Nama : Nur Achmad Abdillah

Nim : 1918107

Kelas : B

Tugas Mobile Programming

1. Berikan Penjelasan Tentang Konsep dasar OOP beserta contoh!

- **Pengertian OOP**

  OOP (Object Oriented Programming) adalah suatu *metode* pemrograman yang berorientasi kepada objek. Tujuan dari OOP adalah untuk mempermudah pengembangan dalam program dengan cara mengikuti model yang telah ada dalam kehidupan sehari – hari. Artinya setiap bagian dari suatu masalah adalah objek. Dan objek tersebut merupakan bagian dari beberapa objek didalamnya. Contohnya saya ibaratkan mobil. Mobil adalah objek. Dari mobil tersebut terdapat objek lagi seperti ban, mesin, spion, dll. Mobil adalah object, yang mana terdiri dari object – object lain. Dari pengertian oop dan contohnya yang saya berikan. Bisa diartikan bahwa oop merupakan object yang terdiri dari sekumpulan object kecil, yang mana object kecil tersebut sebagai pendukung dari object besar.

  Pengertian diatas merupakan contoh oop dalam kehidupan sehari – hari. Sedangkan arti oop yang sebenarnya adalah pradigma pemrograman yang berorientasikan kepada object. semua data dan fungsi dalam pradigma ini dibungkus dalam sebuah Class. Jika dibandingkan dengan pemrograman terstruktur, setiap object dapan menerima pesan, memproses pesan, dan menerima data dari object lainnya. Artinya OOP adalah pemrograman yang saling mengirim data dari satu class ke class yang lain. Data tersebut nantinya akan disimpan didalam object dari suatu class, kemudian dikirimkan ke class lain. Begitulah kira – kira. Mungkin bisa kita sebut OOP dengan Oper – operan data.

- **Konsep Dasar OOP**

  Jika ingin mempelajari OOP java, tentu kita harus mengetahui terlebih dahulu mengenai konsep oop pada java. Konsep ini adalah dasar dari OOP. Berikut ini merupakan Konsep OOP pada java:

1. **Class** = merupakan “blueprint” atau “cetakan” untuk menciptakan suatu object. Class biasa dipergunakan untuk membungkus berbagai attribute dan method yang saling berhubungan menjadi sebuah group agar lebih terorganisir sebagai satu kesatuan. Contoh : Class Kendaraan, Class Bangunan, Class Handphone, dll
2. **Object** = adalah instance dari class. Jika class secara umum merepresentasikan (template) sebuah object, sebuah instance adalah representasi nyata dari class itu sendiri.
3. **Abstraksi** = Proses representasi data dan program dalam bentuk sama dengan pengertiannya (semantik) dengan menyembunyikan rincian atau detail implementasi.
4. **Encapsulation** = Merupakan pembungkus, artinya membungkus class dan menjaga apa saja yang ada didalam class tersebut. Baik method ataupun atribut. Agar tidak bisa diakses oleh class lainnya.

5. **Polimorfisme** = (Perbedaan Bentuk) kemampuan objek yang berbeda kelas dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama dan memutuskan method mana yang akan diterapkan kepada sebuah objek.
6. **Inheritance** = Pewarisan. artinya sebuah class dapat menurunkan property dan method yang dimilikinya kepada class lain. Kemudian Inheritance sendiri mempunyai keyword yaitu keyword *extends*.
7. **Method** = suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu object. Method didefinisikan pada class akan tetapi dipanggil melalui object. Metode menentukan perilaku objek, yakni apa yang terjadi ketika objek itu dibuat serta berbagai operasi yang dapat dilakukan objek sepanjang hidupnya.

Contoh :

- *Class*

```
class Mobil {
 private String nama;
 private String jenis;
 private String warna;
 private String merk;

 public Mobil(String nama, String merk, String jenis, String warna){
  this.nama = nama;
  this.merk = merk
  this.jenis = jenis;
  this.warna = warna;
 }

 public void print_mobil(){
  System.out.println("Nama Mobil : " + nama);
  System.out.println("Merk Mobil : " + merk);
  System.out.println("Jenis Mobil : " + jenis);
  System.out.println("Warna Mobil : " + warna);
 }
}
```

program diatas merupakan contoh program untuk mendeklarasikan *Class*. Nama class tersebut adalah *Mobil*. Didalam class Mobil terdapat atribut *nama, jenis, warna, dan merk. Selain itu, terdapat construktor pada class tersebut dengan parameter (nama, merk, jenis, warna). Lalu pada program this, disitulah nilai data dari parameter dikirim ke variabel dalam class mobil. print_mobil merupakan method dalam class. Method tersebut berfungsi untuk menampilkan data dari variabel – variabel yang telah ditentukan.*

- *Object*

```
class panggil_mobil{
 private Mobil m;

 private String nama;
 private String jenis;
 private String warna;
 private String merk;
}
```

Kita bisa lihat warna, jenis, nama, dan merk merupakan *object* dari *String*. Sedangkan m merupakan object dari class *Mobil*. Object m disini berfungsi untuk memanggil tipe data dan class tersebut. Jika kita bandingkan dengan kehidupan sehari – hari, *paijo jo*. Paijo adalah nama orang, jo adalah objectnya. jadi ketika kita memanggil paijo, kita cukup panggil dengan jo.

