

# Jobsheet 11 –Interface dalam OOP Java

## A. Kompetensi

Setelah menyelesaikan lembar kerja ini mahasiswa diharapkan mampu:

1. Menjelaskan maksud dan tujuan penggunaan Interface;
2. Menerapkan Interface di dalam pembuatan program.

## B. Pendahuluan

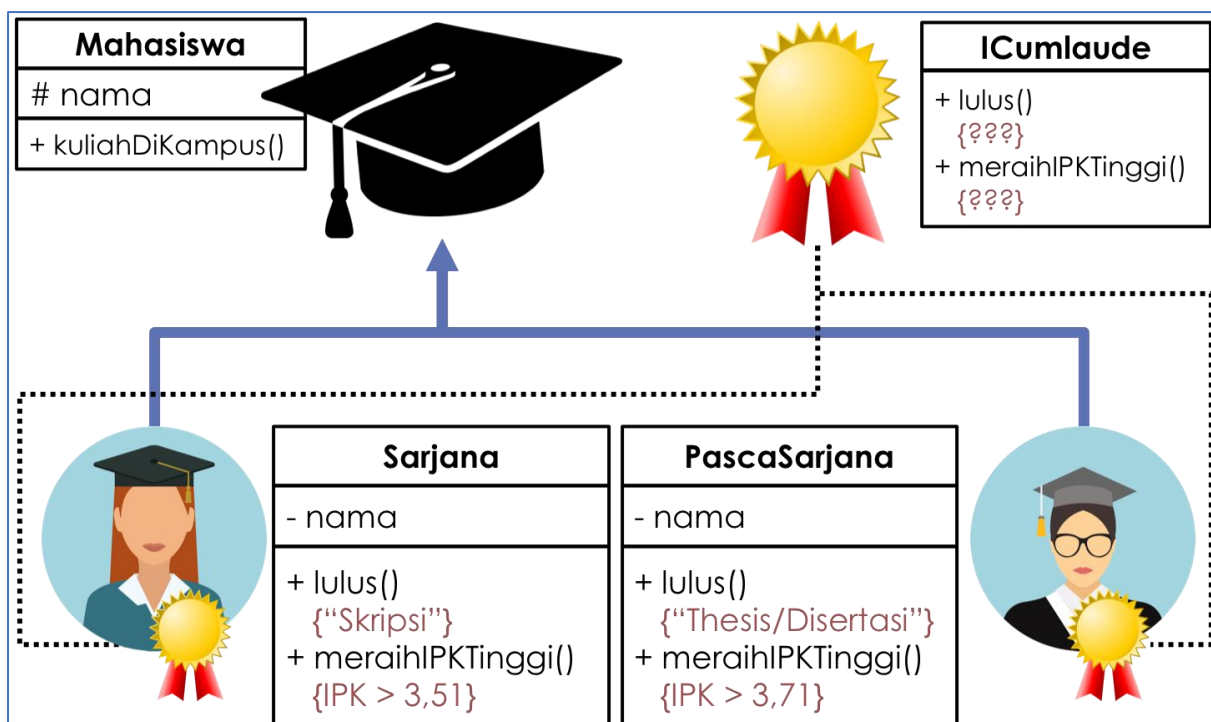
*Interface* adalah struktur data yang hanya berisi *abstract methods*. Tidak ada apa-apa selain *method abstract* pada *interface*, termasuk atribut *getter* dan *setter*.

### 1. Karakteristik:

- a. Tidak ada apa-apa di dalamnya selain *abstract methods*.
- b. Di konvensi bahasa pemrograman Java, namanya dianjurkan untuk selalu diawali dengan huruf kapital 'I'.
- c. Selalu dideklarasikan dengan menggunakan kata kunci `interface`.
- d. Diimplementasikan dengan menggunakan kata kunci `implements`.

### 2. Kegunaan:

Bertindak seperti semacam kontrak/syarat yang HARUS dipenuhi bagi suatu class agar class tersebut dapat dianggap sebagai 'sesuatu yang lain'.



