**Jobsheet 09 – Abstract Class dan Interface**

# Kompetensi

Setelah menyelesaikan lembar kerja ini mahasiswa diharapkan mampu:

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

# Pendahuluan Abstract Class

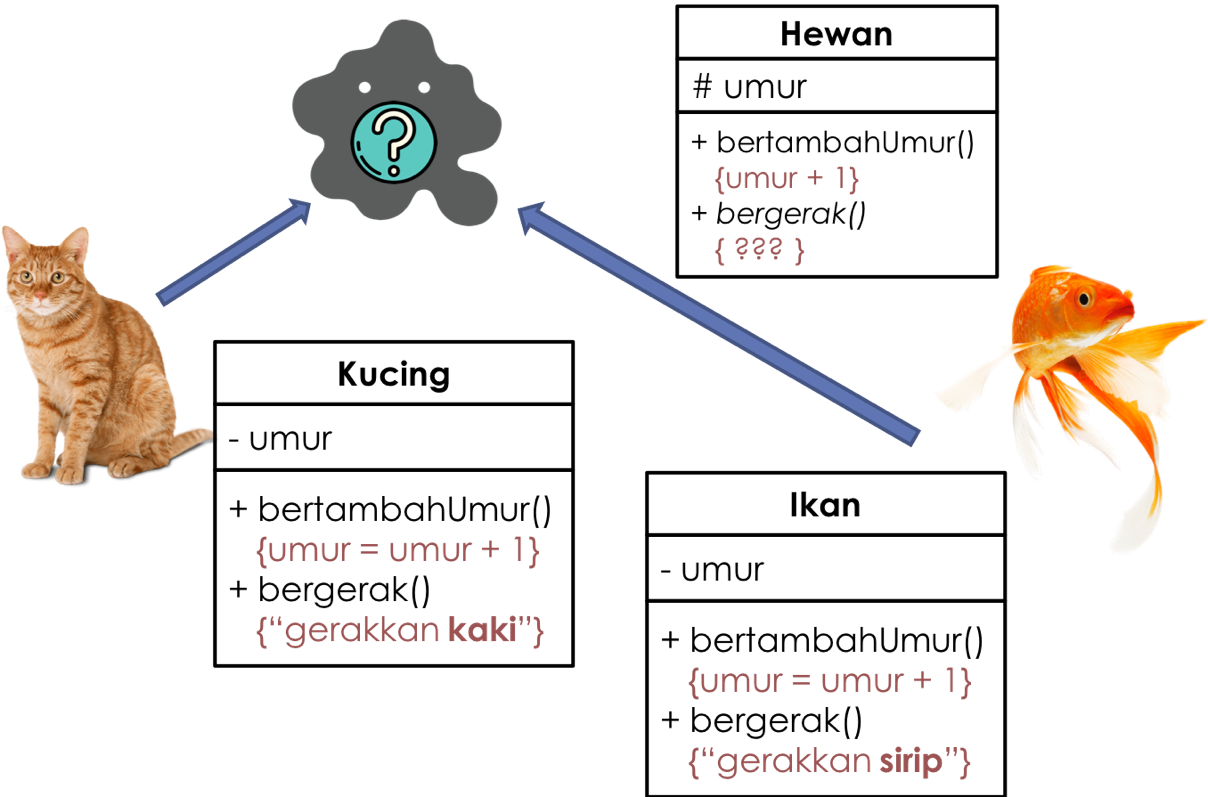
*Abstract Class* adalah class yang tidak dapat diinstansiasi namun dapat di-*extend*. *Abstract class baru* dapat dimanfaatkan ketika ia di-*extend*.

**Karakteristik:**

1. Dapat memiliki *properties* dan *methods* seperti class biasa.
2. Selalu memiliki *methods* yang tidak memiliki tubuh (hanya deklarasinya saja), disebut juga *abstract method*.
3. Selalu dideklarasikan dengan menggunakan kata kunci abstract class.

**Kegunaan:**

Menggambarkan sesuatu yang bersifat umum, yang hanya bisa berfungsi setelah ia dideskripsikan ke dalam bentuk yang lebih spesifik.



