Modul Praktikum

PBO

6

ASCii

MODUL 6 ABSTRACT & KEYWORD FINAL

1. ABSTRACT CLASS

Abstract Class adalah sebuah class yang tidak bisa di-instansiasi (tidak bisa dibuat menjadi objek) dan berperan sebagai 'kerangka dasar' bagi class turunannya. Abstract Class juga class yang mempunyai setidaknya satu Abstract Method.

Untuk membuat Abstract Class, cukup gunakan kata kunci abstract sebelum kata kunci class, deklarasi Abstract Class bisa dibuat dengan contoh dibawah :

```java
public abstract class hewan {
    public abstract void suaraHewan();

    public void tidur() {
        System.out.println("Zzzz...");
    }
}
```

Dari contoh di atas, tidak mungkin membuat objek kelas hewan.

```
1   public class LatihanAbstract {
        Run | Debug
2       public static void main(String[] args) {
3           hewan myObj = new Hewan();
    LatihanAbstract.java 1 of 2 problems
    hewan cannot be resolved to a type Java(16777218)
4       }
```

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 :

```
1  public abstract class hewan { //akses_modifier abstract class nama_class
2      public abstract void suaraHewan(); //akses_modifier abstract method nama_method
3
4      public void tidur() {
5          System.out.println("Zzzz...");
6      }
7  }
```

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.

2. KEYWORD FINAL

Keyword “final” digunakan untuk mencegah suatu class diturunkan atau suatu method di overridding atau suatu variable diubah. Sintaks penggunaan keyword “final”: ○ NullPointerException

```
public final class LatihanFinal {// Penggunaan Final pada Class
    public final String namaHewan = "Sapi";// Penggunaan Final pada Variabel

    public final void tidur() {// Penggunaan Final pada Method
        System.out.println("Zzz...");
    }
}
```

Contoh program:

File Laptop.java

```
public abstract class Laptop {
    protected String merk,pemilik;

    //abstract method
    protected abstract void setMerk(String merk);
    protected abstract String getMerk();
    protected abstract void setPemilik(String pemilik);
    protected abstract String getPemilik();
    protected abstract void tampil();
    protected abstract void hapus();

    //usual method
    protected void hidupkanLaptop(){
        System.out.println("Hidupkan Laptop");
    }
}
```

File LaptopAsus.java

```
3  public class LaptopAsus extends Laptop{
4      //constructor
5      LaptopAsus (String merk){
6          setMerk(merk);
7          merk=null;
8      }
9
10     protected void setMerk(String merk){
11         this.merk=merk;
12         merk=null;
13     }
14     protected String getMerk(){
15         return merk;
16     }
17
18     protected void setPemilik(String pemilik){
19         this.pemilik=pemilik;
20         pemilik=null;
21     }
22     protected String getPemilik(){
23         return pemilik;
24     }
25     protected void tampil(){
26         Systemt.println(getPemilik()+" Memiliki Laptop Merk "+getMerk());
27     }
28     protected void hapus(){
29         merk=null;
30         pemilik=null;
31     }
32 }
```

File LaptopAcer.java

```
3  public class LaptopAcer extends Laptop{
4      //constructor
5      LaptopAcer (String merk){
6          setMerk(merk);
7          merk=null;
8      }
9
10     protected void setMerk(String merk){
11         this.merk=merk;
12         merk=null;
13     }
14     protected String getMerk(){
15         return merk;
16     }
17
18     protected void setPemilik(String pemilik){
19         this.pemilik=pemilik;
20         pemilik=null;
21     }
22     protected String getPemilik(){
23         return pemilik;
24     }
25     protected void tampil(){
26         Systemt.println(getPemilik()+" Memiliki Laptop Merk "+getMerk());
27     }
28     protected void hapus(){
29         merk=null;
30         pemilik=null;
31     }
32 }
```

File Main.java

```java
public final class Main {
    private final String barang = "Laptop";

    // final method
    private final void cetak(Laptop[] ob, String pemilik) {
        System.out.println("Nama Barang : " + barang);
        System.out.println("");
        // polymorph
        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 = "PBO MANTAP";
        Laptop[] ob = {
                new LaptopAsus("Asus"),
                new LaptopAcer("Acer"),
        };
        Main abc = new Main();
        abc.cetak(ob, pemilik);
        pemilik = null;
        ob = null;
        abc = null;
    }
}
```