Nama : **Nuryadin Abutani**

NPM : **19552011182**

Kelas : **TIF RP 19A**

Prodi : **Teknik Informatika**

Matkul : **Object Oriented Programming I**

Daftar isi:

1. OOP

Pemrograman berorientasikan objek (*Object Oriented Programming*) merupakan sebuah paradigma atau teknik pemrograman yang berorientesikan Objek. Fungsi dan variabel dibungkus dalam sebuah objek atau *class* yang dapat saling brinteraksi, sehingga membentuk sebuah program.

1) Variabel disebut atribut atau properti;
2) Fungsi disebut method.

2. Class

Cetak biru (blue print) dari objek dimana sebuah class menggambarkan ciri-ciri objek secara umum. Nama_kelas harus sesuai dengan nama file.

```
Class nama_kelas {
      \\isi dari kelas
}
```

3. Object

Sebuah variabel yang merupakan *instance* atau perwujudan dari *Class*.

4. Atribut

Ciri-ciri yang melekat pada suatu objek.

```
[acces_modifier [tipe_data] [nama_variabel] = [value];
```

5. Method

Fungsi-fungsi yang digunakan untuk memanipulasi nilai-nilai pada atribut dan/atau untuk melakukan hal-hal yang dapat dilakukan oleh objek itu sendiri. Berisi sekumpulan program yang telah terbungkus.

```
[acces_modifier] [tipe_data] nama_method()
```

Bisa memanggil kumpulan program hanya dengan memanggil nama methodnya.

```
setValue();
getValue();
```

6. This

Digunakan sebagai referensi dari class itu sendiri. Membedakan variabel yang dideklarasikan pada parameter di dalam method dengan variabel yang dideklarasikan pada class.

```
class nama_class {
      [acces_modifier] [tipe_data] ] [nama_variabel];
      [acces_modifier] [tipe_data] nama_method([tipe_data]
[nama_parameter]) {
            this.[nama_variabel] = [nama_parameter];
      }
}
```

7. Access Modifier

Memberi batasan hak class maupun method.

Aksesabilitas	private	default	protected	public
Dari class yang sama	**Ya**	**Ya**	**Ya**	**Ya**
Dari package yang sama	**-**	**Ya**	**Ya**	**Ya**
Dari package yang berbeda (subclass)	**-**	**-**	**Ya**	**Ya**
Dari package yang berbeda (nonsubclass)	**-**	**-**	**-**	**Ya**

8. Instance of Class

Objek yang diinstan atau dibuat dari class.

```
[nama_class] [nama_objek] = nama_class(…);
```

9. Inheritance

Sifat atau konsep pewarisan yang bisa mengakses sifat ataupun method dari class lain, dengan memakai kata kunci extends pada class anak/subclass.

```
[acces_modifier] class [nama_subclass] extends [nama_superclass]
{
      \\isi dari kelas
}
```

1) **Superclass** : class asal (orang tua)
2) **Subclass** : class turunan (anak)

10. Constructor

Method khusus yang akan dieksekusi pada saat pembuatan objek *(instance)*. Nama_method harus sesuai dengan nama nama_class constructor dan tidak boleh mengembalikan nilai.

```
[acces_modifier] nama_method(…)  {
      System.out.println("eksekusi method constructor...");
 }
```

11.Super

Merepresentasikan objek dari class induk.

```
class nama_mainclass {
      [acces_modifier] [tipe_data] ] [nama_variabel_mainclass] =
[value];
}
…
class [nama_subclass] extends [nama_ mainclass] {
       [acces_modifier] [tipe_data] nama_method() {
            System.out.println(super.[nama_variabel_mainclass]);
      }
}
```

12.Setter & Getter

1) method yang tidak mengembalikan nilai biasanya berupa sub program berjenis prosedur.

```
public void setUsername(String username){
        this.username = username;
    }
```

2) Method yang mengembalikan nilai biasanya berupa sub program berjenis fungsi .

```
public String getUsername(){
        return this.username;
    }
```

13.Overloading Method

Sebuah class yang memiliki nama method yang sama tapi memiliki parameter dan tipe data yang berbeda. “Dalam satu class”, “Nama method sama”, “Tipe data dan parameter beda”.

```
class Lingkaran {


    // method menghitung luas dengan jari-jari
    float luas(float r){
        return (float) (Math.PI * r * r);
    }


    // method menghitung luas dengan diameter
    double luas(double d){
        return (double) (1/4 * Math.PI * d);
```

14.Overidding method (inheritence)

fungsi atau mathod dari superclass (kelas induk) yang ditulis kembali pada subclassnya (kelas anak).

1) Parameter pada mothod overriding di subclass harus sama dengan parameter yang terdapat pada superclass.
2) Aturan hak akses pada fungsi overriding di subclass tidak boleh lebih ketat di bandingkan dengan hak akses method pada superclass.
3) Method Overriding dapat dibuat dengan menambahkan anotasi `@Override` di atas nama method atau sebelum pembuatan method