# Interface

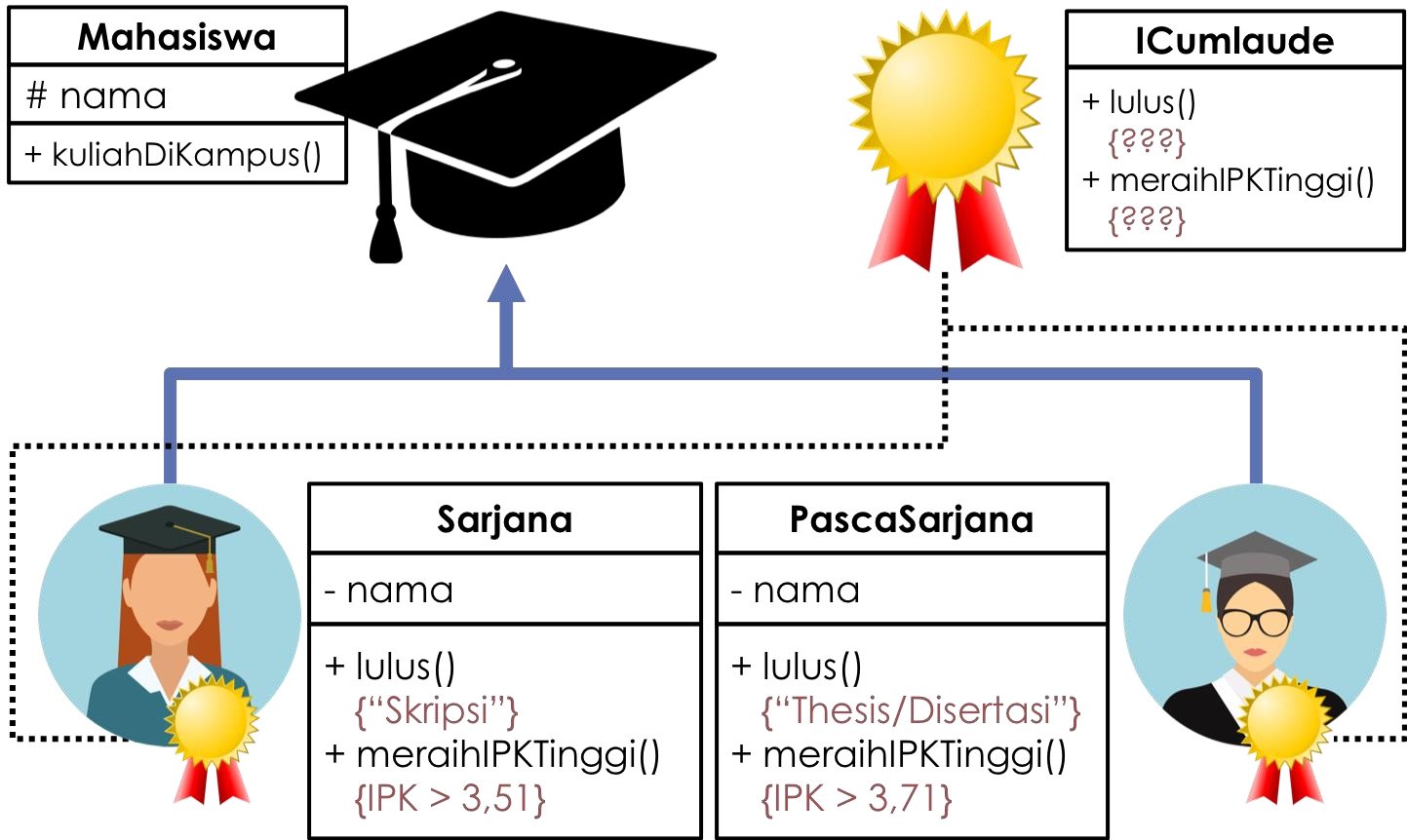
*Interface* adalah struktur data yang hanya berisi *abstract methods*. Tidak ada apa-apa selain *method abstract* pada *interface*, termasuk atribut *getter* dan *setter*.

**Karakteristik:**

1. Tidak ada apa-apa di dalamnya selain *abstract methods*.
2. Di konvensi bahasa pemrograman Java, namanya dianjurkan untuk selalu diawali dengan huruf kapital ‘I’.
3. Selalu dideklarasikan dengan menggunakan kata kunci interface.
4. Diimplementasikan dengan menggunakan kata kunci implements.

**Kegunaan:**

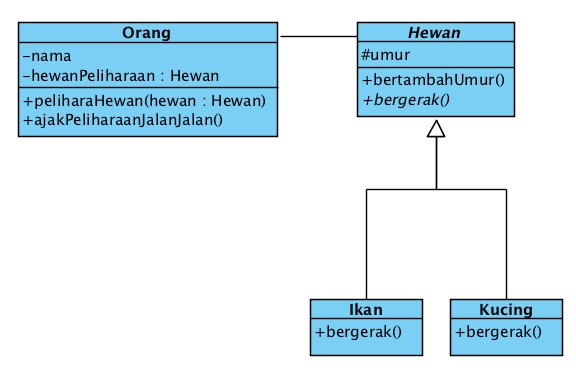
Bertindak seperti semacam kontrak/syarat yang HARUS dipenuhi bagi suatu class agar class tersebut dapat dianggap sebagai ‘sesuatu yang lain’.



