OOP
OBJECT ORIENTED PROGRAMMING

By Muhammad Rizqi Ariadi

Class

- Class bukanlah sebuah objek. Class digunakan untuk membangun sebuah objek dan mendefinisikan atribut dan perilaku obyek yang dibuatnya. Beberapa istilah lain class adalah template, prototype atau blue print.
- Membuat kelas harus sama dengan nama file
- Nama kelas tidak boleh mengandung simbol atau spasi

- Contoh:
- Class NamaKelas {
  - //isi kelas
  - //Statement
  - }

Attribute

- Atribut merupakan data pendukung yang dimiliki oleh kelas
- Atribut dibuat seperti membuat variable

- Contoh:

```
Class NamaKelas{
        //TipeData NamaAtribut;
        Int JumlahMahasiswa;
}
```

Object

- Atribut / Variable yang ada di dalam kelas, bisa di akses atau di panggil pada kelas yang berbeda dengan Objek.

- Membuat objek pada java, menggunakan perintah keyword "new"

  Contoh:

```
Class utama{
        NamaKelas object = new NamaKelas();
        object.warna = "merah" ;
}
```

- Untuk memanggil atribut dalam kelas bisa menggunakan TITIK(.)

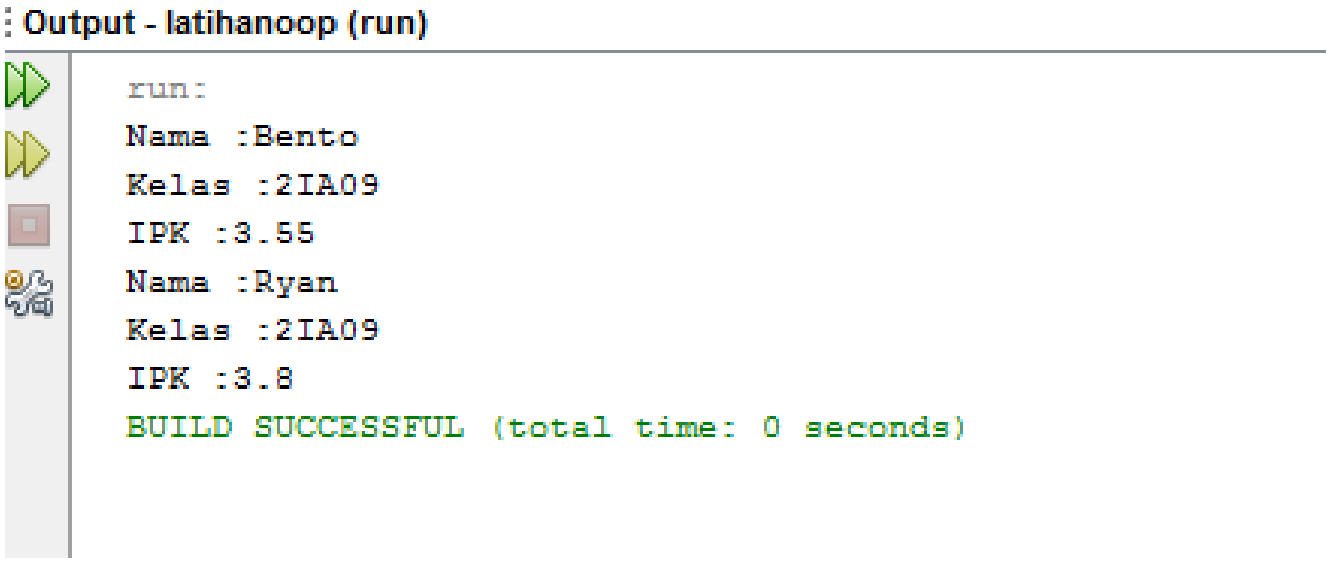
Contoh Program

```java
package latihanoop;
/**
 * @author Rizqi Muhammad
 */
public class playapp {

    public static void main(String[] args) {
        Mahasiswa objek1 = new Mahasiswa();
        Mahasiswa objek2 = new Mahasiswa();

        objek1.nama = "Bento";
        objek1.ipk = 3.55;
        objek1.kelas = "2IA09";

        objek2.nama = "Ryan";
        objek2.ipk = 3.8;
        objek2.kelas = "2IA09";

        objek1.identitas();
        objek2.identitas();
    }

}
```

```java
package latihanoop;

/**
 *
 * @author Rizqi Muhammad
 */
public class Mahasiswa {

    public String nama = " ";
    public String kelas = " ";
    double ipk = 0.0 ;

    public void identitas()
    {
        System.out.println("Nama :" + nama);
        System.out.println("Kelas :" + kelas);
        System.out.println("IPK :" + ipk);
    }


}
```