### 3. Manfaat Interface

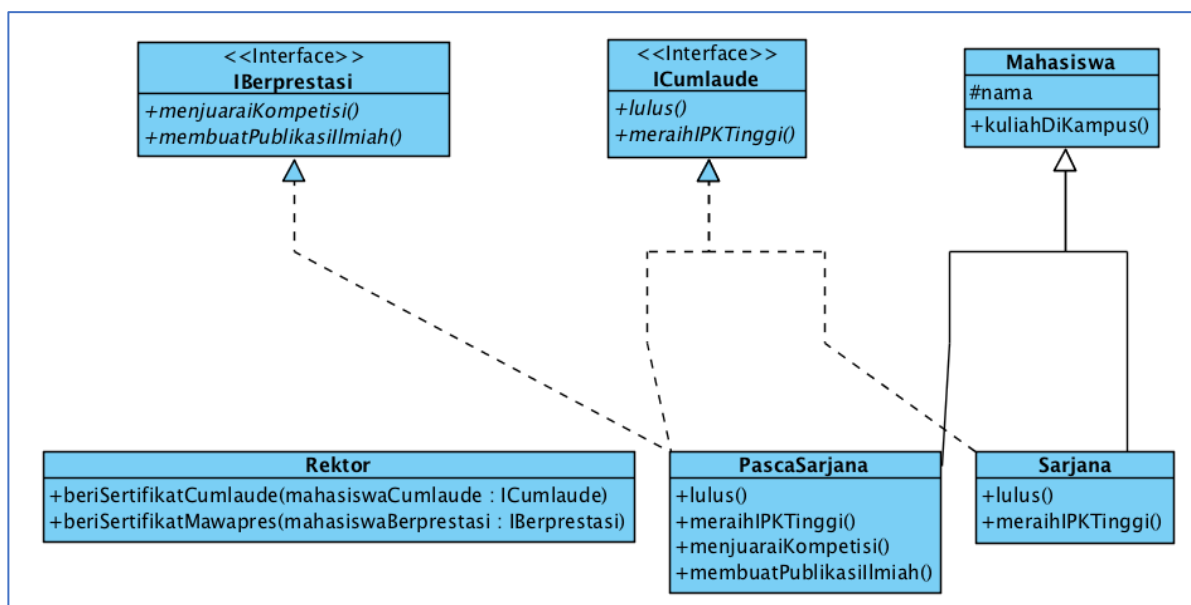
- Interface memungkinkan sebuah subclass diturunkan dari beberapa superclass (**multiple inheritance**)
- Interface lebih mempermudah sistem analis dalam membuat konsep aplikasi
- Pemrograman java menggunakan interface supaya dapat lebih efisien karena adanya multiple inheritance.

### 4. Notasi Class Diagram Interface

- Diawali dengan notasi <<interface>>
- Nama TIDAK dicetak miring
- Nama method dicetak miring
- Implements dilambangkan dengan garis panah putus-putus

### 5. Multiple interfaces implementation

Multiple interface merupakan suatu class dapat mengimplementasikan banyak (lebih dari satu) interface. Pada contoh berikut class PascaSarjana mengimplementasikan interface ICumlaude & IBerprestasi.



### 6. Aturan Penulisan Interface

- Interface adalah sebuah tipe referensi pada java.
- Secara struktur hampir sama dengan class. Ada beberapa aturan dalam penulisan interface:
  - a. Pada class yang mengimplement, modifier method – method hanya boleh public
  - b. Jumlah parameter method interface harus sama dengan class yang meng-implement-nya
  - c. Tidak boleh ada method concrete di interface, apabila ada maka program akan error

```
// Contoh method concrete :
public void output(){
    System.out.println();
}
```

- d. Tidak boleh ada constructor
- e. Interface tidak bisa di instanisasi, tapi bisa di instanisasi melalui class yang meng-*implement*-nya

## 7. Perbedaan Abstract Class dan Interface

Abstract Class	Interface
Dapat berisi abstract dan non-abstract method	Hanya dapat berisi abstract method
Modifiersnya harus dituliskan sendiri	Tidak perlu menulis <code>public abstract</code> di depan nama method. karena secara implisit, modifier untuk method di interface adalah <code>public</code> dan <code>abstract</code>
Dapat deklarasai <i>constant</i> dan instance variable	Hanya dapat mendeklarasikan <i>constant</i> . Secara implisit variable yang dideklarasikan di interface bersifat <code>public</code> , <code>static</code> dan <code>final</code>
Method boleh bersifat <code>static</code> dan <code>final</code>	Method tidak boleh bersifat <code>static</code> dan <code>final</code>
Abstract Class hanya dapat meng- <i>extend</i> satu abstract class lainnya	Suatu interface dapat meng- <i>extend</i> satu atau lebih interface lainnya
Abstract Class hanya dapat meng- <i>implement</i> beberapa interface	Dan tidak dapat meng- <i>implement</i> class atau interface lainnya

