

:: abstract class

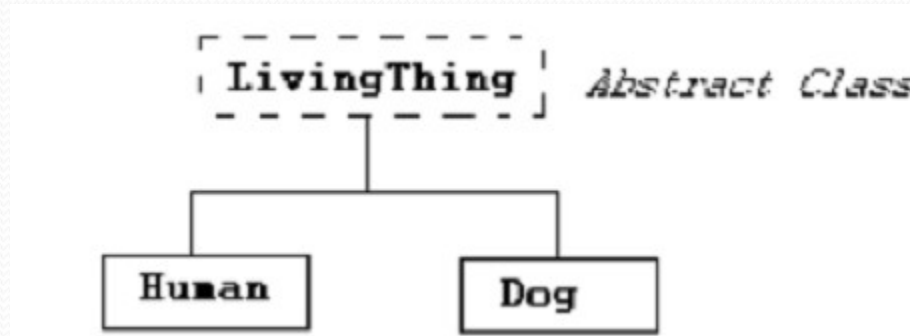
Java



abstract class

- Saat kita membuat sebuah superclass, kita tahu bahwa kita dapat memakai semua metode yang dimilikinya pada class anaknya. Atau kalau kita butuh metode tersebut diperbarui, kita tinggal melakukan override pada metode tersebut.
- Tapi, pada suatu kasus, kita ingin membuat superclass yang bernama LivingThing(makhluk hidup).
- Class ini punya beberapa method yang sudah kita kenal seperti makan, bernafas, tidur, dan berjalan. Ada satu metode yang tidak dapat kita generalisasi. Contohnya adalah method berjalan.
- Human sebagai subclass dari makhluk hidup berjalan dengan 2 kaki, sedangkan Dog berjalan dengan 4 kaki. Untuk membuat superclass yang umum, kita bisa membuat suatu superclass yang memiliki beberapa metode dengan implementasi dan beberapa tidak. Class ini disebut abstract class.

abstract class



Secara umum, abstract class adalah class yang tak dapat diinstansiasi. Biasanya abstract class berada dalam hierarki tertinggi dalam PBO (pemrograman berorientasi objek).

Metode di dalam abstract class yang tidak berisi kode program adalah abstract method (metode abstract). Nantinya, method ini harus di-override oleh subclass-nya. Jadi, kita hanya menulis deklarasi method tanpa tubuh metode dan memakai keyword abstract. Contohnya:

```
public abstract void iniMetode();
```


abstract class

- Deklarasi Method ini disebut sebagai prototype method.
- Sekarang, kita coba buat abstract class pertama kita:

```
1 public abstract class LivingThing
2 {
3     public void breath(){
4         System.out.println("Living Thing breathing...");
5     }
6     public void eat(){
7         System.out.println("Living Thing eating...");
8     }
9     /**
10    * -- abstract method walk
11    * Kita ingin supaya metode ini di-override oleh subclass
12    * dari class LivingThing
13    */
14    public abstract void walk();
15 }
```

Ajib Susanto

abstract class

- Suatu metode yang sudah dideklarasikan abstract harus diakhiri tanpa memakai tubuh program.
- Jika ditulis seperti ini:

```
public abstract class LivingThing
{
    public void breath(){
        System.out.println("Living Thing breathing...");
    }
    public void eat(){
        System.out.println("Living Thing eating...");
    }
    /**
     * -- abstract method walk
     * Kita ingin supaya metode ini di-override oleh subclass
     * dari class LivingThing
     */
    public abstract void walk()
    {}
}
```

Ajib Susanto

abstract class

- maka akan terjadi error seperti berikut:

```
----- Javac -----  
LivingThing.java:14: abstract methods cannot have a body  
    public abstract void walk()  
                        ^  
1 error  
Normal Termination  
Output completed (2 sec consumed).
```

Ketika sebuah class dibuat sebagai subclass dari class LivingThing, class itu harus meng-override metode abstract walk(). Jika tidak, class tersebut tidak akan dapat dikompilasi dan akan terjadi error.