# Praktikum Percobaan 1: Abstract Class

Di dunia ini terdapat banyak jenis hewan. Semua hewan memiliki beberapa karakteristik yang sama, seperti contohnya semua hewan memiliki umur, hewan apapun itu, umurnya akan bertambah sama jumlahnya setiap tahun.

Selain karakteristik yang sama, masing-masing hewan juga memiliki karakteristik yang berbeda satu dengan yang lainnya. Contohnya dalam hal **bergerak**. Cara kucing bergerak berbeda dengan cara ikan bergerak. Kucing bergerak dengan cara melangkahkan kaki-kakinya sedangkan ikan bergerak dengan cara menggerakkan siripnya.

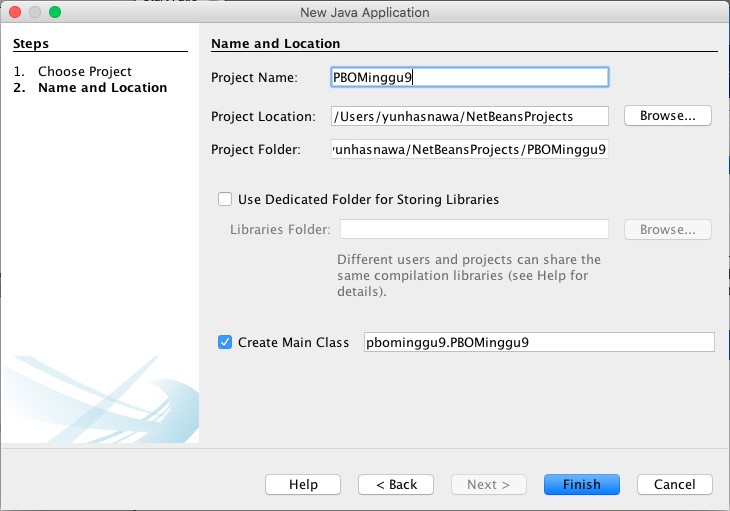


Setiap orang yang memelihara hewan dapat mengajak hewan peliharaannya berjalan (membuat agar hewan peliharaannya bergerak). Namun orang yang memelihara hewan yang berbeda, akan berbeda pula cara hewan peliharaannya dalam bergerak.

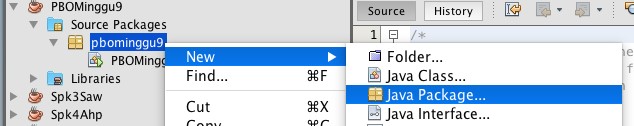
Pada percobaan pertama ini kita akan membuat sebuah program yang menggambarkan

skenario di atas dengan memanfaatkan **abstract class**.

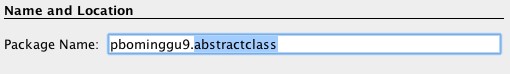
1. Buatlah sebuah project baru di NetBeans dengan nama **PBOMinggu9**



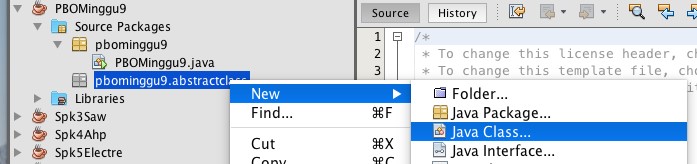
1. Pada package **pbominggu9**, tambahkan package baru dengan cara klik kanan nama package  New  Java Package…



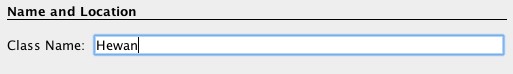
1. Beri nama package tersebut dengan nama **abstractclass**. Semua class yang dibuat pada percobaan 1 ini **diletakkan pada package yang sama**, yaitu *package abstractclass* ini,



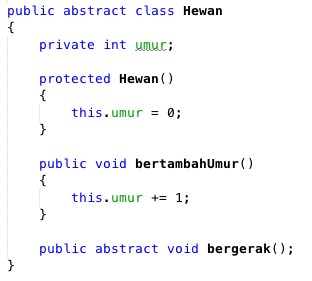
1. Pada package baru tersebut tambahkan *class* baru.



