Overriding

Apa itu overriding?

Method yang ada pada parent class(Superclass) didefinisikan kembali oleh kelas anaknya (subclass).

Jika kita panggil method yang udah di-override dari instance kelas anaknya, maka method yang dipanggil itu punyanya si kelas anak bukan punya kelas parentnya lagi.

Pada overriding kita hanya bisa override methodnya.

try this at home

```
public class Induk{
      public void panggilAku(){
            System.out.println("");
            System.out.println("Hallo, ini induk yang dipanggil");
       }
 }
```

```
public class Anak{
//method sama dengan method induk
     public void panggilAku(){
           System.out.println("");
           System.out.println("Hallo, ini anak yang dipanggil");
     }
}
```

```
public class DemoOverride{
      public static void main(String args[]){
            Anak a= new Anak();
            a.panggilAku();
       }
 }
```

Output dari program:

```
C:\Users\acer\Desktop>java DemoOverride
Hallo, ini anak yang dipanggil
```

Abstract Class

Apa itu Abstract class?

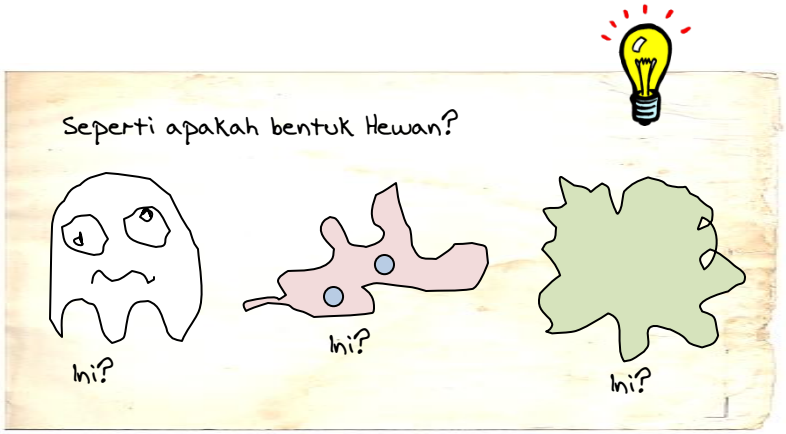
kelas murni yang tidak boleh memiliki objek, dan satu/lebih method-methodnya yang abstract harus diimplementasikan (override) oleh kelas turunannya.

Bentuk Umum:

```
[modifier] abstract class NamaKelas {
      // deklarasi attribute
      // definisi/prototype method
}
```

analogi

Sebuah kelas Dosen bisa diinstansiasi menjadi heru, heri, badu, budi, dsb, tetapi tidak mungkin dapat menginstansiasi kelas MakhlukHidup, kelas Hewan. Karena kedua kelas tersebut terlalu umum (abstract), kelas seperti inilah yang disebut kelas abstract. Dibutuhkan kelas turunan yang lebih khusus.



analogi

Bila kelas MahlukHidup mempunyai method bernafas, maka tidak dapat ditentukan cara suatu mahluk hidup tersebut bernafas (dengan paru-paru, insang, atau stomata), method seperti inilah yang disebut method abstract. Dibutuhkan kelas turunan yang khusus dan method override dari method yang abstract.

Ketika kamu sudah tidak bisa membayangkan bagaimana bentuk suatu kelas seperti MakhlukHidup atau pun Hewan karena terlalu abstraknya/terlalu umum, maka kelas tersebut bisa dijadikan kelas abstrak.