abstract class

- Contoh pembuatan subclass yang benar:

```
public class Human extends LivingThing
{
    public void walk()
    {
        System.out.println("Human walks...");
    }
}
```

Jika class Human tidak meng-override metode walk(), maka akan muncul pesan error seperti berikut:

```
----- Javac -----
Human.java:1: Human is not abstract and does not override abstract method walk() in LivingThing
public class Human extends LivingThing
      ^
1 error
Normal Termination
Output completed (0 sec consumed)
```




abstract class

- Tapi, walau class Human tidak meng-override metode yang lain seperti **breath()** dan **eat()** (yang tidak abstract) tidak akan terjadi error.
- fungsi abstract class tak lain adalah untuk membuat prototype bagi class di tingkatan paling atas dalam hierarki class dan subclass-nya yang menyediakan detail implementasi dari abstract class tersebut.



Contoh abstract class

```
public abstract class Bangun2D {  
    public abstract void cetakLuas();  
    public abstract void cetakKeliling();  
}
```

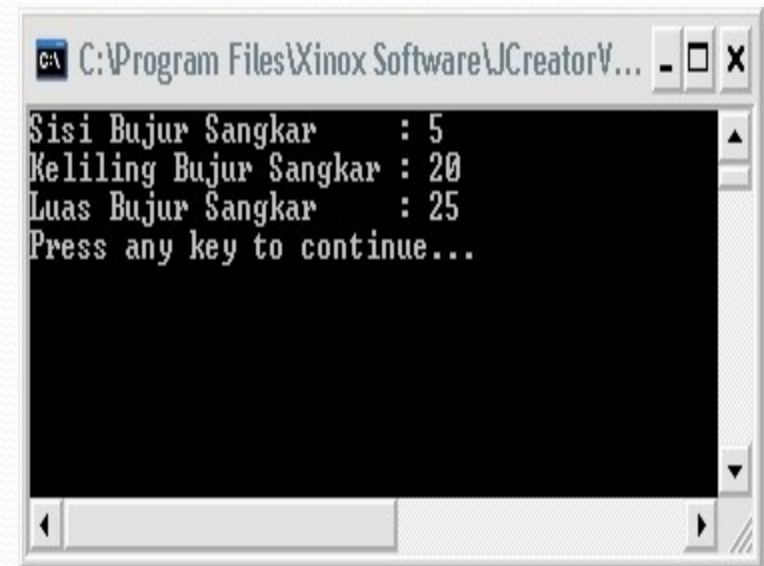
Class turunan

```
public class BujurSangkar extends Bangun2D {  
    private int sisi;  
    public BujurSangkar(int sisi) {  
        this.sisi = sisi;  
    }  
  
    public void cetakLuas() {  
        int luas = sisi * sisi;  
        System.out.println(luas);  
    }  
  
    public void cetakKeliling() {  
        int keliling = 4 * sisi;  
        System.out.println(keliling);  
    }  
}
```


Class Demo

```
class Bangun2DDemo
{
public static void main(String[] args)
{
    BujurSangkar bs=new BujurSangkar(5);
    bs.cetakKeliling();
    bs.cetakLuas();
}

}
```



```
C:\Program Files\Xinox Software\JCreatorV...
Sisi Bujur Sangkar : 5
Keliling Bujur Sangkar : 20
Luas Bujur Sangkar : 25
Press any key to continue...
```

:: interface vs abstract class

- Perbedaan interface dan abstract class cukup terlihat dari pemakaiannya. interface itu diimplementasikan dan abstract class itu diturunkan (diwariskan).

```
1 public interface Kamera
2 {
3     public void setPixel(float pixel);
4     public void ambilGambar();
5 }
```

```
1 public interface Radio
2 {
3     public void setGelombang(String gel);
4 }
```

Ajib Susanto

:: interface vs abstract class

Keduanya dapat dipakai bersamaan dalam satu class. Misalkan seperti contoh di bawah ini:

```
1  abstract class Telepon
2  {
3      protected long nomer;
4      public void telpon()
5      {
6          System.out.println("Sedang menelepon");
7      }
8
9  }
```