1. Beri nama *class* baru tersebut, yaitu class **Hewan**.



1. Pada *class* Hewan tersebut, ketikkan kode berikut ini.



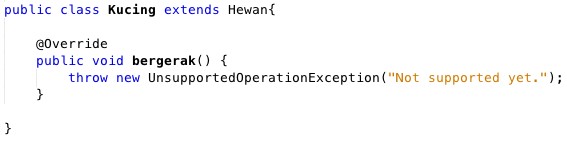
Class Hewan tersebut adalah class abstract berisi property dan method biasa, ditambah sebuah *method abstract* bernama **bergerak()**. *Method* tersebut didepannya terdapat kata kunci **abstract** dan tidak memiliki badan fungsi. Method ini nantinya akan di-*override* oleh *class* mana saja yang menjadi *class* turunan dari class Hewan tersebut.

1. Dengan cara yang sama, buatlah class dengan nama **Kucing** yang meng-*extend* class Hewan. Di dalam class Kucing tersebut, setelah Anda menuliskan kode seperti di bawah, maka akan muncul

ikon lampu peringatan. Klik lampu tersebut dan kemudian pilih **implement all abstract methods**.



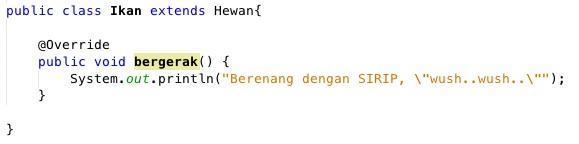
1. Maka akan secara otomatis dibuatkan fungsi yang meng-*override* fungsi *abstract* **bergerak()** yang ada pada class hewan.



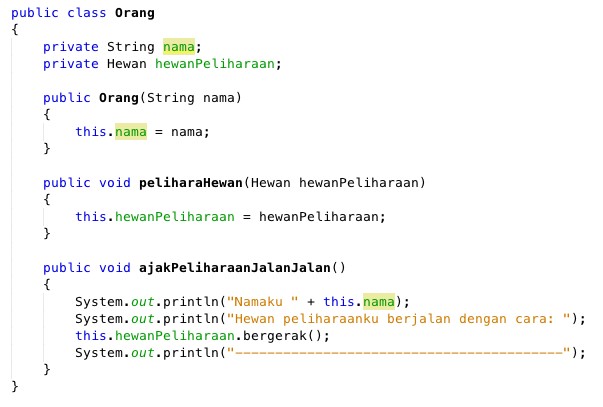
1. Ubahlah badan fungsi tersebut dengan mengganti kode didalamnya menjadi seperti berikut.



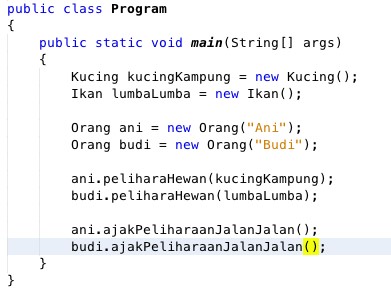
1. Dengan cara yang sama seperti ketika Anda membuat class Kucing, buatlah class Hewan baru bernama **Ikan** dan buatlah kodenya seperti pada gambar dibawah.



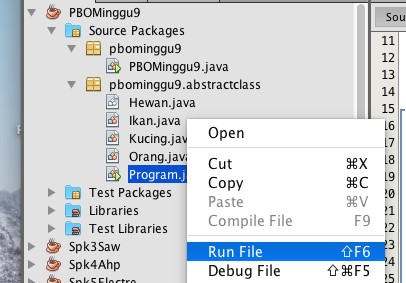
1. Selanjutnya, buatlah class biasa baru yang bernama class **Orang**. Class ini adalah class yang menjadi pengguna dari *class abstract* Hewan yang sudah dibuat sebelumnya. Ketikkan pada class Orang tersebut, baris-baris kode seperti di bawah.



1. Terakhir, buatlah sebuah *Main Class* baru di dalam *package* yang sama. Beri nama class baru tersebut dengan nama class **Program**. Ketikkan didalamnya seperti kode di bawah ini.

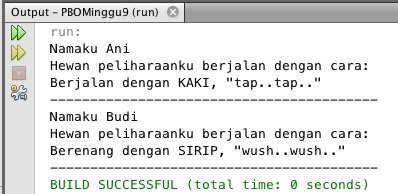


1. Jalankan class tersebut dengan cara klik kanan pada class Program kemudian pilih **Run File** (Shift + F6).



14.

Perhatikan dan amati hasilnya!



