Interface & Abstract Class

Abstract Class

- ➔ Sebuah Class yang masih dalam bentuk abstract. Karena bentuknya yang masih abstract maka tidak bisa dibuat langsung menjadi objek.
- ➔ Cara membuat Abstract Class yaitu dengan menambahkan keyword "Abstract" pada sebuah class.

```
public abstract class Shape{.....
```

Abstract Method

- ➔ Method yang tidak memiliki implementasi atau tidak ada bentuk konkritnya.
- ➔ Intinya Method abstrak merupakan method yang tidak memiliki isi. (Hanya nama saja)

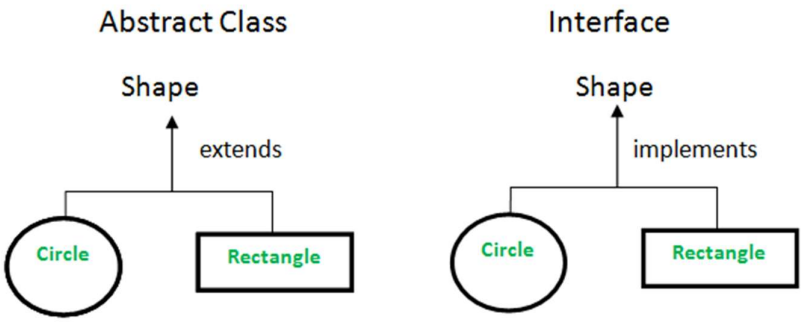
```
// ini abstrak method
abstract void area();

// ini bukan abstrak method
// karena punya implementasi di body method
void luas(){
        System.out.println("Luas");
}
```

Mengapa harus menggunakan Abstract Class?

Abstract Class biasanya digunakan sebagai class induk dari class lainnya. Child class akan membuat versi konkretnya.

Tapi pada suatu konidsi terterntu, class induk tidak ingin di buat sebagai objek, Karena code methodnya belum jelas mau di implementasikan seperti apa (Isi dari methodnya belum tau akan di implementasikan seperti apa). Maka dari itu class dijadikan abstract.

Interface VS Abstract Class

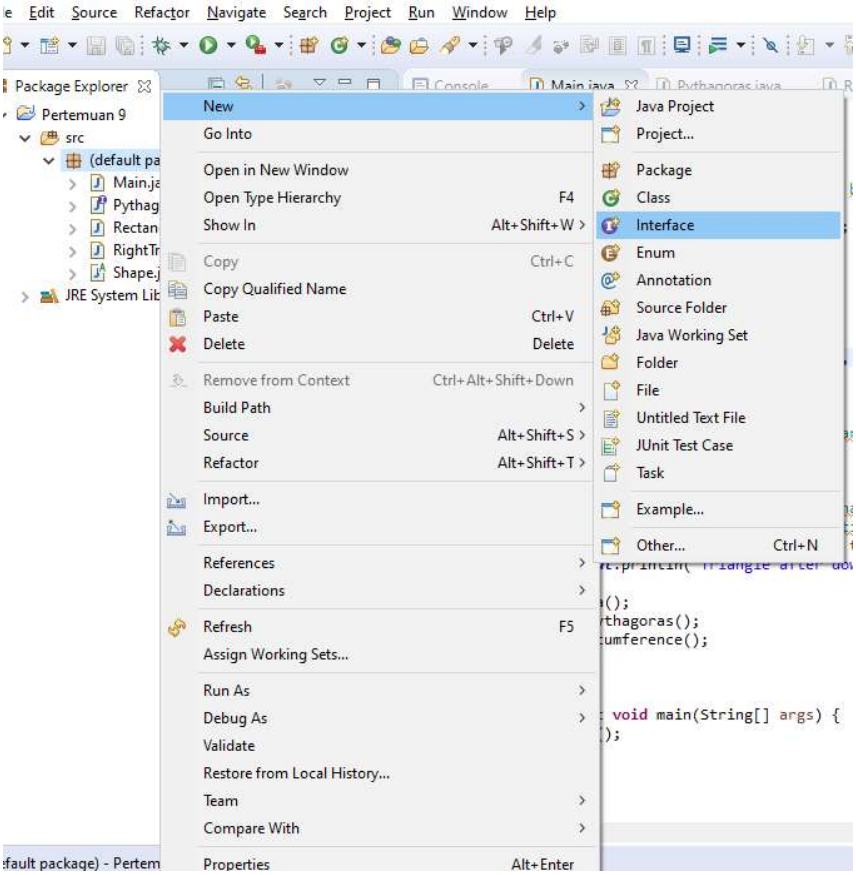


Interface mengandung method yang tidak memiliki body (mirip dengan abstract class). Abstract class dengan interface memiliki sifat yang sama, tapi terdapat perbedaan diantara keduanya.

No	Abstract Class	Interface
1	Dapat membuat property atau variable	Hanya dapat membuat konstanta
2	Tipe variabelnya bias punya final, non-final, static	Hanya punya static dan final variabel
3	Dapat mengimplementasikan kode method seperti class biasa	Harus menggunakan default
4	Dapat memiliki member private dan protected	Harus public semua
5	Diimplementasikan dengan pewarisan "*extends*" keyword	Menggunakan "*implements*" keyword
6	Dapat meng-extends class lain pada java, dan dapat meng-implementasi multiple java interfaces	Interface hanya dapat meng-extends interface lainnya

Interfaces Example

Cara membuat interface, klik kanan pada default package kemudian pilih new, dan pilih interface.



```java
public interface TrackerBehavior {
	//method di interface secara default public final
	//attribute di interface secara default public static final

	public void bite();
	public void sniffing();

	/*
	 * di java ada interface ini ada karena adanya diamond problem
	 * (baca lagi tentang diamond problem)
	 * interface ini berupa sifat
	 * jadi jika diperlukan bisa di implements ke class
	 * dan jika tidak diperlukan tidak perlu di implements
	 */
}


public class goldenRetriever implements TrackerBehavior(){

	@Override
	    public void bite()
	    {
	        System.out.println("The dog bite you!");
	        System.out.println("        __      _");
	        System.out.println("        \.'---.//|");
	        System.out.println("         |\./|  \/");
	        System.out.println("        _|.|.|_  \");
	        System.out.println("       /(  ) ' '  \");
	        System.out.println("      |  \\/   . |  \");
	        System.out.println("jrjc   \_/\__/| |");
	        System.out.println("        V  /V / |");
	        System.out.println("          /__/ /");
	        System.out.println("          /__/ /");

	    }

	    @Override
	    public void sniffing()
	    {
	       System.out.println("The dog sniffing");
	       System.out.println("                 _,)");
	       System.out.println("        _...._.-;-'");
	       System.out.println("     .-'      `(");
	       System.out.println("    /      ;   \\");
	       System.out.println("   ;.' ;`  ,;  ;");
	       System.out.println("  .'' ``. (  \ ;");
	       System.out.println(" / f_ _L \ ;  )\");
	       System.out.println(" \/|` '|\/;; <;/");
	       System.out.println("((; \_/  (() ");
	       System.out.println("     \"         \");


	    }
}
```