## 8. Perbedaan Interface dengan Inheritance

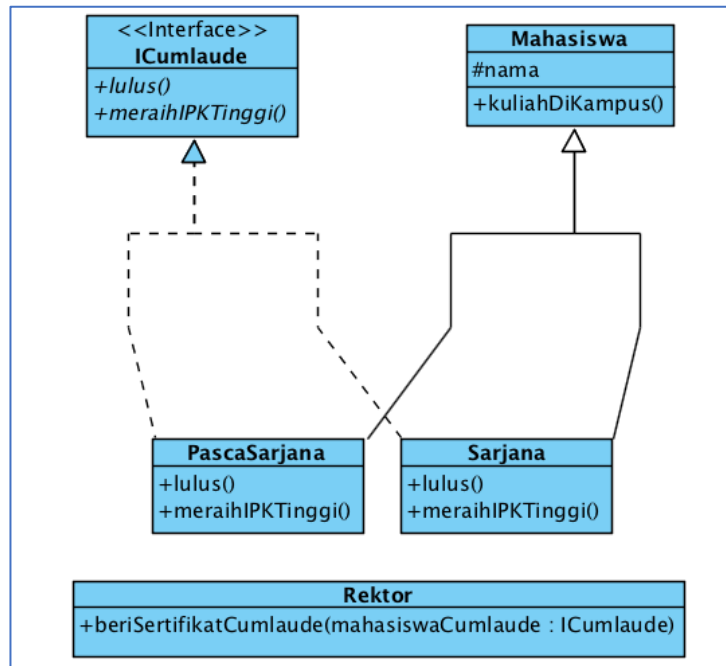
- a. **Inheritance (Pewarisan)** digunakan ketika parent class memiliki atribut dan method lalu semuanya **diwariskan ke subclass**.
- b. Sedangkan Interface digunakan ketika parent tidak memiliki apa-apa, hanya method yang tidak memiliki tubuh dan harus **diimplementasikan pada subclassnya**.
- c. Interface juga memungkinkan terjadinya **multiple inheritance**, yaitu sebuah subclass yang diturunkan dari lebih dari satu superclass.

## 9. Hal-Hal yang tidak boleh dilakukan di Interface (Penyebab terjadinya error)

- a. Jangan membuat variabel di dalam interface, tetapi diperbolehkan untuk membuat konstanta (`final`)
- b. Jangan mengisi method yang telah diinisiasi, cukup tuliskan nama method, tipe data dan parameter saja. Tapi untuk default method diperbolehkan untuk memberikan body. Contoh:
- c. Jangan berikan modifier **private** ataupun **protected** pada method dan konstanta yang ada di dalam interface.
- d. Interface **tidak dapat dibuat objek instance-nya** dengan kata kunci **new**.

### C. PRAKTIKUM - INTERFACE

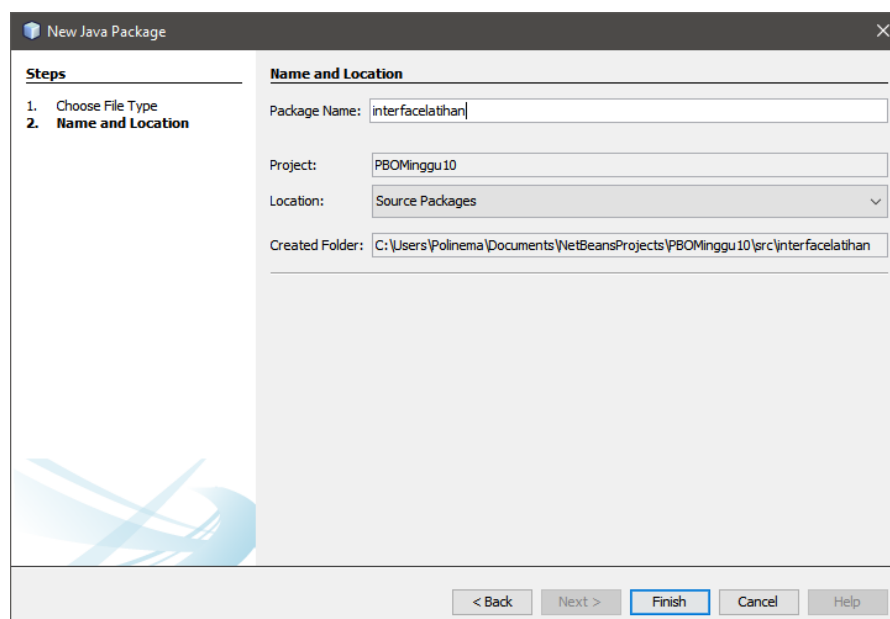
Pada sebuah wisuda, seorang Rektor akan memberikan penghargaan sertifikat *Cumlaude* pada semua mahasiswa yang memenuhi persyaratan. Persyaratan agar seorang mahasiswa dapat disebut sebagai *Cumlaude* berbeda-beda antara mahasiswa Sarjana dan Pasca Sarjana.