- *Abstraksi*

```
class panggil_mobil{
 private Mobil m;

 private String nama;
 private String jenis;
 private String warna;
 private String merk;

 public void main(String[] args){
  nama = "Honda Jazz";
  jenis = "Matic";
  warna = "Silver";
  merk = "Honda";

  m = new Mobil(nama, jenis, warna, merk);
 }
}
```

kita lihat program yang paling bawah. *m = new Mobil(nama, jenis, warna, merk)* itulah yang disebut abstraksi. artinya dalam program, abstraksi tersebut mendeklarasikan class *Mobil* didalam class *jual mobil*, Untuk memanggil class dengan abstraksi kita buat rumus **object = new class(parameter)**.

- *Encapsulation*

Untuk encapsulation, kita perlu mengubah class mobil menjadi seperti dibawah ini:

```
public class Mobil{

 private String nama;
 private String jenis;
 private String merk;
 private String warna;

 public String getNama() {
  return nama;
 }

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

 public String getJenis() {
  return jenis;
 }

 public void setJenis(String jenis) {
  this.jenis = jenis;
 }

 public String getMerk() {
  return merk;
 }

 public void setMerk(String merk) {
  this.merk = merk;
 }

 public String getWarna() {
  return warna;
 }

 public void setWarna(String warna) {
  this.warna = warna;
 }

 public Mobil {
  System.out.println("Memanggil class Mobil. Ini adalah Constructor");
 }
```

Pada program diatas, kita bisa lihat terdapat dua model program (getXX dan setXX). get sendiri berfungsi untuk mengambil data, sedangkan set berfungsi untuk menambah data.

Pada encapsulation, Diwujudkan dalam bentuk class, dan didalam class sendiri terdapat method yang memiliki hak akses tertentu terhadap enviroment/ lingkungkannya. Hak akses ini disebut Acces Modifier. Yang terdiri dari private, protected, dan public.

1. **private** = Memberikan hak akses hanya kepada class tersebut.
2. **protected** = Memberikan hak akses kepada anggota classnya dan turunannya.
3. **public** = memberikan hak akses kepada property dan method agar dapat digunakan di luar class tersebut.

- *Polimorfisme*

```
class panggil_mobil{
 private Mobil m;

 public panggil_mobil(){
  m = new Mobil();
  m.setNama("Honda Jazz");
  m.setJenis("Matic");
  m.setWarna("Silver");
  m.setMerk("Honda");
 }
}
```

kita lihat dari program diatas, Pada constructor, kita panggil class mobil dengan object m. lalu dibawahnya terdapat method – method seperti setNama, setJenis dll yang notabene berasal dari class mobil. itulah yang disebut polimorfisme dimana dalam suatu class, terdapat method yang memungkinkan pemrogram menyampaikan pesan tertentu keluar dari hirarki obyeknya, dimana obyek yang berbeda memberikan tanggapan/respon terhadap pesan yang sama sesuai dengan sifat masing-masing obyek.

- *Inheritance*

```
class jual_mobil() extends panggil_mobil{
 public void main(String[] args){
  System.out.println("Nama Mobil : " + m.getNama());
  System.out.println("Jenis Mobil : " + m.getJenis());
  System.out.println("Warna Mobil : " + m.getWarna());
  System.out.println("Merk Mobil : " + m.getMerk());
 }
}
```

Pada contoh program diatas, saya membuat class jual_mobil dengan turunan dari panggil_mobil. m.getnama() dll merupakan milih class mobil, namun telah dipanggil di class mobil, sehingga class jual_mobil bisa mengaksesnya tanpa perlu mendeklarasikan class mobil terlebih dahulu.

2. Buatlah program sederhana menggunakan konsep OOP dengan menggunakan bahasa pemrograman

1. Script Program

Super Class (Class Mobil)

```
public abstract class Mobil {
    String Tahun,Nama,Harga;
    public String MembeliMobil(){
         return ("1. Pergi Ke dealer\n"+
                 "2. Pilih Mobil Sesuai Kebutuhan.\n" +
                 "3.  Tentukan  Waktu  yang  Tepat  Membeli
Mobil.\n" +
                 "4. Tawar Harga di Bawah Bujet.\n" +
```

```
                "5. Jangan Pernah Menyebut Bujet Anda.\n" +
                "6. Jangan Diskusikan Pembayaran bila Harga
Belum Cocok.\n"+
                "7. Kalau udah cocok GASSSS BELI AJA");
     }
     abstract String Menyalakan_Mobil();
     abstract int Harga();
}
```

