PERTEMUAN 14

INTERFACE , ABSTRACT CLASS DAN ABSTRACT METHOD

Interface

Interface

- Interface mendefinisikan aturan perilaku yang dapat di implementasikan oleh kelas manapun. Interface mendefinisikan satu set method tanpa menyediakan implementasinya.
- Setiap kelas yang mengimplementasikan interface akan terikat oleh interface tersebut untuk mengimplementasikan semua method yang ada di dalam interface.

Interface (Lanjutan)

- Secara subtansi Interface merupakan kumpulan dari method abstrak dan konstanta.
- Interface memiliki kemiripan dengan kelas abstrak karena keduanya memuat method abstrak.
- Untuk menjadikan sebuah kelas mengimplementasikan interface maka ditambahkan keyword **implements** kemudian dilanjutkan nama interface.

Mengapa menggunakan interface?

- Mendeklarasikan *method* yang akan diimplementasikan oleh satu atau beberapa kelas
- Menangkap kesamaan di antara beberapa kelas tanpa perlu memasukkannya dalam hirarki kelas.
- Mensimulasikan konsep pewarisan banyak kelas dengan mendeklarasikan kelas yang mengimplementasikan beberapa interface sekaligus

```
public interface interfaceElektronik
{
  public void on();
  public void off();
}
```

```
public class Radio implements interfaceElektronik
{
    boolean mesin=false;
    String[] channel={"Gajah Mada FM","Smart
  FM","Buana FM","DINUS FM"};
    int volume=0;

    public void on()
    {
        mesin=true;
    }
    public void off()
    {
        mesin=false;
    }
```

… Continue

```
public class Kipas implements interfaceElektronik
  {
     boolean mesin=false;
     int kecepatanKipas=0;

     public void on()
     {
         mesin=true;
     }
     public void off()
     {
         mesin=false;
     }

… Continue
```

Kelas Abstrak

Sebuah kelas dapat dideklarasikan sebagai kelas abstrak. Tujuanya :

1. Agar suatu kelas ***tidak dapat di instansiasikan*** sebagai sebuah objek dan hanya dapat diturunkan.
2. Agar satu kelas lain dapat memperluasnya dengan jalan menjadi subclass darinya

Untuk membuat sebuah kelas atau method menjadi abstrak maka menggunakan keyword **abstract** setelah modifier pada deklarasi kelas atau method. Contoh kelas abstrak

```
public abstract class Unggas
{

}
```

Method Abstrak

- Method abstrak
  - Method abstrak adalah method yang memiliki definisi namun tidak memiliki implementasi
  - Method di dalam abstract class boleh berupa method abstrak ataupun bukan.
  - Untuk membuat abstract method, hanya menuliskan deklarasi method tanpa body dan gunakan keyword abstract

```
public abstract class animal {
  private int jmlKaki;
  public abstract void walk();
  public abstract void sound();
}
```

```
class kucing extends animal
{
    public void walk()
    {
        System.out.println("Berjalan dengan 4 kaki");
    }
    public void sound()
    {
        System.out.println("Meeooo...ong");
    }
}
```

```
class ayam extends animal
{
    public void walk()
    {
        System.out.println("Berjalan dengan 2 kaki");
    }
    public void sound()
    {
        System.out.println("Kuku Ruyuuuukk");
    }
}
```

- Menggunakan abstract class untuk mendefinisi jenis-jenis yang luas dari behavior yang ada di puncak hirarki class object-oriented programming, dan menggunakan subclassnya untuk menyediakan detail implementasi dari abstract class.

Perbedaan Interface dan Kelas Abstrak

Interface	Kelas Abstrak
Tidak dapat membuat implementasi method	Dapat membuat implementasi method
Sebuah kelas dapat mengimplementasikan beberapa interface	Sebuah kelas hanya dapat meng-Extends satu superclass.

Contoh Abstract Class dan Abstract Method

```
import java.text.DecimalFormat;
import java.util.Scanner;
abstract class htl{
    private String nama;
    htl(String nama){
       this.nama=nama;
    }
    public String namahotel(){
        return nama;
    }
    public abstract double totalbayar();
    public abstract void hotel();
}
class hotel1 extends htl{
    private int lamamenginap;
    private double hargakamar;
    public hotel1(String nama, int lamamenginap, double hargakamar){
        super(nama);
        setlama(lamamenginap);
        setharga(hargakamar);
    }
    public void hotel(){
        System.out.println("=====HOTEL=======");
    }
    public void setlama(int lama1){
        lamamenginap=lama1;
    }
```

(....Lanjutan) Contoh Abstract Class dan Abstract Method

```
    public void setlama(int lama1){
        lamamenginap=lama1;
    }
    public void setharga(double harga1){
        hargakamar=harga1;
    }
    public String nama(){
        return super.namahotel();
    }
    public int lama_menginap(){
        return lamamenginap;
    }
    public double harga_kamar(){
        return hargakamar;
    }
    public double totalbayar(){
        return (lamamenginap*hargakamar);
    }
}

public class hotel{
    public static void main(String[]args){
        Scanner sc=new Scanner(System.in);
        DecimalFormat df=new DecimalFormat("0,000");
        System.out.println("======HOTEL======");
        System.out.print("\nNama :");
        String nama= sc.nextLine();
```

(....Lanjutan) Contoh Abstract Class dan Abstract Method

```
public class hotel{
    public static void main(String[]args){
        Scanner sc=new Scanner(System.in);
        DecimalFormat df=new DecimalFormat("0,000");
        System.out.println("======HOTEL======");
        System.out.print("\nNama :");
        String nama= sc.nextLine();
        System.out.print("Lama Menginap :");
        int lamamenginap= sc.nextInt();
        System.out.print("Harga Kamar :");
        double hargakamar=sc.nextDouble();
        System.out.println("Total Bayar :");
        double totalbayar=lamamenginap * hargakamar;
        hotel1 h=new hotel1(nama,lamamenginap,hargakamar);
        h.hotel();
        System.out.println("Nama : "+  h.nama());
        System.out.println("Lama Menginap : "+h.lama_menginap()+" hari");
        System.out.println("Harga Kamar : "+df.format(h.harga_kamar()));
        System.out.println("Total Bayar : "+df.format(h.totalbayar()));

}
}
```