan untuk mengakses, menggunakan dan mengubah data-data tersebut. Dalam java class juga bisa dianggap sebagai type dari object.

Setiap class memiliki dua hal utama. Yang pertama adalah yang kita sebut state/atribut/property. State dalam obyek tidak lain adalah nilai dari setiap variable. Bagian yang kedua adalah tool atau method yang menentukan utility yang dimiliki oleh class.

1. **Pendeklarasian class**

Sintaks pendeklarasian class dalam java adalah

*modifiers* class NamaClass {

<*class attribute list*>

<*class method list*>

}

*modifier* menentukan bagaimana class dapat ditangani pada fase development berikutnya. Ada tiga jenis modifier dalam class yaitu *public, final, abstract* dan *friendly.*

* *Public*

Dengan menempatkan public didepan pendeklarasian class, mempunyai arti bahwa class didefinisikan untuk public. Ini berarti class tersebut dapat dipakai atau dikembangkan (extend) oleh sembarang object.

* *Final*

Dengan menempatkan final didepan pendeklarasian class menandakan bahwa class tersebut tidak bias memiliki subclass (turunan). Dengan kata lain class tersebut tidak bias dikembangkan.

* *Abstract*

Dengan menempatkan class didepan pendeklarasian class, menandakan bahwa kita tidak bias menciptakan obyek dari class tersebut. Class tersebut harus dipakai sebagai super class dari class lain. Dalam abstract class paling tidak ada satu method yang belum komplit.

* *Friendly*

Secara default modifiers class adalah friendly ,artinya class tersebut tersebut boleh dikembangkan atau dipakai oleh class lain dalam satu paket (package) , jika kita tidak memberikan modifiers didepan pendefinisian class maka class bersifat friendly.

<*class attribute list*> adalah daftar semua atribute (variable )dari class, sedangkan <*class method list*> adalah daftar semua method (fungsi\operasi) yang bias dilakukan oleh class tersebut.

Sintaks pendefinsian class attribute :

*modifiers* data\_typenama attribute ;

Sintaks penderfinsian class method :

*modifiers*return\_typenama method (<parameters list>)

Akses modifiers dalam class attribute maupun class method menentukan bagaimana aksesbilitas attribute dan method dari class lain. Ada beberapa akes modifier yaitu:

* *Friendly*

Secara default akses modifiers adalah friendly , yang berarti bahwa attribute |mothode bias diakses oleh class lain dalam satu paket (package ).

* *Public*

Akses modifiers public membuat attribute\method tersebut dapat diakses oleh pihak luar manapun.

* *Private*

Akses modifiers private membuat attribute\method hanya dapat diaks esoleh class itu sendiri

* *Protected*

Akses modifiers protected membuat attribute\method hanya dapat diakses oleh class itu sendiri dan class turunanya (subclass nya ).

* *Final*

akses modifiers final membuat attribute tersebut tidak boleh diubah pada saat eksekusi . sebab attribute ini tidak boleh diubah maka harus diberi nilai pada saat pedeklarasian missal *final int SIZE 5*;

* *Static*

Akses modifiers static mebuat attribute tersebut mempunyai nilai sama untuk semua obyek dari class tersebut.

**2.Konstruktur**

Konstruktur adalah method dengan property yang unik dan untuk keperluan ini alisasi pada saat obyek diciptakan. Jadi konstruktur digunakan memberikan niali kepada attribute tertentu pada saat obyek diciptakan. Kontruktordidefinisikandengannama yang samadengannama class. Konstruktor tidak perlu return\_type sebab konstruktur secara actual dipanggil sebagai method. Konstruktur ini dipanggil pada saat kita memberikan perintah new.

**3. Inherintance (Pewarisan)**

Inheritance (Pewarisan) adalah suatu mekanisme yang memungkinkan satu class mewarisi (memiliki) attribute dan method dari class lain. Mekanisme pewarisan ini diperlukan untuk mengembangkan class-class yang sudah ada tanpa harus membuka source code dari class-class tersebut. Class yang mewarisi disebut subclass sedangkan calss yang diwarisi disebut super class. Pewarisan class dalam java menggunakan sintaks:

*modifier*<NamaSubClass>**extends**<NamaSuperCalss> {

}

**4. Kata Kunci *this***

***this*** adalah referensi yang menunjuk pada obyek aktual. Nilai this ini menunjukkan bahwa obyek yang dimaksud adalah obyek yang sedang aktif itu sendiri. Contoh penggunaan dari ***this*** adalah sbb:

