Konsep OOP

Pemrograman berorientasi objek (OOP) ialah sebuah metode pemrograman berbasiskan hirarki kelas-kelas dan melibatkan objek. Pemrograman Berorientasi Objek merupakan fitur utama pada Java dimana syarat utamanya kita harus membuat program berupa kelas . Sebuah kelas  ialah sebuah struktur yang mendefinisikan data dan method untuk bekerja pada data tersebut.  Ketika kita membuat program OOP di java, semua data diselubung oleh sebuah kelas. *Berikut rumus umum dari OOP :*

 //membuat objek dari suatu kelas menggunakan new

nama kelas  nama\_objek = new nama\_kelas

//memanggil method dari objeknya

nama\_objek.nama\_method();

//mengeset properties

nama\_objek.nama\_properties=value;

Kelas

Di dalam kelas kita mendeklarasikan berbagai variabel/properties dan method, contoh berikut menampilkan contoh dasar pembuatan kelas di java :

//deklarasi nama kelas StudInfo

class StudInfo {

//deklarasi variabel

   String name;

   String nim;

   String kelas;

//deklarasi dan inisialisasi method printDetails()

   void printDetails() {

        System.out.println( name + " " + nim + " " + kelas );

   }

}

//kelas utama StudentDetails

class StudentDetails {

     public static void main ( String args[] ) {

        // membuat objek bernama student dari kelas StudInfo

        StudInfo student = new StudInfo();

        // inisialisasi variable/properties objek dengan data

        student.name = "Risman Hidayat";

        student.nim = "11130076";

        student.kelas = "11.7C.01";

 // memanggil method printDetails untuk menampilkan data

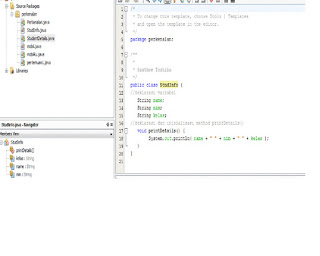
 student.printDetails();

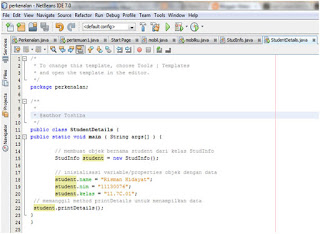
}

}

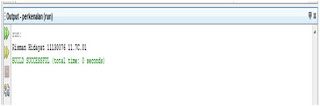
Pada program diatas, kita menggunakan 2 kelas yaitu StudentInfo yang berisi variabel dan method siswa dan kelas StudentDetails yang berisi inisialisasi variabel dan pemanggilan fungsi printDetails, dengan terlebih dahulu membuat objek dengan nama student. Perlu diperhatikan di sini, ketika anda akan menyimpan nama sebuah file, maka wajib berikan nama file dari kelas yang ada fungsi main. Dari contoh diatas , kelas yang mempunyai fungsi main ialah kelas StudentDetails, oleh karena itu nama file yang tepat ialah StudentDetails.java

Ketika program ini dijalankan ,hasilnya :

[](http://4.bp.blogspot.com/-H_ZOjjM4gf8/UVkh9HRR59I/AAAAAAAAAEM/RkPP30GwFQ4/s1600/1.jpg)

[](http://3.bp.blogspot.com/-boRsB66ki9I/UVkj1vuuq_I/AAAAAAAAAEU/Lnp3RtP5kZU/s1600/2.jpg)

Hasil Output :

[](http://1.bp.blogspot.com/-jx7-rtgfVco/UVkkfTMDg4I/AAAAAAAAAEc/FkiqaK2AW2s/s1600/3.jpg)

Inheritance

Untuk membuat kelas baru dari komponen kelas yang sudah ada, dapat kita gunakan turunan atau inheritance.  Di java kita menggunakan fungsi extendsuntuk menandakan suatu kelas ialah turunan dari kelas lain

//kelas Human ialah kelas yang akan diturunkan (base class atau super class)

class Human

{

    int hands = 2;

}

//kelas Employee ialah kelas turunan dari kelas Human (subclass)

class Employee extends Human {

     String name;

     double salary ;

//membuat konstruktor untuk inisialisasi

//idealnya untuk menginisialisasi variable kita gunakan konstruktor

Employee ( String n, double s ) {

          name = n;

          salary = s;

}

String getDetails()

