Modul

Encapsulation dan Abstraction

1. Tujuan Praktikum

- Memahami proses enkapsulasi
- Memahami fungsi dan cara mengimplementasikan interface

2. Alat dan Bahan

- Teks Editor
- Java JDK
- Command Prompt

3. Teori Dasar

3.1 Encapsulation

Encapsulation adalah proses pembungkusan data dari suatu kelas untuk menyembunyikan data yang sensitive dari kelas lain. Enkapsulasi juga biasa disebut *information hiding*. Dalam enkapsulasi, variabel pada suatu kelas akan disembunyikan dari kelas lain. Artinya, variable pada suatu kelas tidak dapat di akses atau di modifikasi secara langsung dari kelas lain. Variabel hanya dapat diakses melalui method tertentu pada kelas tersebut. Dalam proses enkapsulasi, terdapat dua jenis method yang biasa digunakan, yaitu:

- Setter: Digunakan untuk mengatur nilai dari suatu variable
- Getter: Digunakan untuk mendapatkan nilai dari suatu variabel

3.2 Abstraction

Abstraksi data adalah proses menyembunyikan detail tertentu dan hanya menampilkan informasi penting kepada pengguna. Dalam bahasa pemrograman Java, abstraksi dapat dilakukan menggunakan kelas abstrak dan interface.

- Abstract Class

  Abstract class adalah sebuah class yang tidak dapat di instansiasi (tidak dapat dijadikan objek) dan berperan sebagai kerangka dasar bagi class turunannya. Syntax penulisan abstract class adalah sebagai berikut:

```
Access_modifier abstract class namakelas{
// isi kelas
}
```

Berikut adalah beberapa poin yang perlu diketahui mengenai kelas abstrak:

1) Dapat meng-extends kelas atau meng-implement interface.
2) Seperti kelas induk secara umum, dapat diakses oleh kelas turunan dengan meng-extends kelas induk.
3) Kelas abstrak dapat menggunakan keyword abstract dan non abstract pada methodnya.
4) Kelas turunan tidak dapat meng-extends lebih dari satu kelas abstract.
5) kelas abstract dapat menggunakan konstruktor dan main method

- Interface

Interface merupakan sebuah tipe referensi pada Java. Interface adalah class yang tidak memiliki tubuh pada method-methodnya. Method pada kelas yang mengimplementasikan suatu interface harus sama persis dengan method yang ada pada interface tersebut. Syntax penulisan interface adalah sebagai berikut:

```
Access_modifier interface namainterface{
// isi interface
}
```

Berikut adalah beberapa point yang perlu diketahui mengenai interface:

1) Menggunakan keyword "implement" untuk mengimplementasi pada kelas turunan.
2) Interface hanya dapat memiliki abstract pada methodnya.
3) Interface dapat di implementasikan lebih dari satu interface pada kelas turunannya.
4) Interface hanya dapat memiliki final dan static variable, artinya nilai variable pada interface tidak dapat diubah.
5) Interface tidak dapat meng-extends abstrack kelas tapi dapat meng-extends interface lain.
6) Method dan field pada interface akan selalu bersifat public
7) Dalam interface tidak dapat menggunakan konstruktor dan main method

4. Langkah Praktikum

4.1 Enkapsulasi

1. Buat kelas baru dengan nama Buku seperti kode berikut

```
public class Buku{

}
```

2. Tambahkan variable judul dan pengarang dengan access modifier private seperti kode berikut

```
…
   private String judul;
   private String pengarang;
…
```

3. Tambahkan method setter untuk memberi nilai pada variable judul dan pengarang seperti kode berikut:

```
…
   public void setJudul(String judul){
        this.judul = judul;
   }
   public void setPengarang(String pengarang){
        this.pengarang = pengarang;
   }
…
```

4. Tambahkan method getter untuk mengembalikan nilai dari variable judul dan pengarang seperti kode berikut

```
…
   public String getJudul(){
        return judul;
   }
   public String getPengarang(){
        return pengarang;
   }
…
```