Sub Class (Class Fiat500)

```
public class Fiat500 extends Mobil {
    @Override
    public String MembeliMobil() {
        return super.MembeliMobil();
    }

    @Override
    String Menyalakan_Mobil() {
        return ("1. Masukkan kunci ke dalam kontak, Nyalakan
kontak\n "+
                "2. Putar kunci di dalam kontak dan hidupkan
mesin mobil");
    }

    @Override
    int Harga() {
       return 700000000;
    }
    public static void main(String[]args){
        Fiat500 M = new Fiat500();
      M.Nama = "Fiat500";
      M.Tahun = "1957";
      System.out.println("==============");
      System.out.println("Nama  :"+M.Nama);
      System.out.println("Tahun :"+M.Tahun);
      System.out.println("==============");
      System.out.println("Membeli                          Mobil
:\n"+M.MembeliMobil());
      System.out.println("==============");
      System.out.println("Menyalakan                       Mobil
:\n"+M.Menyalakan_Mobil());
      System.out.println("Harga            :"+M.Harga());
    }
}
```

Sub Class (Class Morris)

```
public class Morris extends Mobil {

    @Override
    public String MembeliMobil() {
        return super.MembeliMobil();
    }
     @Override
```

```
    String Menyalakan_Mobil() {
        return ("1. Masukkan kunci ke dalam kontak, Nyalakan
kontak\n "+
                "2. Putar kunci di dalam kontak dan hidupkan
mesin mobil");
    }

    @Override
    int Harga() {
       return 500000000;
    }
    public static void main(String[]args){
        Morris R = new Morris();
      R.Nama = "Morris";
      R.Tahun = "1961";
      System.out.println("==============");
      System.out.println("Nama  :"+R.Nama);
      System.out.println("Tahun :"+R.Tahun);
      System.out.println("==============");
      System.out.println("Membeli                          Mobil
:\n"+R.MembeliMobil());
      System.out.println("==============");
      System.out.println("Menyalakan                       Mobil
:\n"+R.Menyalakan_Mobil());
      System.out.println("Harga            :"+R.Harga());
    }
}
```

Sub Class (Class Rolls_Roys)

```
public class Rolls_Roys extends Mobil{
    @Override
    public String MembeliMobil() {
        return super.MembeliMobil();
    }
     @Override
    String Menyalakan_Mobil() {
        return ("1. Masukkan kunci ke dalam kontak, Nyalakan
kontak\n "+
                "2. Putar kunci di dalam kontak dan hidupkan
mesin mobil");
    }

    @Override
    int Harga() {
       return 1500000000;
    }
    public static void main(String[]args){
        Rolls_Roys S = new Rolls_Roys();
      S.Nama = "Rolls Roys";
      S.Tahun = "2015";
      System.out.println("==============");
      System.out.println("Nama  :"+S.Nama);
      System.out.println("Tahun :"+S.Tahun);
      System.out.println("==============");
      System.out.println("Membeli                          Mobil
:\n"+S.MembeliMobil());
      System.out.println("==============");
      System.out.println("Menyalakan                       Mobil
:\n"+S.Menyalakan_Mobil());
```

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

2. Hasil Program
   - Class Fiat500

```
==============
Nama  :Fiat500
Tahun :1957
==============
Membeli Mobil :
1. Pergi Ke dealer
2. Pilih Mobil Sesuai Kebutuhan.
3. Tentukan Waktu yang Tepat Membeli Mobil.
4. Tawar Harga di Bawah Bujet.
5. Jangan Pernah Menyebut Bujet Anda.
6. Jangan Diskusikan Pembayaran bila Harga Belum Cocok.
7. Kalau udah cocok GASSSS BELI AJA
==============
Menyalakan Mobil :
1. Masukkan kunci ke dalam kontak, Nyalakan kontak
 2. Putar kunci di dalam kontak dan hidupkan mesin mobil
Harga             :700000000
BUILD SUCCESSFUL (total time: 0 seconds)
```

   - Class Morris

```
run:
==============
Nama  :Morris
Tahun :1961
==============
Membeli Mobil :
1. Pergi Ke dealer
2. Pilih Mobil Sesuai Kebutuhan.
3. Tentukan Waktu yang Tepat Membeli Mobil.
4. Tawar Harga di Bawah Bujet.
5. Jangan Pernah Menyebut Bujet Anda.
6. Jangan Diskusikan Pembayaran bila Harga Belum Cocok.
7. Kalau udah cocok GASSSS BELI AJA
==============
Menyalakan Mobil :
1. Masukkan kunci ke dalam kontak, Nyalakan kontak
 2. Putar kunci di dalam kontak dan hidupkan mesin mobil
Harga             :500000000
BUILD SUCCESSFUL (total time: 0 seconds)
```

   - Class Rolls_Roys

```
run:
==============
Nama  :Rolls Roys
Tahun :2015
==============
Membeli Mobil :
1. Pergi Ke dealer
2. Pilih Mobil Sesuai Kebutuhan.
3. Tentukan Waktu yang Tepat Membeli Mobil.
4. Tawar Harga di Bawah Bujet.
5. Jangan Pernah Menyebut Bujet Anda.
6. Jangan Diskusikan Pembayaran bila Harga Belum Cocok.
7. Kalau udah cocok GASSSS BELI AJA
==============
Menyalakan Mobil :
1. Masukkan kunci ke dalam kontak, Nyalakan kontak
 2. Putar kunci di dalam kontak dan hidupkan mesin mobil
Harga             :1500000000
BUILD SUCCESSFUL (total time: 0 seconds)
```