PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

PERTEMUAN KE-5

ABSTRACT CLASS DAN INTERFACE

TUJUAN PRAKTIKUM

Setelah mengikuti praktikum ini, mahasiswa diharapkan mampu memahami dan menerapkan konsep abstract class dan interface.

TOOLS

Tools yang diperlukan untuk melakukan praktikum ini adalah Java Development Kit (JDK) untuk mengkompilasi dan menjaalankan program Java serta code editor atau IDE untuk menulis program Java.

LANDASAN TEORI

Kelas abstrak merupakan kelas yang memiliki metode abstrak, yaitu metode tanpa implementasi. Hal tersebut dilakukan untuk memfasilitasi kelas yang lain yang memiliki implementasi yang berbeda-beda. Contoh implementasi kelas abstrak dalam pemrograman Java dari notasi algoritma adalah sebagai berikut :

Notasi Algoritma	Program Java

abstract class BangunDatar
{atribut}
warna: string, protected
border: string, protected
 	
{method}
abstract function getLuas() → real
abstract function getKeliling() → real
 	
{end class BangunDatar}
	abstract class BangunDatar {
	{atribut}
	protected String warna;
	protected String border;
	
	{method}
	public **abstract** double getLuas();
	public **abstract** double getKeliling();
	
	} //end class BangunDatar

Interface adalah kontrak yang harus dijalankan oleh sebuah kelas ketika kelas tersebut membuat implementasi sebuah interface. Contoh implementasi interface dalam pemrograman Java dari notasi algoritma adalah sebagai berikut:

Notasi Algoritma	Program Java

interface IResize
procedure zoomIn()
procedure zoomOut()
{end interface IResize}
 	
class Persegi **realize** IResize()
procedure zoomIn()
sisi ← sisi * 1.1
 	
procedure zoomOut()
sisi ← sisi * 0.9
 	
{end class Persegi}
	Public interface IResize{
	public void zoomIn();
	public void zoomOut();
	} //end interface IResize
	
	class Persegi **implements** IResize(){
	public void zoomIn(){
	sisi = sisi * 1.1
	}
	public void zoomOut(){
	sisi = sisi * 0.9
	}
	} //end class Persegi

LANGKAH PRAKTIKUM

Bagian 1 – Abstract Class

1. Gunakan kembali file program BangunDatar, Persegi, dan Lingkaran yang telah dibuat pada praktikum ke-4. Simpan pada folder baru. Tuliskan nama file, deskripsi, pembuat, dan tanggal pada bagian awal setiap file Anda sebagai komentar.
2. Setiap objek bangun datar memiliki luas dan keliling, meskipun implementasi luas dan keliling bergantung pada bentuk spesifik bangun datar. Oleh karena itu, ubahlah class BangunDatar menjadi abstract class, lalu tambahkan abstract method getLuas() dan getKeliling().

```
public abstract class BangunDatar {
     …
     public abstract double getLuas();
     public abstract double getKeliling();
}
```

   Pada class Persegi dan Lingkaran (yang meng-extend class BangunDatar) pastikan abstract method pada class BangunDatar telah diimplementasi semua, sehingga Persegi dan Lingkaran dapat dijadikan sebagai non-abstract class.

3. Buatlah sebuah main class dan main method dalam kelas tersebut. Dalam main method tersebut coba buatlah contoh objek dari class BangunDatar, Persegi dan Lingkaran. Lakukan pula eksperimen dengan menggunakan reference type BangunDatar saat membuat objek Lingkaran maupun persegi seperti contoh berikut:

```
BangunDatar B1 = new BangunDatar();
BangunDatar P1 = new Persegi(10);
Persegi P2 = new Persegi(5);
BangunDatar L1 = new Lingkaran(7);
Lingkaran L1 = new Lingkaran (14)
```

   Dari pembuatan objek seperti contoh di atas. Adakah kode yang bermasalah pada saat dijalankan? Selanjutnya, cobalah panggil method-method yang telah dibuat dari objek Persegi dan Lingkaran tersebut.

4. Sebuah bangun datar dapat memiliki luas ataupun keliling yang sama dengan bangun datar lainnya, meskipun memiliki bentuk spesifik yang berbeda. Tambahkan method di class BangunDatar untuk mengecek apakah sebuah objek bangun datar memiliki luas yang sama dengan bangun datar lainnya, seperti contoh berikut:

```
public boolean isEqualLuas(BangunDatar X){
      return this.getLuas() == X.getLuas();
}
```

   Tambahkan pula method serupa untuk mengecek apakah sebuah objek bangun datar memiliki keliling yang sama dengan objek bangun datar lainnya.

5. Panggilah method yang telah dibuat pada Langkah nomor 4 pada main method dari objek-objek yang telah dibuat pada langkah nomor 3.