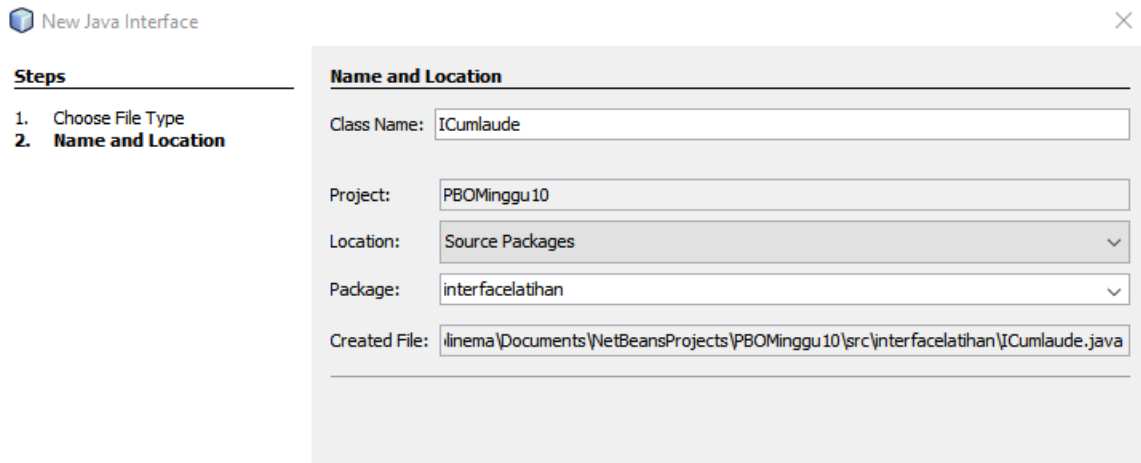
Untuk menjadi *cumlaude*, mahasiswa Sarjana harus mengerjakan **skripsi** dan memiliki IPK lebih tinggi dari 3,51. Sedangkan untuk mahasiswa Pasca Sarjana, mereka harus mengerjakan **tesis** dan meraih IPK lebih tinggi dari 3,71.

Pada percobaan ini kita akan mencoba menerjemahkan skenario di atas ke dalam sebuah aplikasi sederhana yang memanfaatkan interface.

1. Pada project latihan **PBOMinggu10**, buat sebuah package baru bernama **interfacelatihan**. Kemudian klik **Finish**



2. Pada package yang baru dibuat tersebut, tambahkan sebuah **interface** baru dengan cara klik kanan pada package → **New** → **Java Interface...** Beri nama interface baru tersebut dengan nama **ICumlaude**.



3. Pada interface ICumlaude tersebut, tambahkan 2 *abstract methods* bernama **lulus()** dan **meraihIPKTINGGI()**.

```
public interface ICumlaude
{
    public abstract void lulus();
    public abstract void meraihIPKTINGGI();
}
```

4. Berikutnya, buatlah sebuah class baru bernama **Mahasiswa** dengan baris-baris kode seperti dibawah ini.

```
public class Mahasiswa
{
    protected String nama;

    public Mahasiswa(String nama)
    {
        this.nama = nama;
    }

    public void kuliahDiKampus()
    {
        System.out.println("Aku mahasiswa, namaku " + this.nama);
        System.out.println("Aku berkuliah di kampus.");
    }
}
```

5. Selanjutnya, buatlah class baru bernama **Sarjana** yang merupakan **turunan** dari class Mahasiswa. Class Sarjana tersebut dibuat meng-**implements** interface ICumlaude yang sudah dibuat sebelumnya tadi. Ketikkan kode di bawah pada class tersebut. **Tips:** Anda dapat menggunakan fasilitas *override* otomatis dengan cara yang sama yaitu dengan mengklik ikon lampu peringatan seperti pada percobaan 1.

```
public class Sarjana extends Mahasiswa implements ICumlaude
{
    public Sarjana(String nama)
    {
        super(nama);
    }

    @Override
    public void lulus() {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    @Override
    public void meraihIPKTinggi() {
        throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

6. Selanjutnya sesuaikan isi dari *method* **lulus()** dan **meraihIPKTinggi()** agar sama dengan baris kode di bawah.

```
public class Sarjana extends Mahasiswa implements ICumlaude
{
    public Sarjana(String nama)
    {
        super(nama);
    }

    @Override
    public void lulus()
    {
        System.out.println("Aku sudah menyelesaikan SKRIPSI");
    }

    @Override
    public void meraihIPKTinggi()
    {
        System.out.println("IPK-ku lebih dari 3,51");
    }
}
```

Perhatikan pada baris kode di atas, class Sarjana meng-**extend** class Mahasiswa, ini berarti, **Sarjana adalah Mahasiswa** sementara itu agar semua objek dari class Sarjana ini nantinya dapat disebut sebagai **Cumlaude** maka ia harus meng-**implements** interface **ICumlaude**.

7. Kemudian dengan **cara yang sama** buatlah class baru bernama **PascaSarjana** dengan baris kode seperti di bawah ini.

```
public class PascaSarjana extends Mahasiswa implements ICumlaude
{
    public PascaSarjana(String nama)
    {
        super(nama);
    }

    @Override
    public void lulus()
    {
        System.out.println("Aku sudah menyelesaikan TESIS");
    }

    @Override
    public void meraihIPKTinggi()
    {
        System.out.println("IPK-ku lebih dari 3,71");
    }
}
```

8. Lalu buatlah sebuah class baru bernama **Rektor**. Class ini adalah class yang memanfaatkan class-class Mahasiswa yang telah dibuat sebelumnya.