15. Pertanyaan diskusi:

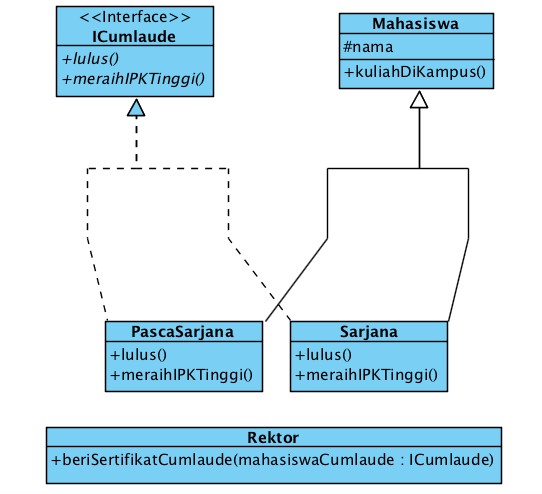
Bolehkah apabila sebuah class yang meng-*extend* suatu *abstract class* tidak mengimplementasikan *method abstract* yang ada di class induknya? Buktikan!

* Tidak, sebuah calss yang meng-extend suatu abstract class harus mengiplementasikan method abstract yang ada di induknya.



# Percobaan 2: Interface

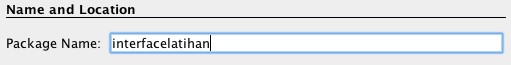
Pada sebuah wisuda, seorang Rektor akan memberikan penghargaan sertifikat *Cumlaude* pada semua mahasiswa yang memenuhi persyaratan. Persyaratan agar seorang mahasiswa dapat disebut sebagai *Cumlaude* berbeda-beda antara mahasiswa Sarjana dan Pasca Sarjana.



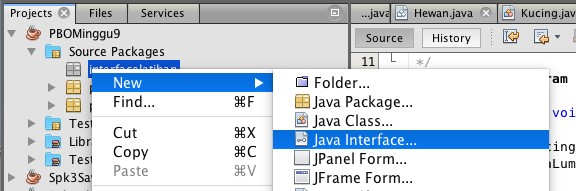
Untuk menjadi *cumlaude*, mahasiswa Sarjana harus mengerjakan **skripsi** dan memiliki IPK lebih tinggi dari 3,51. Sedangkan untuk mahasiswa Pasca Sarjana, mereka harus mengerjakan **tesis** dan meraih IPK lebih tinggi dari 3,71.

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

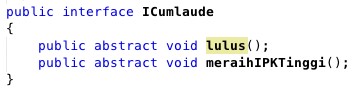
1. Pada **project yang sama**, buatlah sebuah package baru bernama **interfacelatihan**.



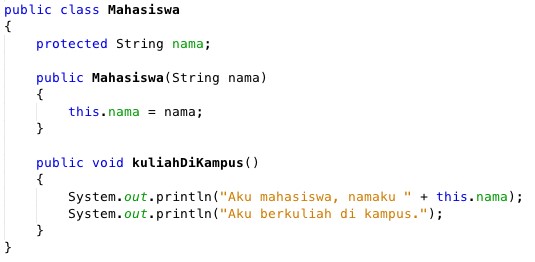
1. Pada package yang baru dibuat tersebut, tambahkan sebuah **interface** baru dengan cara klik kanan pada package  **New**  **Java Interface**… Beri nama interface baru tersebut dengan nama **ICumlaude**.



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

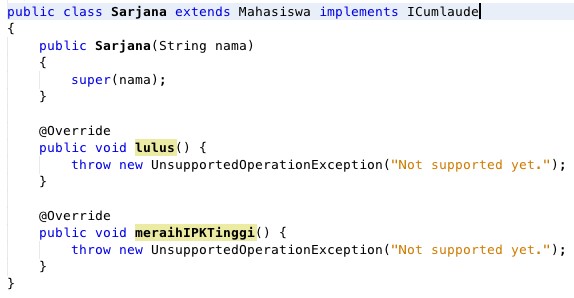


1. Berikutnya, buatlah sebuah class baru bernama **Mahasiswa** dengan baris-baris kode seperti dibawah ini.

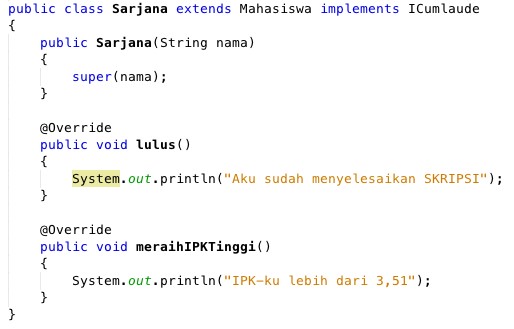


1. Selanjutnya, buatlah class baru bernama **Sarjana** yang merupakan **turunan** dari class

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.