5. Buat kelas baru dengan nama BukuMain seperti kode berikut:

```
public class BukuMain {
   public static void main(String[] args) {
   }
}
```

6. Tambahkan kode berikut pada kelas BukuMain

```
…
   Buku buku = new Buku();
   buku.setJudul("Pemrograman Java");
   buku.setPengarang("Budi");
   System.out.println("Judul Buku: "+buku.getJudul());
   System.out.println("Pengarang: "+buku.getPengarang());
…
```

7. Save, compile dan jalankan program

```
D:\Program\Java\java\modul\modul 4>javac BukuMain.java

D:\Program\Java\java\modul\modul 4>java BukuMain
Judul Buku: Pemrograman Java
Pengarang: Budi
```

4.2 Interface

4.2.1 Interface dan Implements

1. Buat interface baru dengan nama Karyawan seperti kode berikut:

```
public interface Karyawan{

}
```

2. Tambahkan method bertipe String dengan nama getJabatan seperti kode berikut

```
…
void  setJabatan(String jabatan);
…
```

3. Buat kelas baru dengan nama KaryawanMagang yang meng- implement interface Karyawan seperti kode berikut:

```
public class KaryawanMagang implements Karyawan {

}
```

4. Tambahkan variable jabatan dengan tipe String seperti kode berikut:

```
…
   private String jabatan;
…
```

5. Tambahkan method yang meng-*override* method setJabatan pada interface Karyawan

```
…
   public void setJabatan(String jabatan){
        this.jabatan = jabatan;
   }
…
```

6. Tambahkan method getter untuk mengembalikan nilai variable jabatan pada kelas KaryawanMagang seperti kode berikut:

```
…
   public String getJabatan(){
      return jabatan;
   }
…
```

7. Buat kelas baru dengan nama Main seperti kode berikut

```
public class Main {
   public static void main(String[] args) {
   }
}
```

8. Tambahkan Kode berikut

```
…
   KaryawanMagang karyawan1 = new KaryawanMagang();
   karyawan1.setJabatan("Akuntan");
   System.out.println("Jabatan:"+karyawan1.getJabatan());
…
```

9. Save, compile, dan jalankan program

```
D:\Program\Java\java\modul\modul 4\Coba>java Main
Jabatan:Akuntan
```

4.2.2 Interface extends interface

1. Buat interface baru dengan nama Penduduk seperti kode berikut

```
public interface Penduduk {

}
```

2. Tambahkan variabel warganegara dan method setNama dan setNik seperti kode berikut

```
…
   String warganegara = "Indonesia";
   void setNama(String nama);
   void setNik(String nik);
…
```

3. Tambahkan keyword extends Penduduk pada interface Karyawan seperti kode berikut

```
public interface Karyawan extends Penduduk{
    void  setJabatan(String jabatan);
    }
```

4. Tambahkan variable nama dan nik pada kelas KaryawanMagang seperti kode berikut:

```
…
   private String nik;
   private String nama;
…
```

5. Tambahkan method yang meng-*overrinde* method setNama dan setNik dari interface Penduduk pada kelas KaryawanMagang seperti kode berikut

```
…
   public void setNama(String nama){
        this.nama = nama;
   }
   public void setNik(String nik){
        this.nik = nik;
   }
…
```

6. Tambahkan method getter untuk mengembalikan nilai dari variable nama dan nik pada kelas KaryawanMagang seperti kode berikut

```
…
   public String getNama(){
        return nama;
   }
   public String getNik(){
        return nik;
   }
…
```

7. Tambahkan kode berikut pada kelas Main

```
…
   karyawan1.setNama("Budi");
   karyawan1.setNik("7370002909940001");
   System.out.println("Nama:"+karyawan1.getNama());
   System.out.println("NIK:"+karyawan1.getNik());
   System.out.println("warganegara:"+karyawan1.warganegara);
…
```