```
public class Rektor
{
    public void beriSertifikatCumlaude(ICumlaude mahasiswa)
    {
        System.out.println("Saya REKTOR, memberikan sertifikat cumlaude.");
        System.out.println("Selamat! silahkan perkenalkan diri Anda..");

        mahasiswa.lulus();
        mahasiswa.meraihIPKTinggi();

        System.out.println("-----");
    }
}
```

9. Terakhir, buatlah sebuah class baru bernama **interfacemain** yang diletakkan pada **package yang sama** dengan class-class percobaan 2. Tambahkan baris kode berikut ini:

```

1  import interfamelatihan.ICumlaude;
4  import interfamelatihan.Mahasiswa;
5  import interfamelatihan.PascaSarjana;
6  import interfamelatihan.Rektor;
7  import interfamelatihan.Sarjana;
8
9  /**
10   *
11   * @author Polinema
12   */
13  public class interfamelatihan {
14
15      public static void main(String[] args) {
16          Rektor pakrektor = new Rektor();
17
18          Mahasiswa mhsBiasa = new Mahasiswa("Charlie");
19          Sarjana sarjanaCumlaude = new Sarjana("Dini");
20          PascaSarjana masterCumlaude = new PascaSarjana("Elok");
21
22          pakrektor.beriSertifikatCumlaude(mhsBiasa);
23          pakrektor.beriSertifikatCumlaude(sarjanaCumlaude);
24          pakrektor.beriSertifikatCumlaude(masterCumlaude);
25      }
26
27  }

```

10. Pada baris kode tersebut, apabila Anda mengetikkan semua class dengan benar, maka akan terdapat *error* dan class Program tidak dapat dieksekusi. Perbaikilah kode Anda agar program yang Anda buat mengeluarkan output seperti berikut ini:

```

: Output - PBOMinggu10 (run)

run:
Saya REKTOR,memberikan sertifikat cumlaude.
Selamat! silahkan perkenalkan diri Anda ...
Aku sudah menyelesaikan SKRIPSI
IPK-ku lebih dari 3.51
-----
Saya REKTOR,memberikan sertifikat cumlaude.
Selamat! silahkan perkenalkan diri Anda ...
Aku sudah menyelesaikan TESIS
IPK-ku lebih dari 3.71
-----
BUILD SUCCESSFUL (total time: 0 seconds)

```

#### D. PERTANYAAN PERCOBAAN 1

- 1) Pada langkah ke 9, pada baris program ke 3 terdapat warning pada script tersebut. Jelaskan penyebab terjadinya hal tersebut ?

```

1  import interfamelatihan.ICumlaude;
4  import interfamelatihan.Mahasiswa;
5  import interfamelatihan.PascaSarjana;
6  import interfamelatihan.Rektor;
7  import interfamelatihan.Sarjana;

```

- 2) Pada langkah ke 9, pada baris program ke 3. Apa yang terjadi jika script tersebut dihilangkan? Jelaskan menurut pemahaman anda.



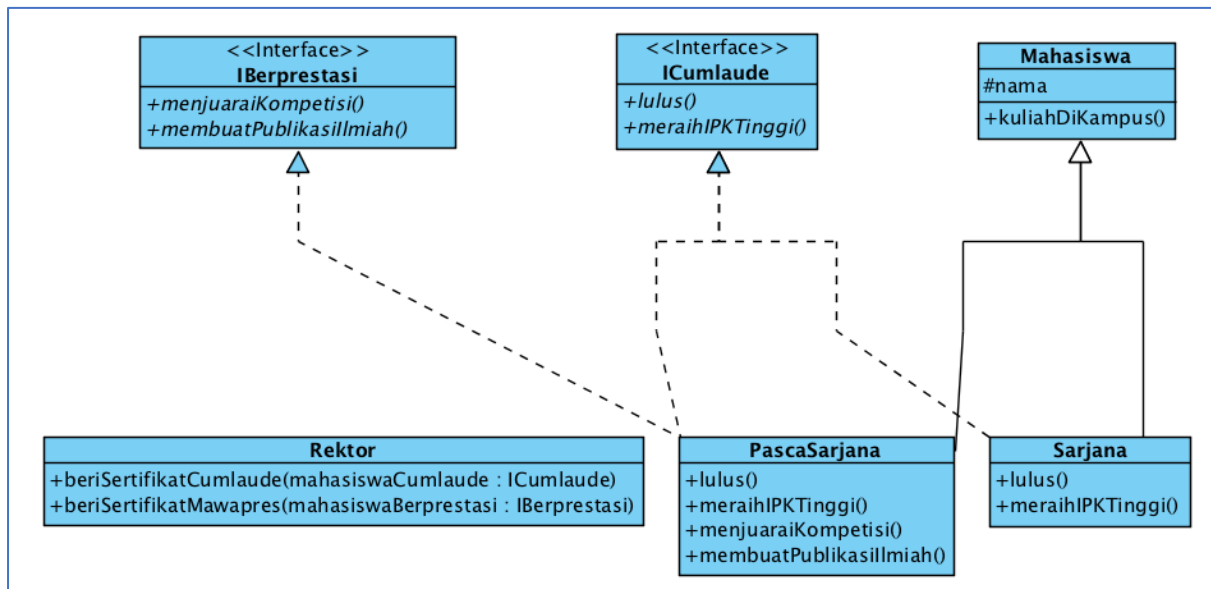
- 3) Mengapa pada langkah nomor 9 terjadi error? Jelaskan!
- 4) Dapatkah method **kuliahDiKampus()** dipanggil dari objek **sarjanaCumlaude** di class **Program**? Mengapa demikian?
- 5) Dapatkah method **kuliahDiKampus()** dipanggil dari parameter **mahasiswa** di method **beriSertifikatCumlaude()** pada class **Rektor**? Mengapa demikian?
- 6) Modifikasilah method **beriSertifikatCumlaude()** pada class **Rektor** agar hasil eksekusi class **Program** menjadi seperti berikut ini:

```
Output - PBOMinggu10 (run)

run:
Saya REKTOR,memberikan sertifikat cumlaude.
Selamat! silahkan perkenalkan diri Anda ...
Aku mahasiswa, namaku Dini
Aku berkuliah di kampus.
Aku sudah menyelesaikan SKRIPSI
IPK-ku lebih dari 3.51
-----
Saya REKTOR,memberikan sertifikat cumlaude.
Selamat! silahkan perkenalkan diri Anda ...
Aku mahasiswa, namaku Elok
Aku berkuliah di kampus.
Aku sudah menyelesaikan TESIS
IPK-ku lebih dari 3.71
-----
BUILD SUCCESSFUL (total time: 2 seconds)
```

### E. Percobaan 2: Multiple Interfaces Implementation

Pada percobaan kali ini kita akan memodifikasi program yang telah dibuat pada Percobaan 2 sehingga pada program tersebut nantinya akan terdapat sebuah class yang meng-*implements* lebih dari 1 *interface*.



Bayangkan pada skenario sebelumnya, dimana seorang rektor juga akan **beriSertifikatMawapres()** pada sebuah acara wisuda. Mahasiswa yang berhak menerima penghargaan tersebut tentunya adalah mahasiswa yang berprestasi, dimana kriteria prestasi di sini berbeda antara mahasiswa Sarjana dengan mahasiswa Pasca Sarjana. Pada percobaan ini, kita akan menentukan kriteria prestasi yaitu: harus **menjuaraiKompetisi()** dan **membuatPublikasiIlmiah()**.

1. Pada package yang sama dengan package pada Percobaan Sebelumnya, tambahkan sebuah *interface* baru yang bernama **IBERprestasi**. Tambahkan baris kode seperti berikut didalamnya.

```

public interface IBERprestasi
{
    public abstract void menjuaraiKompetisi();
    public abstract void membuatPublikasiIlmiah();
}

```

2. Selanjutnya, **modifikasilah** class **PascaSarjana** dengan menambahkan interface baru **IBERprestasi** dibelakang kata kunci **implements**. Lalu dengan cara yang sama seperti sebelumnya, kliklah ikon lampu peringatan untuk **meng-generate** semua method *abstract* dari interface **IBERprestasi** pada class **PascaSarjana**.

```

13 public class PascaSarjana extends Mahasiswa implements ICumlaude, IBERprestasi
{

```

3. Modifikasilah *method* yang telah di-generate oleh NetBeans menjadi seperti berikut.

```

@Override
public void menjuaraiKompetisi() {
    System.out.println("Saya telah menjuarai kompetisi INTERNASIONAL");
}

@Override
public void membuatPublikasiIlmiah() {
    System.out.println("Saya menerbitkan artikel di jurnal INTERNASIONAL");
}

```

4. Tambahkan *method* **beriSertifikatMawapres()** dengan baris kode seperti di bawah, pada class **Rektor**.

```

public void beriSertifikatMawapres(IBerprestasi mahasiswa)
{
    System.out.println("Saya REKTOR, memberikan sertifikat MAWAPRES.");
    System.out.println("Selamat! Bagaimana Anda bisa berprestasi?");

    mahasiswa.menjuaraiKompetisi();
    mahasiswa.membuatPublikasiIlmiah();

    System.out.println("-----");
}

```

5. Terakhir, modifikasilah *method* **main()** pada class **MultipleInterfaceMain** Anda. *Comment*-lah semua baris yang terdapat *method* **beriSertifikatCumlaude()**, lalu tambahkan baris kode baru seperti pada gambar di bawah ini.

```

16 public class MultipleInterfaceMul {
17
18     /**
19      * @param args the command line arguments
20      */
21     public static void main(String[] args) {
22         Rektor pakRektor = new Rektor();
23
24         Sarjana sarjanaCum = new Sarjana("Dini");
25         PascaSarjana masterCum = new PascaSarjana("Elok");
26
27         pakRektor.beriSertifikatMawapres(sarjanaCum);
28         pakRektor.beriSertifikatMawapres(masterCum);
29     }
30 }

```

6. Akan terdapat *error* pada langkah-5, sehingga program tidak dapat dieksekusi. Perbaikilah kode programnya, sehingga hasil eksekusi menjad **sama** seperti pada *screenshot* di bawah ini.