**5. Overriding Method**

Overriding method adalah suatu mekanisme pewarisan class di mana subclass mendefinisikan ulang satu atau lebih method (fungsi) yang ada di supper class. Dalam overriding ini subclass bias mendefinisikan method (fungsi) yang di-override dengan fungsi yang baru sama sekali, atau menambahi method (fungsi) yang di-override yang sudah ada dengan fungsinal-fungsinal yang lain. Contoh class yang menggunakan mekanisme overriding :

**LEMBAR KERJA DAN TUGAS**

1. Mendefinisikan Class

public class Kotak {

private int Panjang, Lebar, Tinggi;

public void setPanjang (intpjg) {

Panjang=pjg;

}

public void setLebar (intlbr) {

Lebar=lbr;

}

public void setTinggi (inttg) {

Tinggi=tg;

}

public int getPanjang () {

returnPanjang;

}

public int getLebar () {

returnLebar;

}

public int getTinggi () {

returnTinggi;

}

public int luas () {

inthasil;

hasil=2\*Panjang\*Lebar+2\*Panjang\*Tinggi+2\*Lebar\*Tinggi;

return hasil;

}

public int volume () {

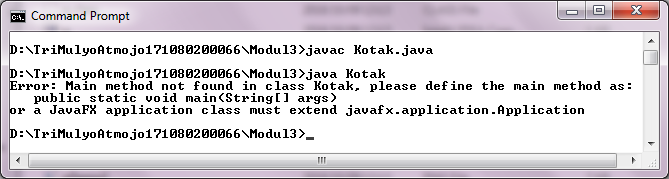
inthasil;

hasil=Panjang\*Lebar\*Tinggi;

return hasil;

}

}



public class TestKotak {

public static void main (String argv []) {

Kotak k1;

intvol,luas,pjg;

k1=new Kotak () ;

k1.setPanjang (10) ;

k1.setLebar (15) ;

k1.setTinggi (4) ;

vol=k1.volume() ;

luas=k1.luas() ;

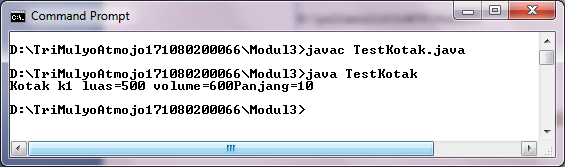
pjg=k1.getPanjang ();

System.out.println("Kotak k1 luas="+luas+" volume="+vol+"

Panjang="+pjg);

}

}



1. Menggunakan konstruktur

public class kotak2 {

private int Panjang, Lebar, Tinggi;

public kotak2 () {

Panjang=0;

Lebar=0;

Tinggi=0;

}

public kotak2(intpjg,intlbr, inttg) {

Panjang= pjg;

Lebar= lbr;

Tinggi= tg;

}

public void setPanjang(intpjg) {

Panjang= pjg;

}

public void setLebar(intlbr) {

Lebar= lbr;

}

public void setTinggi(inttg) {

Tinggi= tg;

}

public int getPanjang () {

returnPanjang;

}

public int getLebar () {

returnLebar;

}

public int getTinggi () {

returnTinggi;

}

public int luas () {

int hasil;

hasil=2\*Panjang\*Lebar+2\*Panjang\*Tinggi+2\*Lebar\*

Tinggi;

return hasil;

}

public int volume () {

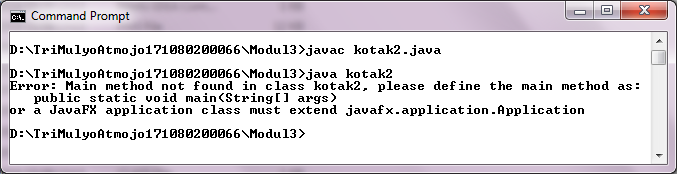
int hasil;

hasil=Panjang\*Lebar\*Tinggi;

return hasil;

}

}



public class TestKotak2 {

public static void main (String argv []) {

kotak2 k1, k2;

int vol1, luas1, pjg1, vol2, luas2, pjg2;

k1=new kotak2(10,15,4);

k2=new kotak2();

vol1=k1.volume();

luas1=k1.luas();

pjg1=k1.getPanjang();

System.out.println("Kotak2 k1 Luas = "+luas1+" volume = "+vol1+" Panjang = "+pjg1);

vol2=k2.volume();