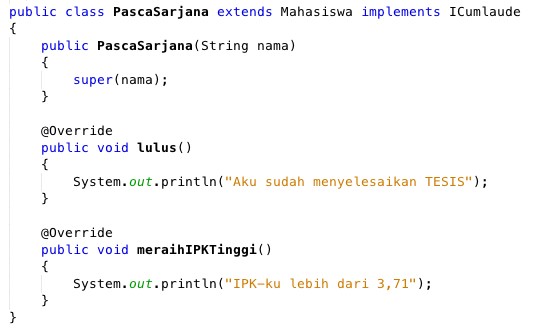


1. Selanjutnya sesuaikan isi dari *method* **lulus()** dan **meraihIPKTinggi()** agar sama dengan baris kode di bawah.

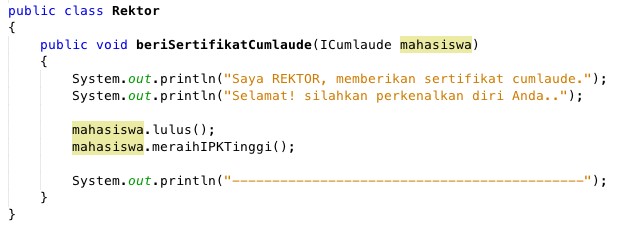


Perhatikan pada baris kode di atas, class Sarjana meng-**extend** class Mahasiswa, ini berarti, **Sarjana adalah Mahasiswa** sementara itu agar semua objek dari class Sarjana ini nantinya dapat disebut sebagai **Cumlaude** maka ia harus meng-**implements** interface **ICumlaude**.

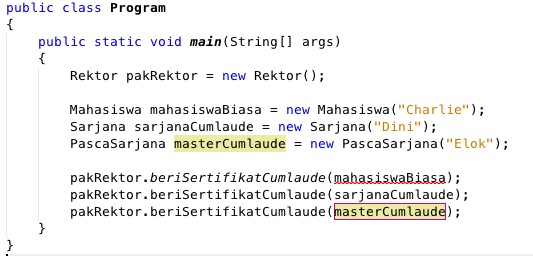
1. Kemudian dengan **cara yang sama** buatlah class baru bernama **PascaSarjana** dengan baris kode seperti di bawah ini.



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



1. Terakhir, buatlah sebuah class baru bernama **Program** yang diletakkan pada **package yang sama** dengan class-class percobaan 2. Tambahlan baris kode berikut ini:



public class Rektor {

    public void beriSertifikatCumlaude(Sarjana sarjana) {

        System.**out**.println("Saya Rektor, memberikan sertifikat cumlaude");

        System.**out**.println("Selamat! Silahkan perkenalkan diri Anda...");

        sarjana.lulus();

        sarjana.meraihIPKTinggi();

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

    }

    public void beriSertifikatCumlaude(PascaSarjana pascaSarjana) {

        System.**out**.println("Saya Rektor, memberikan sertifikat cumlaude");

        System.**out**.println("Selamat! Silahkan perkenalkan diri Anda...");

        pascaSarjana.lulus();

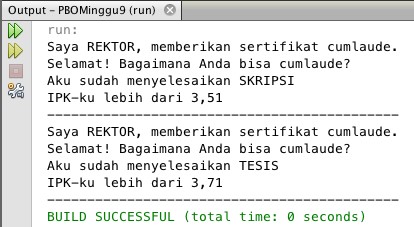
        pascaSarjana.meraihIPKTinggi();

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

    }

}

1. 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:



1. Pertanyaan diskusi:
   1. Mengapa pada langkah nomor 9 terjadi error? Jelaskan!

* Karena pada class rektor tidak terdapat parameter pada method berisertifikatCumlaude().
  1. Dapatkah method **kuliahDiKampus()** dipanggil dari objek **sarjanaCumlaude** di class **Program**? Mengapa demikian?
* Bisa, karena class sarjanaCumlaude meng-inherit class mahasiswa yang mana hal ini memberi akses class sarjanaCumlaude agar bisa menggunakan method didalam class Mahasiswa.

public class Main {

    public static void main(String[] args) {

        Rektor pakRektor = new Rektor();

        Mahasiswa budi = new Mahasiswa("Budi");

        Sarjana dimas = new Sarjana("Dimas");

        PascaSarjana bambang = new PascaSarjana("Bambang");