:: interface vs abstract class

- Tambahkan abstract class di atas ke class HandPhone yang sudah kita buat, sehingga class HandPhone menjadi:

```
1 class HandPhone extends Telepon implements Radio,Kamera
2 {
3     private String gelombang;
4     private float pixel;
5
6     public void ambilGambar()
7     {
8         System.out.println("Gambar terambil...");
9     }
10    public void setGelombang(String gel)
11    {
12        this.gelombang=gel;
13    }
14    public void setPixel(float pixel)
15    {
16        this.pixel=pixel;
17    }
18    public void setNomer(long no)
19    {
20        this.nomer=no;
21    }
22 }
```


Demo Class

```
1 class TeleponDemo
2 {
3     public static void main(String[] args)
4     {
5         Handphone hp=new Handphone();
6         hp.setNomor(8183434);
7         hp.telpon();
8         hp.setPixel(1024);
9         hp.ambilGambar();
10        hp.setGelombang("FM 101.2");
11    }
12 }
```



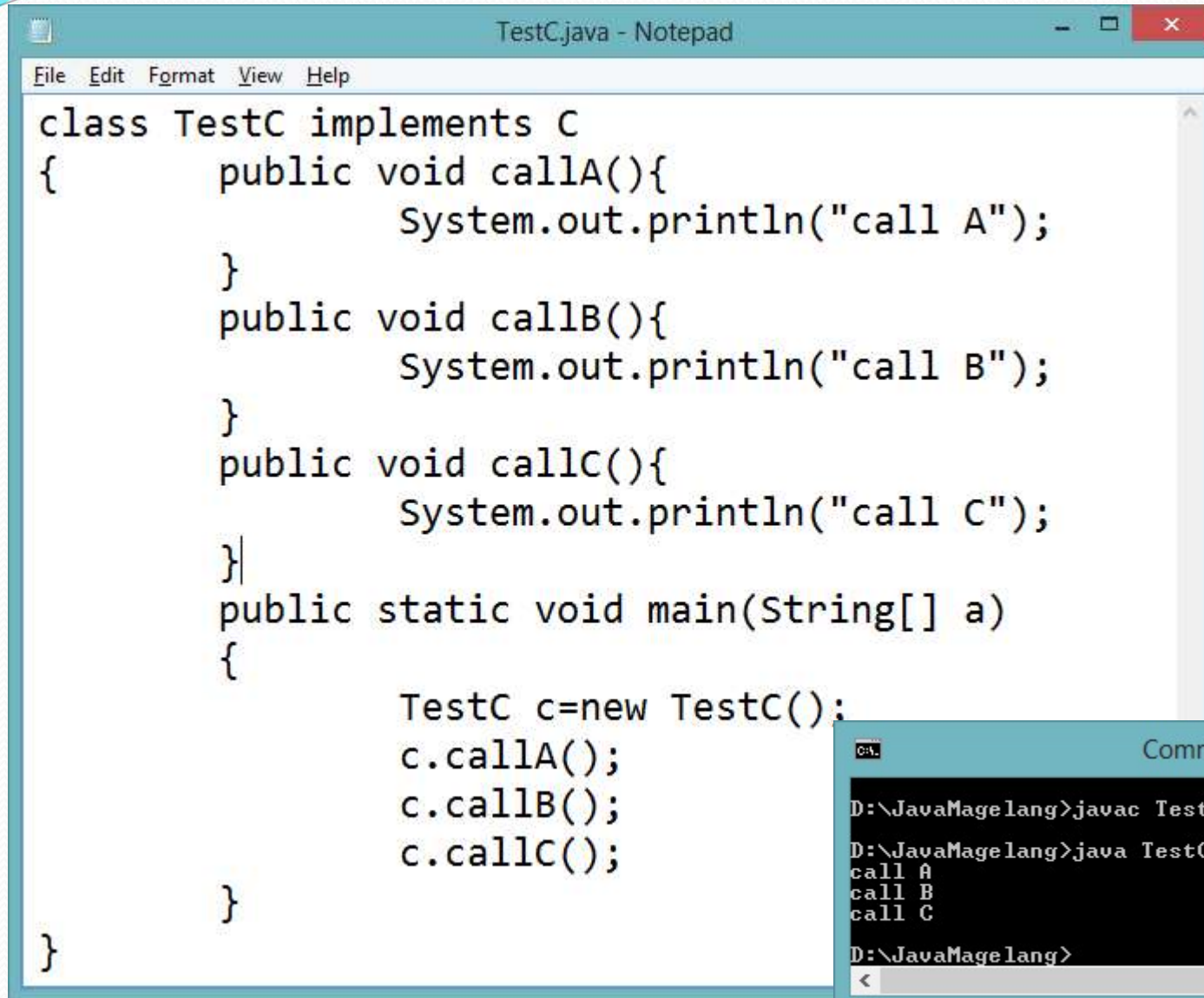
Contoh

```
A.java - Notepad
File Edit Format View Help
interface A
{
    void callA();
}
```

```
B.java - Notepad
File Edit Format View Help
interface B
{
    void callB();
}
```