{

         return "\nName : " + name + "\nSalary : " + salary;

}

}

//Kelas manager turunan dari kelas Employee

class Manager extends Employee

{

    String dept ;

   Manager ( String n, double s, String d ) {

    super(n,s);//memanggil konstruktor kelas atas (Employee)

           dept = d;

}

String getDetails()

{

    return  super.getDetails() + "\nDept : " + dept

                 + "\nHands : " + super.hands;}

 }

class SuperTestEmployee

{

public static void main ( String args[] )

{

Manager m1 = new Manager("Risman", 1000.0, "Vice President");

          System.out.println(m1.getDetails());

          Manager m2 = new Manager ("Desi", 2000.0,"Finance");

          System.out.println(m2.getDetails());

          Manager m3 = new Manager ("Aris", 3000.0, "Sales");

          System.out.println(m3.getDetails());

}

}

Hasil Output :

[](http://3.bp.blogspot.com/-uHHj_9ut1Dw/UVklOUtIZBI/AAAAAAAAAEk/lQzWjs5FdLQ/s1600/4.jpg)

Pada program diatas, kita membuat 3 buah kelas, yaitu kelas Human, Employee dan Manager . Untuk menginisialisasi variable idealnya kita menggunakan Konstruktor, dimana penulisannya harus sama dengan nama kelas. Jadi jika kelas bernama Employee , maka konstruktornya bernama Employee() atau terdiri dari beberapa parameter yang kita inginkan. Konstruktor yang tidak mempunyai parameter kita sebut sebagai default constructor. Penggunaan fungsi super untuk memanggil konstruktor dari kelas atas (base class). Jika kita membuat konstruktor dengan nama yang sama , namun berbeda isi parameternya, ini disebut sebagai overloading constructor

Contoh :

class Cload

{

String pname;

int qty;

int price;

//konstruktor dengan 3 parameter

Cload (int prodqty, String prodname, int prodprice )

{

pname=prodname;

qty=prodqty;

price=prodprice;

}

//konstruktor dengan 2 parameter bertipe int dan String

Cload (int q, String p1name)

{

pname=p1name;

price=q;

qty=price/10; }

void print() {

System.out.println ("nama produk : " + pname);

System.out.println ("Kuantitas :" + qty);

System.out.println ("Harga: " + price);}

public static void main (String args[]) {

Cload prods;

prods = new Cload (10,"Apples", 10);

prods.print();

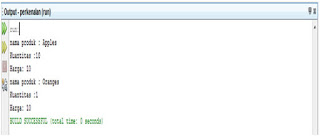
prods = new Cload (10,"Oranges");

prods.print();

}}

Program diatas akan menginisialisasi variable menggunakan konstruktor yang berbeda parameter  dan menampilkan datanya.

Hasil Output :

[](http://2.bp.blogspot.com/-zEmPjrEDN9M/UVklrUlqDII/AAAAAAAAAEs/X7Q7y5Atlgs/s1600/5.jpg)

                                                Gambar Hasil overloading constructor

Pada program diatas terlihat kita bisa mengeset konstruktor dengan berbagai parameter, dan program otomatis akan mencari dan mengenal konstruktor mana  yang harus dicari sesuai dengan parameter yang kita berikan.

Di java, agar kita dapat menggunakan method pada berbgai kelas, namun implementasi method tersebut berbeda-beda, kita harus mendeklarasikan method tersebut di dalam kelas abstract. Contoh :

abstract class Bentuk {

…

public abstract float hitungluas();

…

}

public class Lingkaran extends Bentuk {

…

float radius;

public float hitungluas();

{

return radius \*22/7;

}

}

Pada program diatas, kita memilih sebuah method abstrak yaitu hitungluas(), oleh karena itu di kelas Lingkaran yang merupakan turunan dari kelas Bentuk, dapat mendefinisikan isi method hitungluas() untuk lingkaran.  Tentu saja rumus luas untuk persegi panjang dan bidang lainnya berbeda, disinilah kelebihan penggunakan kelas abstrak. Sedangkan jika anda ingin suatu kelas tidak dapat diturunkan harus menggunakan keyword final.  Penggunaan final pada variable justru membuat variabel menjadi suatu konstanta.  Beberapa konsep OOP yang harus dipelajari Anda lebih jauh yaitu Polimorphisme, Interface dan method overriding