*// pakRektor.beriSertifikatCumlaude(budi);*

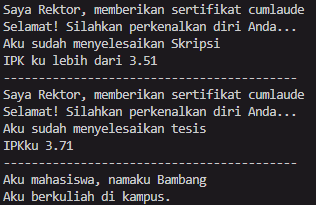
        pakRektor.beriSertifikatCumlaude(dimas);

        pakRektor.beriSertifikatCumlaude(bambang);

        bambang.kuliahDiKampus();

    }

}



* 1. Dapatkah method **kuliahDiKampus()** dipanggil dari parameter **mahasiswa** di method **beriSertifikatCumlaude()** pada class **Rektor?** Mengapa demikian?
* Bisa, karena terdapat parameter dengan objek sarjana/pascaSarjana dalam method beriSertifikatCumlaude().

public class Rektor {

    public void beriSertifikatCumlaude(Sarjana sarjana) {

        System.**out**.println("Saya Rektor, memberikan sertifikat cumlaude");

        System.**out**.println("Selamat! Silahkan perkenalkan diri Anda...");

        sarjana.lulus();

        sarjana.meraihIPKTinggi();

        sarjana.kuliahDiKampus();

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

    }

    public void beriSertifikatCumlaude(PascaSarjana pascaSarjana) {

        System.**out**.println("Saya Rektor, memberikan sertifikat cumlaude");

        System.**out**.println("Selamat! Silahkan perkenalkan diri Anda...");

        pascaSarjana.lulus();

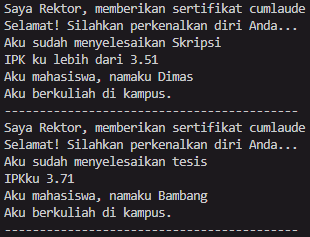
        pascaSarjana.meraihIPKTinggi();

        pascaSarjana.kuliahDiKampus();

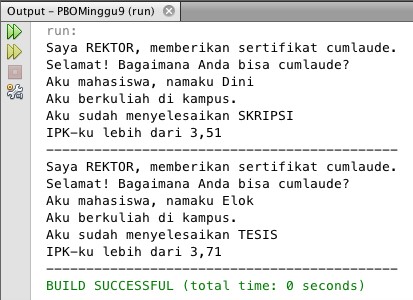
        System.**out**.println("------------------------------------------");

    }

}



* 1. Modifikasilah method **beriSertifikatCumlaude()** pada class **Rektor** agar hasil eksekusi class **Program** menjadi seperti berikut ini:



public class Rektor {

    public void beriSertifikatCumlaude(Sarjana sarjana) {

        System.**out**.println("Saya Rektor, memberikan sertifikat cumlaude");

        System.**out**.println("Selamat! Silahkan perkenalkan diri Anda...");

        sarjana.kuliahDiKampus();

        sarjana.lulus();

        sarjana.meraihIPKTinggi();

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

    }

    public void beriSertifikatCumlaude(PascaSarjana pascaSarjana) {

        System.**out**.println("Saya Rektor, memberikan sertifikat cumlaude");

        System.**out**.println("Selamat! Silahkan perkenalkan diri Anda...");

        pascaSarjana.kuliahDiKampus();

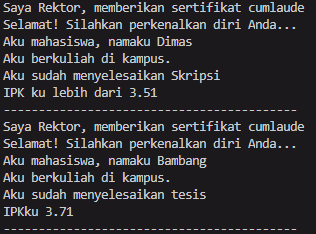
        pascaSarjana.lulus();

        pascaSarjana.meraihIPKTinggi();

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

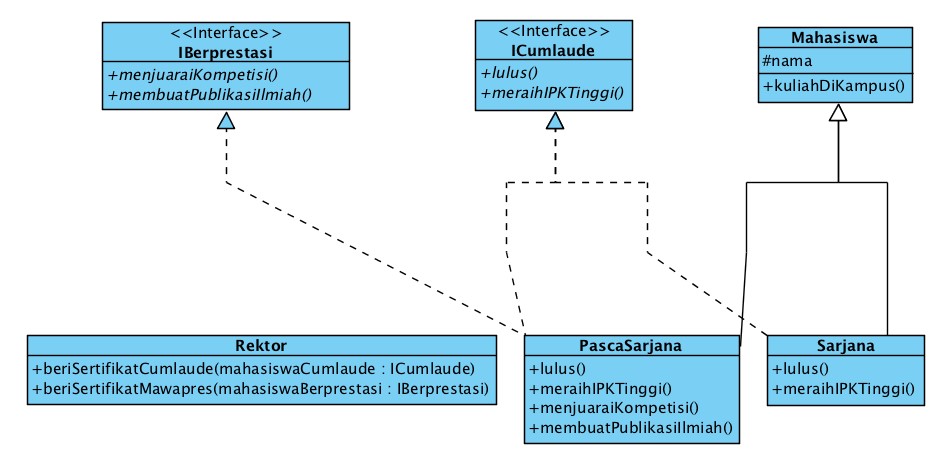
    }

}



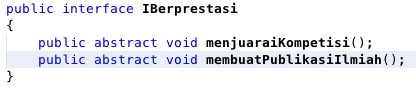
# Percobaan 3: Multiple Interfaces Implementation