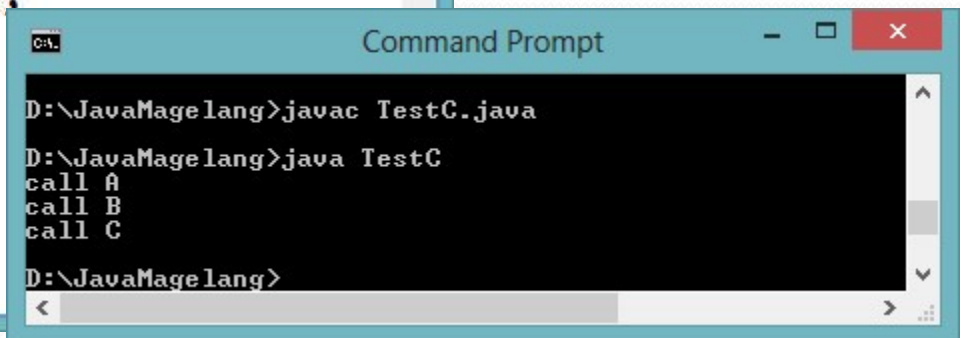
```
C.java - Notepad
File Edit Format View Help
interface C extends A,B
{
    void callC();
}
```


Contoh



```
TestC.java - Notepad
File Edit Format View Help

class TestC implements C
{
    public void callA(){
        System.out.println("call A");
    }
    public void callB(){
        System.out.println("call B");
    }
    public void callC(){
        System.out.println("call C");
    }
    public static void main(String[] a)
    {
        TestC c=new TestC();
        c.callA();
        c.callB();
        c.callC();
    }
}
```



```
C:\> Command Prompt

D:\JavaMagelang>javac TestC.java
D:\JavaMagelang>java TestC
call A
call B
call C
D:\JavaMagelang>
```

Contoh

Zoo3Abstract.java - Notepad

File Edit Format View Help

```
abstract class Binatang{ //abstract class
    abstract void makan(); //abstract method
    abstract void tidur();
    void mati(){
        System.out.println("mati.....");
    }
}
class Harimau extends Binatang{
    void makan(){
        System.out.println("harimau makan.....");
    }
    void tidur(){
        System.out.println("harimau tidur.....");
    }
}
class Bebek extends Binatang{
    void makan(){
        System.out.println("bebek makan.....");
    }
    void tidur(){
        System.out.println("bebek tidur.....");
    }
}
```


Contoh

Zoo3Abstract.java - Notepad

File Edit Format View Help

```
public class Zoo3Abstract{

    static void test(Binatang a)
    {
        a.makan();
        a.tidur();
        a.mati();
    }
    public static void main(String[] a)
    {
        Harimau macan=new Harimau();
        Bebek donald=new Bebek();
        //Binatang b=new Binatang();
        //--> error abstract class tdk bisa dibuat object
        test(macan);
        test(donald);
    }
}
```

Rehat Sejenak



Ajib Susanto