try this at home

```
abstract class Hewan {
    protected String nama;
    protected int jumKaki;
    protected boolean bisaTerbang = false;
    public Hewan(String nama, int kaki, boolean terbang) {
      this.nama = nama;
      jumKaki = kaki;
      bisaTerbang = terbang;
    }
    public abstract void bersuara();
    public static void makan() {
      System.out.println("nyam, nyam, nyam");
    }
     public void lihatHewan() {
      System.out.println("");
      System.out.println("nama          : "+nama);
      System.out.println("jumlah kaki   : "+jumKaki);
      System.out.println("bisa terbang : "+bisaTerbang);
     }
}
```

```
class Sapi extends Hewan {
      public Sapi() {
            super("sapi", 4, false);
      }
      public void bersuara() {
            System.out.println("\n moooaahhhh,mooooaaahhh");
      }
      public static void main(String[] args) {
            Sapi s = new Sapi();
            s.lihatHewan();
            s.bersuara();
      }
}
```

```
class Perkutut extends Hewan {
      public Perkutut(){
            super("perkutut",2,true);
       }
      public void bersuara() {
            System.out.println("\ncuit, cuit, cuit");
      }
      public static void main(String[] args) {
            Perkutut p = new Perkutut();
            p.lihatHewan();
            p.bersuara();

      }
}
```

Output dari program:

```
C:\Users\acer\Desktop>java Sapi

nama            : sapi
jumlah kaki     : 4
bisa terbang    : false

 moooaahhhh,mooooaaahhh

C:\Users\acer\Desktop>java Perkutut

nama            : perkutut
jumlah kaki     : 2
bisa terbang    : true

cuit, cuit, cuit
```

Interface

Apa itu interface?

Interface adalah kelas yang paling abstract, yang berisi daftar deklarasi method (seluruh method belum memiliki implementasi).

Analogi Interface

Interface dapat dianalogikan sebagai kontrak yang dapat dipakai oleh setiap kelas.
Dalam kehidupan nyata dapat diketahui ada manusia yang bekerja sebagai da'i, dosen, tentara, penyanyi, pengacara, dan sebagainya, tentunya manusia-manusia tersebut selain harus memiliki method standard sebagai seorang manusia, juga harus memiliki method yang sesuai dengan pekerjaannya.
Dengan demikian untuk membuat objek seorang budi bekerja sebagai dosen, harus dibuat kelas yang merupakan turunan kelas manusia yang meng-implementasikan interface dosen.

Bagaimana bentuk deklarasinya?

```
[modifier] interface NamaInterface {
       // deklarasi konstanta
       // deklarasi method

} // catatan : modifier static tidak boleh digunakan dalam interface
```

Bagaimana bentuk Implementasinya?

```
[modifier] class NamaKelas implements NamaInterface {
       // penggunaan konstanta
       // implementasi method
}
```

try this at home

```
public interface TkPhone{
       static final String MAKER= "YES-KIA";
       public int getHarga(int id);
}
```

```
public class PhoneImpl implements TkPhone{
       public int getHarga(int id){
              int hr=0;
              if (id==1){
                     hr=500;
              }
              if (id==2){
                     hr=1000;
              }
              return hr;
       }
}
```

Output dari program:

```
C:\Users\acer\Desktop>java TokoHP

HP ini dibuat oleh PT YES-KIA
Harga HP in sekitar $ 500
```

```
public class TokoHP{
       public static void main(String args[]){
              PhoneImpl pp = new PhoneImpl();

              System.out.println("");
              System.out.println("HP ini dibuat oleh PT "+pp.MAKER);
              System.out.println("Harga HP in sekitar $ "+pp.getHarga(1));
       }
}
```

Keyword super

Maksudnya??

Keyword super digunakan untuk merefer/ mengacu superclass dari suatu class, yaitu member dari suatu superclass, baik method maupun atribut.

Supaya kita lebih mengerti, maka mari kita lihat dan coba kode berikut ini!

try this at home

```
public class PersegiPanjang{
      private int p;
      private int l;
      public PersegiPanjang(int p, int l){
            this.p=p;
            this.l=l;
      }
      public int Luas(){
            return p*l;
      }
}
```

```
public class Balok extends PersegiPanjang{
      private int t;

      public Balok(int p, int l,int t){
            super(p,l);
            this.t=t;
      }

      public void LihatVolume(){
            int v= super.Luas()*t;
            System.out.println("");
            System.out.println("Volume = "+v);
      }

      public static void main(String args[]){
            Balok b = new Balok(5,4,3);
            b.LihatVolume();
      }
}
```