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 mem**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()**.

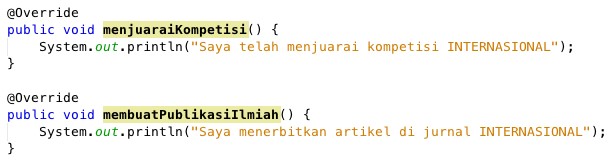
1. Pada package **yang sama dengan package pada Percobaan 2**, tambahkan sebuah *interface* baru yang bernama **IBerprestasi**. Tambahkan baris kode seperti berikut didalamnya.



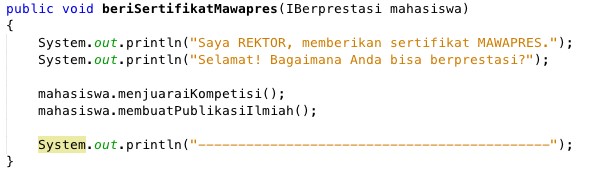
1. Selanjutnya, **modifikasilah** class **PascaSarjana** dengan menambahkan interface baru **IBerprestasi** dibelakang kata kunci **implements**. Lalu dengan cara yang sama seperti sebelumnya, kliklah ikon lampu peringatan untuk **meng-*generate*** semua method *abstract* dari interface IBerprestasi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



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



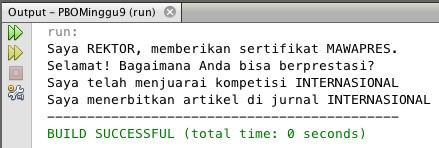
1. Tambahkan *method* **beriSertifikatMawapres()** dengan baris kode seperti di bawah, pada class **Rektor**.



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



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



public class PascaSarjana extends Mahasiswa implements *iCumlaude* {

    public PascaSarjana(String nama) {

        super(nama);

    }

    public void lulus() {

        System.**out**.println("Aku sudah menyelesaikan tesis");

    }

    public void meraihIPKTinggi() {

        System.**out**.println("IPKku 3.71");

    }

    public void menjuaraiKompetisi() {

        System.**out**.println("Saya Menjuarai kompetisi INTERNASIONAL");

    }

    public void membuatPublikasiIlmiah() {

        System.**out**.println("Saya menerbitkan artikel di jurnal internasional");

    }

}

public void beriSertifikatMawapres(PascaSarjana mahasiswa) {

        System.**out**.println("Saya rektor, memberikan sertifikat mawapres");

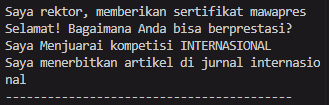
        System.**out**.println("Selamat! Bagaimana Anda bisa berprestasi?");

        mahasiswa.menjuaraiKompetisi();

        mahasiswa.membuatPublikasiIlmiah();

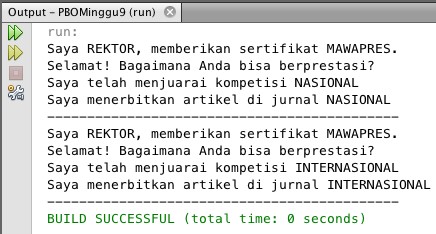
        System.**out**.println("-----------------------------------------");

    }



1. Pertanyaan diskusi:

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**.



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");

    }

    public void menjuaraiKompetisi() {

        System.**out**.println("Saya Menjuarai kompetisi INTERNASIONAL");

    }

    public void membuatPublikasiIlmiah() {

        System.**out**.println("Saya menerbitkan artikel di jurnal internasional");

    }

}

public void beriSertifikatMawapres(Sarjana mahasiswa) {

        System.**out**.println("Saya rektor, memberikan sertifikat mawapres");

        System.**out**.println("Selamat! Bagaimana Anda bisa berprestasi?");

        mahasiswa.menjuaraiKompetisi();

        mahasiswa.membuatPublikasiIlmiah();

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

    }