Deklarasi Object

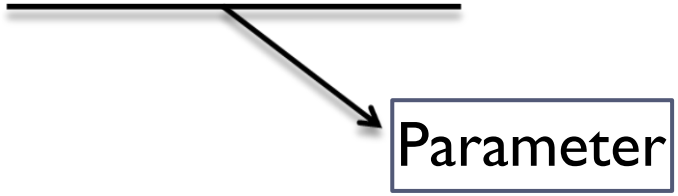
Output


Output - latihanoop (run)
run:
Nama :Bento
Kelas :2IA09
IPK :3.55
Nama :Ryan
Kelas :2IA09
IPK :3.8
BUILD SUCCESSFUL (total time: 0 seconds)

Parameter

- Parameter adalah Tipe data dan Variable Yang disisipkan dalam suatu method
- Membuat parameter:

```
Void NamaProcedure(TipeData Variable){
        //code
}

TipeData NamaFunction(TipeData Variable, TipeData Variable2){
        //code
        return (value);
}
```



This (Keyword)

- Keyword This untuk mengakses kelas itu sendiri
- Biasanya penggunaan keyword this untuk mengakses atribut kelas, jika atribut dan parameter yang ada dimethod sama.
- Contoh :

```
Class NamaKelas{
        String varNama;
        void CariNama(String nama){
        this.varNama = nama;
        }
}
```

Encapsulation

- Merupakan suatu proses membungkus sebuah attribut(variable) kedalam suatu method
- Di encapsulation variable akan disembunyikan dari class dan package lain, dan hanya dapat dirubah valuenya attributnya bila menggunakan suatu method

- Proses Encapsulation

1. Deklarasikan attributnya menggunakan hak akses private
2. Buat 2 method (per Attribut), untuk mengambil value attribut dan untuk mengubah value attributnya.

Contoh Encapsulation

```
public class NamaClass{
        private String nama;
        public String getCariNama(){
                return this.nama;
        }

        public void setNama(String nama){
                this.nama = nama;
        }
}
```

Getter untuk mengambil Value

Setter untuk mengubah Value

Contoh Program Encapsulation

```java
package encapsulation;

public class Lingkaran{
//Atribut class lingkaran yang dibuat private
    private double Jari2;

//Constructor
public Lingkaran(double Jari2){
    this.Jari2 = Jari2;
}

//Accesor
public double get_Luas(){

    double luas = Math.PI*Jari2*Jari2;
    return luas;
}

//Mutator
public void set_Jari(double Jari2){
    this.Jari2 = Jari2;
}
//accesor
public double get_Keliling(){
    double Keliling = 2.0*Math.PI*Jari2;
    return Keliling;
    }
}
```

```java
import encapsulation.Lingkaran;
import java.io.DataInputStream;


public class Main  {

    public static void main(String[] args)throws Exception {



        //membuat Inputan dengan variable I
        DataInputStream I = new DataInputStream(System.in);
        System.out.print("Masukan Jari2 : ");
        String Input = I.readLine();
        double r = Double.parseDouble(Input);
         //Instansiasi Objek
        Lingkaran a = new Lingkaran(r);

         System.out.println("Luas Lingkaran : "+a.get_Luas());
         System.out.println("Keliling Lingkaran : "+a.get_Keliling());


    }
}
```

Polimorfisme

- Objek banyak bentuk, sebuah objek bisa memiliki banyak bentuk, bisa dari class Parent (super class) namun valuenya bisa dipanggil menggunakan class turunannya
- Sebuah kelas bisa bertingkahlaku atau memiliki sifat kelas turunannya (kelas parent atau super namun bisa memiliki kelas turunannya)
- Sebuah kelas dapat disamarkan menjadi kelas parentnya (super)

- Contoh:

```
KelasParent Objek = new KelasTurunan();
```

Overriding

- Method Overriding merupakan method dari parrent class yang ditulis kembali oleh subclass
- Parameter yang terdapat pada method Overriding di subclass harus sama dengan parameter yang terdapat pada parent class.
- Aturan hak akses, hak akses method Overriding di subclass tidak boleh lebih ketat di bandingkan dengan hak akses method pada parent class.