8. Save compile dan jalankan program

```
D:\Program\Java\java\modul\modul 4\Coba>java Main
Jabatan:Akuntan
Nama:Budi
NIK:7370002909940001
Warganegara:Indonesia
```

4.2.3 Implements lebih dari satu interface

1. Buat interface baru dengan nama Mahasiswa seperti kode berikut

```
public interface Mahasiswa {

}
```

2. Tambahkan method setUniversitas dan setJurusan pada interface Mahasiswa seperti kode berikut

```
…
   void setUniversitas(String universitas);
…
```

3. Ubah header kelas KaryawanMagang menjadi seperti berikut untuk mengimplement interface Mahawsiwa

```
public class KaryawanMagang implements Karyawan, Mahasiswa {

}
```

4. Tambahkan variable universitas pada kelas KaryawanMagang seperti kode berikut

```
…
   private String universitas;
…
```

5. Tambahkan method yang meng-*overrinde* method setUniversitas dari interface Mahasiswa pada kelas KaryawanMagang seperti kode berikut

```
…
   public void setUniversitas(String universitas){
        this.universitas=universitas;
   }
…
```

6. Tambahkan method getter untuk mengembalikan nilai dari variable universitas pada kelas KarwayanMagang seperti kode berikut

```
…
    public String getUniversitas(){
         return universitas;
    }
…
```

7. Tambahkan kode berikut pada kelas Main

```
…
   karyawan1.setUniversitas("Universitas Hasanuddin");
   System.out.println("Asal Universitas:
   :"+karyawan1.getUniversitas());
…
```

8. Save, compile, dan jalankan program

```
D:\Program\Java\java\modul\modul 4\Coba>java Main
Jabatan:Akuntan
Nama:Budi
NIK:7370002909940001
Warganegara:Indonesia
Asal Universitas: :Universitas Hasanuddin
```

4.3 Abstract Class

1. Buat kelas abstrak baru dengan nama PNS seperti kode berikut

```
public  abstract class PNS{

}
```

2. Tambahkan kode berikut pada kelas PNS

```
…
   private String nip;
   PNS(String nip)
   {
       this.nip = nip;
   }
   public String getNip()
   {
       return nip;
   }
   public abstract void setKeahlian(String keahlian);
…
```

3. Buat kelas baru dengan nama AbstrakMain seperti kode berikut:

```
public class AbstrakMain {
   public static void main(String[] args) {
   }
}
```

4. Tambahkan kode berikut pada kelas AbstrakMain

```
…
   PNS pns = new PNS("374928");
…
```

5. Save, compile dan jalankan program

```
D:\Program\Java\java\modul\modul 4>javac AbstrakMain.java
AbstrakMain.java:32: error: PNS is abstract; cannot be instantiated
               PNS pns = new PNS("879879");
                         ^
1 error
```

Program akan error karena kelas abstrak tidak bisa dijadikan objek

6. Buat kelas baru dengan nama Guru seperti kode berikut

```
public class Guru extends PNS {
}
```

7. Tambahkan kode berikut pada kelas Guru

```
…
   private String keahlian, nama, nik;
   Guru(String nip){
       super(nip);
   }
   public void setKeahlian(String keahlian){
       this.keahlian = keahlian;
   }
   public String getKeahlian(){
       return keahlian;
   }
…
```

8. Ubah kode pada kelas AbtrakMain menjadi seperti berikut

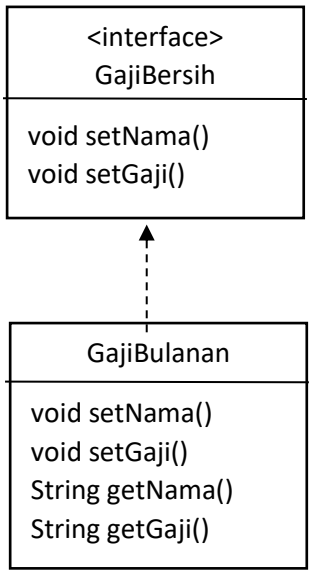
```
public class AbstrakMain {
   public static void main(String[] args) {
      Guru guru = new Guru("374928");
      guru.setKeahlian("Matematika");
      System.out.println("NIP: "+guru.getNip());
      System.out.println("Keahlian: "+guru.getKeahlian());
   }
}
```