```

: Output - PBOMinggu10 (run)

run:
Saya REKTOR, memberikan sertifikat MAWAPRES.
Selamat! Bagaimana Anda bisa berprestasi?
Saya telah menjuarai kompetisi INTERNASIONAL
Saya menerbitkan artikel di jurnal INTERNASIONAL
-----
BUILD SUCCESSFUL (total time: 0 seconds)

```

## F. PERTANYAAN PERCOBAAN 1

1. Pada script code interface **IBerprestasi**, modifikasi script tersebut sesuai dengan gambar dibawah ini :

Diganti : protected	←	<code>public</code>	<code>abstract void menjuaraiKompetisi();</code>
Diganti : private	←	<code>public</code>	<code>abstract void membuatPublikasiIlmiah();</code>

```

public interface IBerprestasi {
    abstract void menjuaraiKompetisi();
    abstract void membuatPublikasiIlmiah();
}

```

Dari perubahan script diatas, apa yang terjadi ? serta jelaskan alasannya (capture hasilnya)

2. Perhatikan script code dibawah ini :

```

22 public static void main(String[] args) {
    IBerprestasi prestasi = new IBerprestasi();
}

```

Jelaskan menurut anda, mengapa hasil dari script code tersebut error ?

3. Apabila **Sarjana Berprestasi** harus **menjuarai kompetisi NASIONAL** dan **menerbitkan artikel di jurnal NASIONAL**, maka modifikasilah class-class yang terkait pada aplikasi Anda agar di class **Program** objek **pakRektor** dapat memberikan sertifikat mawapres pada objek **sarjanaCumlaude**.

```

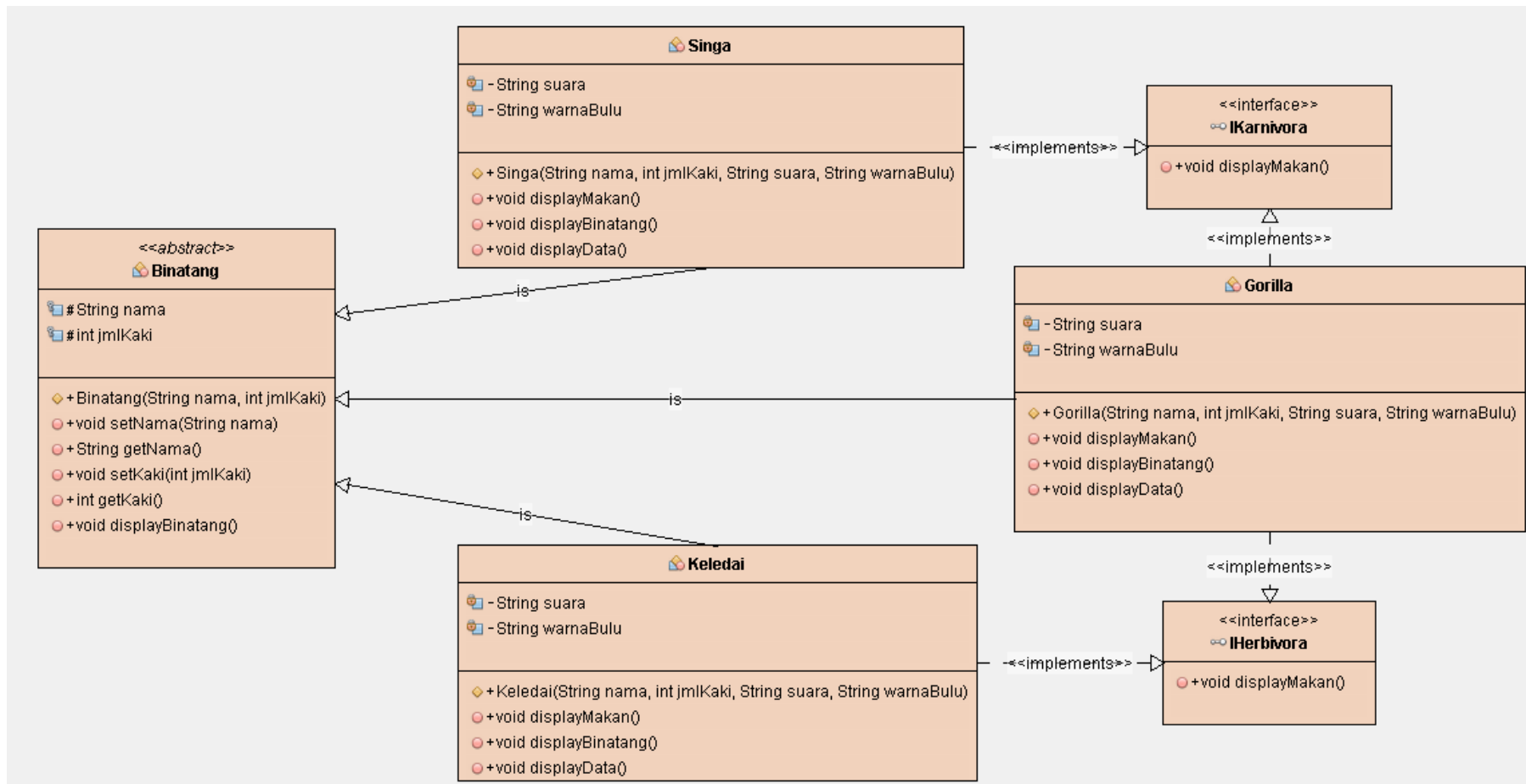
: Output - PBOMinggu10 (run)