Contoh Overriding

- Public class orangtua{

```
        public void hobi(){
        System.out.println("suka memancing");
        }
}
```

- Public class anak extends orangtua{

```
        public void hobi(){
        System.out.println("suka main bola");
        }
}
```

Overloading

- Overloading adalah kemampuan membuat dengan nama method yang sama pada satu kelas, namun berbeda parameternya (Nama method bisa sama tetapi parameter berbeda)
- Bisa juga membuat dua atau lebih method dengan jumlah parameter sama, namun tipe datanya berbeda
- Intinya setiap method harus memiliki parameter yang berbeda.
- Java mendukung fitur overloading

Contoh Overloading

- Void nama(String nama){
  //code;
  }

- Void nama(int umur){
  //code;
  }

- Void nama(String nama,int umur){
  //code;
  }

Contoh Program Overloading & Overriding

```
package polymorphism;

public class polymorphism {

    public static void main(String[] args) {
        // TODO code application logic here
    Barang brg1 = new Barang();
    brg1.DataBarang("Acer 4741G","310051");
    brg1.DataBarang("Acer 4741G","310051",4750000,20);
    Ganti brg2 = new Ganti();
    brg2.DataBarang("Acer 4741G","310051",4750000,20);
    }
}
```

Overloading

```
package polymorphism;

public class Barang {

    public void DataBarang(String nm,String kode){
        System.out.println("\n\n");
        System.out.println("Nama Barang = "+nm);
        System.out.println("Kode Barang = "+kode);
    }

    public void DataBarang(String nm,String kode,int hrg,int stok)
    {
        System.out.println("\n\n");
        System.out.println("Nama Barang = "+nm);
        System.out.println("Kode Barang = "+kode);
        System.out.println("Harga = "+hrg);
        System.out.println("Stok = "+stok);
    }
```

```
package polymorphism;

public class Ganti extends Barang{

    public void DataBarang(String nm, String kode, int hrg, int stok)
    {
    System.out.println("\n\n");
    System.out.println("Nama Barang = "+nm);
    System.out.println("Kode Barang = "+kode);
    System.out.println("Biaya = "+hrg*stok);
    }
}
```

Overriding

Contoh Program Polimorfisme

```java
package Polimorfisme;

class Hewan{
    public void bersuara(){
        System.out.println("suara biasa");
    }
}

class Kucing extends Hewan{
    public void bersuara(){
        System.out.println("Meong Miaw");
    }
}

class Burung extends Hewan{
    public void bersuara(){
        System.out.println("Cuit Cuit");
    }
}

class Ayam extends Hewan{
    public void bersuara(){
        System.out.println("Kukuruyuk");
    }
}
```

```java
public class Demo {

    public static void main(String[] args) {
        Hewan H = new Hewan();
        H.bersuara();

        Hewan K = new Kucing();
        K.bersuara();

        Hewan B = new Burung();
        B.bersuara();

        Hewan A = new Ayam();
        A.bersuara();
    }

}
```

Constructor

- Constructor adalah method khusus yang dijalankan secara otomatis saat sebuah objek dibuat
- Constructor mirip seperti method, perbedaanya dapat langsung diakses dengan hanya mendeklarasikan objeknya
- Constructor memiliki parameter dan dapat juga dioverloading

Contoh Constructor

- Pembuatan Constructor harus sesuai dengan nama kelasnya , tidak ada kata void, dan tidak ada return value
- Class user{

```
    user(String nama){
    //code;
    }
}
```

Memanggil Constructor

- NamaKelas Objek = new NamaConstructor(“Parameter”);

Contoh Program

```java
package Constructor;

public class App {

    public static void main(String[] args) {
        info objek1 = new info("ari");
        objek1.tampilkan();
    }

}
```

```java
package Constructor;

public class info {
    String nama;

    info(String nama){
        this.nama = nama;
    }

    public void tampilkan(){
        System.out.println("Nama saya :" + nama);
    }
}
```

```
run:
Nama saya :reza
BUILD SUCCESSFUL (total time: 0 seconds
```