9. Save, compile dan jalankan program

```
D:\Program\Java\java\modul\modul 4>java AbstrakMain
NIP: 93848717957
Keahlian: Matematika
```

5. Tugas Evaluasi

Buatlah kelas GajiBulanan untuk menghitung gaji yang diterima tiap bulan setelah dipotong pajak. Kelas GajiBulanan meng-*implement* interface GajiBersih seperti gambar berikut



Buat kelas baru dengan nama main untuk memasukkan input dan menampilkan output Input dari program adalah nama dan gaji, outputnya adalah nama dan gaji setelah dipotong pajak

6. Tes 1

1. Tuliskan pengertian enkapsulasi
   Jawab:

2. Apa perbedaan dari method setter dan getter
   Jawab:

3. Sebutkan salah satu perbedaan abstract class dan interface
   Jawab:

4. Tuliskan syntax penulisan abstract class
   Jawab:

5. Apa perbedaan abstract class dan kelas tanpa keyword abstract
   Jawab:

7. Test2

1. Apa output dari program berikut

```
public interface Herbifora {
    String makanan = "tumbuhan";
    void makan(){
        System.out.println("Makan tumbuhan");
    }
}
public class Rusa {
    public void makan(){
        System.out.println("Makan rumput");
    }
}
public class PostTest {
    public static void main(String[] args) {
        Rusa rusa = new Rusa();
        rusa.makan();
    }
}
```

   a. Makan tumbuhan
   b. Makan rumput
   c. Error karena variable pada interface memiliki nilai
   d. Error karena method pada interface memiliki body

2. Program berikut error karena

```
public interface Hewan {
    void bernapas();
}
public interface Herbifora implements Hewan{
    void makan();
}
public class Rusa {
    public void makan(){System.out.println("Makan rumput");}
    public void bernapas();
}
```

   a. Kelas rusa tidak meng-*override* semua method pada interface Herbifora
   b. Kelas rusa tidak meng-*override* semua method pada interface Hewan
   c. Method bernapas pada kelas Rusa tidak memiliki body

d. Interface Herbifora meng-implements interface Hewan

3. Output dari program berikut adalah

```
interface Kurang{
    int a = 10;
    int setKurang(int a);
}
class Decrement implements Kurang{
    public int setKurang(int a){return --a;}
}
class PostTest{
    public static void main(String[] args) {
        Decrement d = new Decrement();
        d.setKurang(d.a);
        System.out.println(d.a);
    }
}
```

a. 9
b. 10
c. Error karena variable a tidak di deklarasikan pada kelas Decrement
d. Error karena variable a tidak di deklarasikan pada kelas PostTest

4. Apa output dari program berikut:

```
abstract class Tumbuhan{
    void berbunga(){System.out.println("Tumbuhan berbunga");}
    abstract void fotosintesis();
}
class Lumut extends Tumbuhan{
    public void fotosintesis(){
        System.out.println("Berfotosintesis");}
}
class PostTest{
    public static void main(String[] args) {
        Lumut lumut = new Lumut();
        lumut.fotosintesis();
    }
}
```

a. Berfotosintesis
b. Error karena method berbunga pada kelas abstrak memiliki body
c. Error karena method berbunga tidak di override pada kelas Lumut
d. Error karena method pada kelas abstrak tidak bersifat public

5. Output dari program berikut adalah:

```
interface Bunga{
    void warna();
}
class Melati implements Bunga{
    void warna(){System.out.println("Warna putih");}
}
class PostTest {
    public static void main(String[] args) {
        Melati melati = new Melati();
        melati.warna();
    }
}
```

a. Warna putih
b. Program error karena method warna bukan public
c. Program error karena interface Bunga bukan public
d. Program error karena kelas Melati bukan public