MODUL 8
FINAL & ABSTRACT

8.1 Final Method dan Final Class

Keyword "final" digunakan untuk mencegah suatu class diturunkan atau suatu method di overridding atau suatu variable diubah.

Sintaks penggunaan keyword "*final*" pada class :

```
akses_modifier final namaClass
```

Sintaks penggunaan keyword "*final*" pada method :

```
akses_modifier final tipeMethod namaMethod()
{
      ..... // definisi method
```

Sintaks penggunaan keyword "*final*" pada variable (konstanta) :

```
akses_modifier final tipeData namaVariable
```

8.2. Abstract Class dan Abstract Method

Abstract Class adalah sebuah class yang tidak bisa di-*instansiasi* (tidak bisa dibuat menjadi objek) dan berperan sebagai '*kerangka dasar'* bagi class turunannya. Di dalam *abstract class* umumnya akan memiliki *abstract method*. Cara untuk membuat sebuah *abstract class* adalah :

```
akses_modifier abstract class namaClassAbstrak
{
      ......... // definisi class
```

Abstract Method adalah sebuah '*method dasar*' yang harus diimplementasikan ulang di dalam class anak (*child class*). *Abstract method* ditulis tanpa isi dari *method*, melainkan hanya '**signature'**-nya saja. **Signature** dari sebuah *method* adalah bagian method yang terdiri dari nama method dan parameternya (jika ada). Cara untuk membuat sebuah *abstract method* adalah :

```
abstract void move(double x, double y);
```

Abstract class digunakan di dalam *inheritance* (*pewarisan class*) untuk '*memaksakan'* implementasi method yang sama bagi seluruh class yang diturunkan dari *abstract class*. *Abstract class* digunakan untuk membuat struktur logika penurunan di dalam pemrograman objek.

Java memiliki aturan-aturan dalam penggunaan method abstrak dan class abstrak sebagai berikut:

a) Class yang di dalamnya terdapat abstract method harus dideklarasikan sebagai abstract class.
b) Abstract class tidak dapat diinstansi, tetapi harus di turunkan.
c) Abstract class tidak dapat diinstansi (menjadi objek dari class abstract), tetapi kita dapat mendeklarasikan suatu variable yang bertipe abstract class dan membuat instansi dari variabel tersebut yang bertipe class turunan dari abstract class tersebut (*teknik polymorphism*).
d) Sebuah class dapat dideklarasikan sebagai abstract class meskipun class tersebut tidak memiliki abstract method.
e) Abstract method tidak boleh mempunyai body method dan demikian juga sebaliknya bahwa method yang tidak ditulis body methodnya maka harus dideklarasikan sebagai abstract method.

Contoh source code :

Laptop.java

```
1  // abstract class
2  public abstract class Laptop {
3      // attribut class
4      protected String merk,pemilik;
5
6      // abstract method
7      protected abstract void setMerk(String merk);
8      protected abstract String getMerk();
9      protected abstract void setPemilik(String pemilik);
10     protected abstract String getPemilik();
11     protected abstract void tampil();
12     protected abstract void hapus();
13
14     // method "biasa"
15     protected void hidupkanLaptop(){
16        System.out.println("Hidupkan Laptop");
17    }
18 }
```

LaptopAsus.java

```java
public class LaptopAsus extends Laptop {
    // constructor
    LaptopAsus(String merk)
    {   setMerk(merk);
        merk = null;
    }

    protected void setMerk(String merk) {
        this.merk = merk;
        merk = null;
    }
    protected String getMerk() {
        return merk;
    }

    protected void setPemilik(String pemilik)
    {   this.pemilik = pemilik;
        pemilik = null;

    }
    protected String getPemilik()
    {   return this.pemilik;
    }

    protected void tampil()
    {   System.out.println(getPemilik()+" memiliki laptop merk "+getMerk());
    }
    protected void hapus()
    {   merk = null;
        pemilik = null;
    }
}
```

LaptopDell.java

```java
public class LaptopDell extends Laptop {
    // constructor
    LaptopDell(String merk)
    {   setMerk(merk);
        merk = null;
    }

    protected void setMerk(String merk) {
        this.merk = merk;
        merk = null;
    }
    protected String getMerk() {
        return merk;
    }

    protected void setPemilik(String pemilik)
    {   this.pemilik = pemilik;
        pemilik = null;

    }
    protected String getPemilik()
    {   return this.pemilik;
    }

    protected void tampil()
    {   System.out.println(getPemilik()+" memiliki laptop merk "+getMerk());
    }
    protected void hapus()
    {   merk = null;
        pemilik = null;
    }
}
```

LaptopToshiba.java

```java
public class LaptopToshiba extends Laptop {
    // constructor
    LaptopToshiba(String merk)
    {   setMerk(merk);
        merk = null;
    }

    protected void setMerk(String merk) {
        this.merk = merk;
        merk = null;
    }
    protected String getMerk() {
        return merk;
    }

    protected void setPemilik(String pemilik)
    {   this.pemilik = pemilik;
        pemilik = null;

    }
    protected String getPemilik()
    {   return this.pemilik;
    }

    protected void tampil()
    {   System.out.println(getPemilik()+" memiliki laptop merk "+getMerk());
    }
    protected void hapus()
    {   merk = null;
        pemilik = null;
    }
}
```

LaptopCetak.java

```
// final class
public final class LaptopCetak {
    // final variable/konstanta
    private final String barang = "Laptop";
    // final method
    private final void cetak(Laptop[] ob, String pemilik) {
        System.out.println("Nama Barang : "+barang);
        System.out.println("");
        // polimorfisme
        for (int i=0;i<ob.length;i++)
         {  ob[i].getMerk();
            ob[i].setPemilik(pemilik);
            ob[i].getPemilik();
            ob[i].tampil();
            ob[i].hapus();
            System.out.println("########################");
         }
        ob = null;
        pemilik = null;
    }

    public static void main (String []args) {
        String pemilik ="Ahmad";
        Laptop[] ob =  {  new LaptopAsus("Asus"),
                          new LaptopDell("Dell"),
                          new LaptopToshiba("Toshiba")
                       };

        LaptopCetak abc = new LaptopCetak();
        abc.cetak(ob,pemilik);

        pemilik = null;
        ob = null;
        abc = null;
     }
}
```

MODUL 8

LATIHAN SOAL

1. Buatlah aplikasi penjumlahan, pengurangan, perkalian dan pembagian menggunakan konsep abstract, final dan polimorfisme dengan memasukkan parameter bilangan A = 6.5 dan bilangan B = 0.5 !