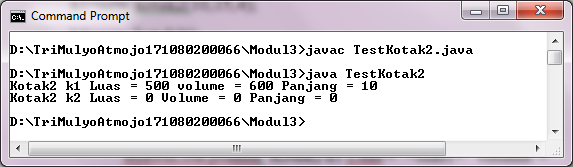
luas2=k2.luas();

pjg2=k2.getPanjang();

System.out.println("Kotak2 k2 Luas = "+luas2+" Volume = "+vol2+" Panjang = "+pjg2);

}

}



1. Menggunakan Inheritance (pewarisan)

public class KotakHitam extends kotak2 {

private String rekaman;

public KotakHitam() {

super();

rekaman="";

}

public KotakHitam(intpjg, intlbr, inttg, String rkm) {

super (pjg,lbr,tg);

rekaman=rkm;

}

public void rekam(String teks) {

rekaman=rekaman+""+teks;

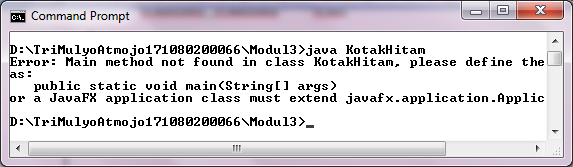
}

public void putar rekaman () {

System.out.println(rekaman);

}

}



public class TestKotakHitam {

public static void main (String argv []) {

KotakHitam k1, k2;

int vol1, luas1, pjg1, vol2, luas2, pjg2;

k1=new KotakHitam (10,15,4,"Hello Java ");

k2=new KotakHitam ();

vol1=k1.volume();

luas1=k1.luas();

pjg1=k1.getPanjang();

System.out.println("Kotak k1 Luas = "+luas1+" Volume = "+vol1+" Panjang = "+pjg1);

vol2=k2.volume();

luas2=k2.luas();

pjg2=k2.getPanjang();

System.out.println("Kotak k2 Luas = "+luas2+" Volume = "+vol2+" Panjang = "+pjg2);

k1.rekam("Ada suaraPesawatTerbang");

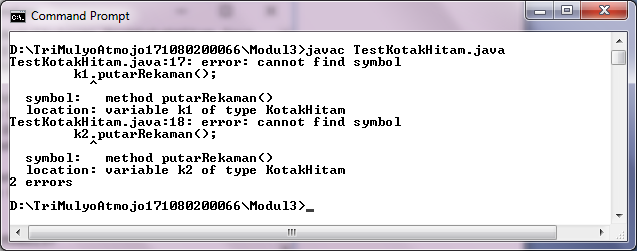
k2.rekam("Ada TandaSirenedariarahutara");

k1.PutarRekaman();

k2.putarRekaman();

}

}



1. Menggunakan kata kunci this

public class point {

privateint x, y;

public point (int x, int y) {

this.x = x;

this.y= y;

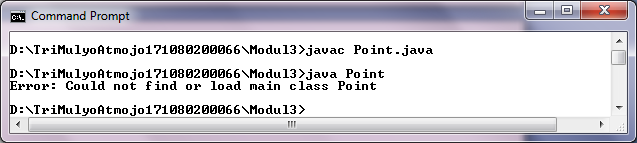
}

public point () {

this (10,10);

}

}



1. Menggunakan Overriding Method

public class p {

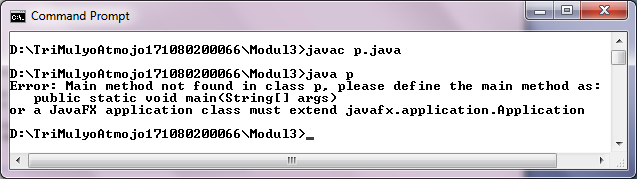
int a=5;

public void cetak () {

System.out.println("super class a="+a);

}

}



public class pDemo1 extends p {

int b=10;

public void cetak () {

System.out.println ("class aktual : b=" +b);

}

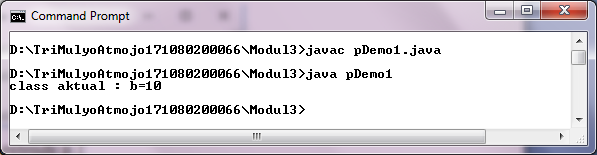
public static void main (String[] args) {

pDemo1 p=new pDemo1();

p.cetak ();