run:
Saya REKTOR, memberikan sertifikat MAWAPRES.
Selamat! Bagaimana Anda bisa berprestasi?
Saya telah menjuarai kompetisi NASIONAL
Saya menerbitkan artikel di jurnal NASIONAL
-----
Saya REKTOR, memberikan sertifikat MAWAPRES.
Selamat! Bagaimana Anda bisa berprestasi?
Saya telah menjuarai kompetisi INTERNASIONAL
Saya menerbitkan artikel di jurnal INTERNASIONAL
-----
BUILD SUCCESSFUL (total time: 1 second)

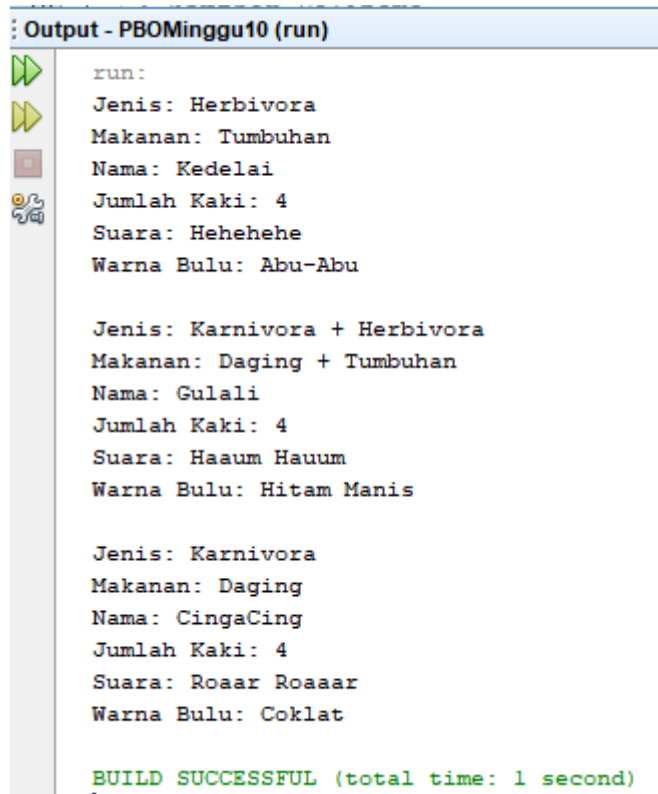
```

## G. TUGAS

Terdapat sebuah UML diagram dari sistem yang mengidentifikasi tentang karakteristik dan jenis makanan dari beberapa binatang, seperti gambar dibawah ini. **Singa** hanya makan jenis daging yang tergolong sebagai jenis **Karnivora**, sedangkan **Keledai** termasuk binatang **Herbivora** karena hanya mengkonsumsi jenis tumbuhan. Sedangkan untuk Gorilla dapat mengkonsumsi makanan dari kedua jenis tersebut, dapat disebut dengan **omnivora** dan dapat dikategorikan sebagai **kombinasi dari Herbivora dan Karnivora**. Dalam UML tersebut, juga terdapat beberapa karakteristik dari binatang seperti **nama, suara, warnabulu dan jumlah kaki**.



Dengan penjelasan kasus dan UML diatas, implementasikan sebuah sistem pada Java dengan output seperti gambar dibawah ini. **Hasil tugas dapat dibentuk laporan dengan capture script code beserta penjelasannya. (Sertakan juga hasil UML dari sistem anda).**



```
run:
Jenis: Herbivora
Makanan: Tumbuhan
Nama: Kedelai
Jumlah Kaki: 4
Suara: Hehehehe
Warna Bulu: Abu-Abu

Jenis: Karnivora + Herbivora
Makanan: Daging + Tumbuhan
Nama: Gulali
Jumlah Kaki: 4
Suara: Haaum Hauum
Warna Bulu: Hitam Manis

Jenis: Karnivora
Makanan: Daging
Nama: CingaCing
Jumlah Kaki: 4
Suara: Roaar Roaaar
Warna Bulu: Coklat

BUILD SUCCESSFUL (total time: 1 second)
```

=====Selamat Mengerjakan=====