}

}



public class pDemo2 extends p {

int b=10;

public void cetak () {

System.out.println ("Class Aktual : b ="+b);

super.cetak(); //memangilcetak di super class p

}

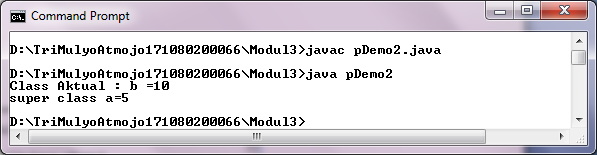
public static void main (String argv []) {

pDemo2 p=new pDemo2();

p.cetak ();

}

}



**TUGAS**

1. Buatlah class Titik yang merepresentasikantitikkoordinatduadimensi x dan y

public class Titik {

privateint x, y;

publicTitik() {

x=3;

y=2;

}

publicTitik( int x1, int y1) {

x=x1;

y=y1;

}

public void setx(int x1) {

x=x1;

}

public void sety(int y1) {

y=y1;

}

publicintgetx () {

return x;

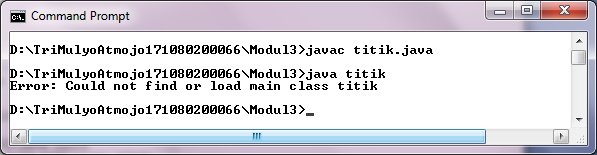
}

publicintgety () {

return y;

}

}



1. Buatlah class Titik3D turunandari class Titik yang merepresentasikantitikkoordinattigadimensi x, y dan z.

public class Titik3D extends Titik {

publicint z ;

public Titik3D () {

super ();

z = 6;

}

public Titik3D (int x1, int y1, int z1) {

super (x1,y1);

z = z1;

}

public void setz (int z1) {

z=z1;

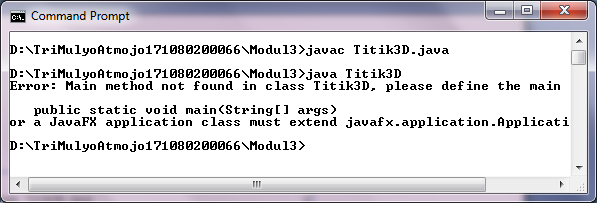
}

publicintgetz() {

return z;

}

}



public class TestTitik3D {

public static void main (String argv[]) {

Titik3D t1, t2;

int x2, y2, z2, x3, y3, z3;

t1 = new Titik3D (8,4,6);

t2 = new Titik3D ();

x2 = t1.getx();

y2 = t1.gety();

z2 = t1.getz();

System.out.println ("Koordinat ("+x2+" , "+y2+" , "+z2+")") ;

x3 = t2.getx();

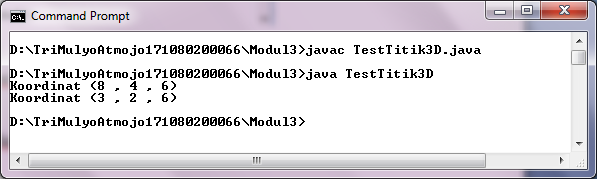
y3 = t2.gety();

z3 = t2.getz();

System.out.println ("Koordinat ("+x3+" , "+y3+" , "+z3+")");

}

}



1. Buat class konversi yang anggotanya

* Atribut
* jarak (dalam meter)
* Method
* meter ke kilo();
* kilo ke meter();

public class Konversi1 {

private int jarak;

public void setJarak (intjrk) {

jarak=jrk;

}

public int getJarak() {

return jarak;

}

public int m2k() {

int hasil;

hasil=jarak/1000;

return hasil;

}

public int k2m() {

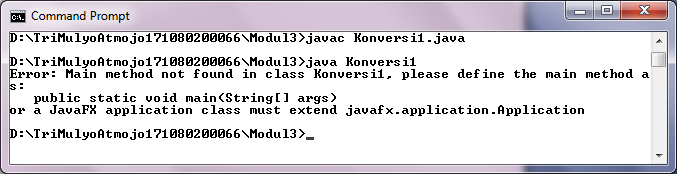
int hasil;

hasil=jarak\*1000;

return hasil;

}

}



public class TestKonversi1 {

public static void main ( String argv[]) {

Konversi1 k1;

double m2k,k2m,jrk;

k1=new Konversi1();

k1.setJarak(5000);

m2k=k1.m2k();

k2m=k1.k2m();

jrk=k1.getJarak();

System.out.println("m ke km="+m2k);

System.out.println("km ke m="+k2m);

}

}