   Apakah method yang dibuat pada langkah nomor 4 dapat digunakan untuk membandingkan objek bangun datar yang berbeda?

   Jika BangunDatar tidak dijadikan sebagai abstract class, dapatkah Anda membuat method isEqualLuas() dan isEqualKeliling pada class BangunDatar? Mengapa?

   Apakah kelebihan saat class BangunDatar dijadikan sebagai abstract class daripada non-abstract class?

Bagian 2 – Interface

1. Buatkah interface IResize yang menunjukkan sifat bahwa sebuah objek dapat di-resize, seperti contoh berikut:

```
11    public interface IResize {
12        //menambah ukuran menjadi 10% lebih besar
13        public void zoomIn();
14
15        //mengurangi ukuran menjadi 10% lebih kecil
16        public void zoomOut();
17
18        //menskalakan ukuran sesuai dengan input percent yang diberikan
19        public void zoom(int percent);
20    }
```

2. Jadikan class Persegi dan Lingkaran mengimplementasi interface IResize seperti contoh berikut:
   Pada header class Persegi tambahkan implement IResize:

```
11    public class Persegi extends BangunDatar implements IResize {
```

   Lalu, implementasikan semua method yang ada pada interface IResize di class Persegi

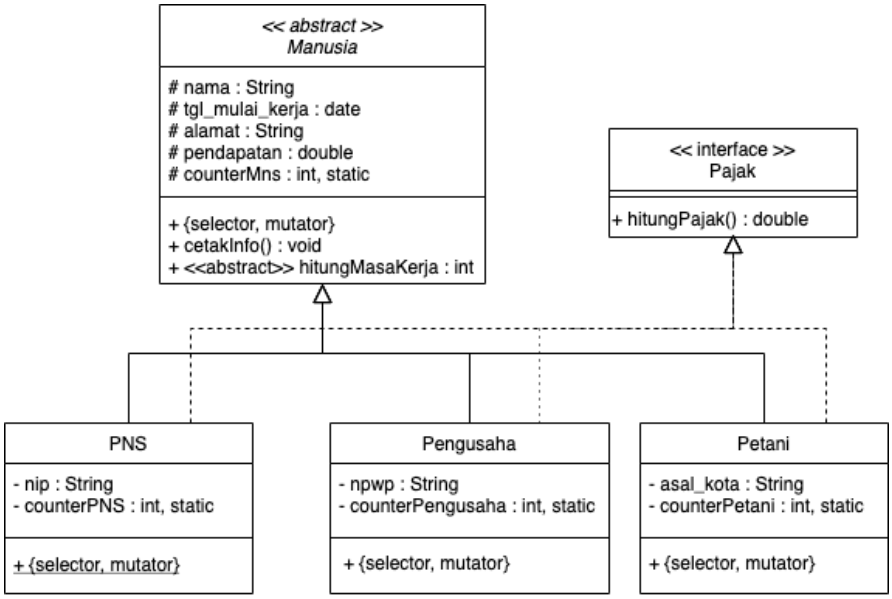
```
52        @Override
53        public void zoomIn(){
54            sisi = sisi * 1.1;
55        }
56
57        @Override
58        public void zoomOut(){
59            sisi = sisi * 0.9;
60        }
61
62        @Override
63        public void zoom(int percent){
64            sisi = sisi * percent/100;
65        }
```

3. Lakukan hal yang sama pada class Lingkaran.
4. Lakukan eksperimen pemanggilan method yang telah dibuat dari IResize pada main method. Bagaimana hasilnya?

Perhatikan bahwa sebuah interface tidak hanya dapat diimplementasi oleh class-class yang sejenis, berbeda dengan hubungan pewarisan class yang umumnya superclass memiliki sejumlah subclass yang sejenis. Interface IResize ini dapat direalisasi oleh kelas lain memiliki behaviour serupa, seperti class Garis, Bangun3D, dan seterusnya.
Apakah keuntungan saat method zoomIn(), zoomOut(), dan zoom() dikemas dalam interface IResize dibanding dijadikan sebagai abstract method dalam class BangunDatar?

LATIHAN

Pada kasus pelaporan pajak tahunan, seorang programmer memetakan sebagian perhitungan pajak ke dalam sebuah *class diagram* berikut :



Kamus Rumus:

Kamus	**hitungMasaKerja**	**hitungPajak**
PNS	= (now – tgl_mulai_kerja) + A	= 10% * pendapatan
Pengusaha	= (now – tgl_mulai_kerja) + B	= 15% * pendapatan
Petani	= (now – tgl_mulai_kerja) + C	= 0

*****Keterangan**:** A = digit ke 14 nim anda, B = digit ke 13 nim anda, dan C = digit ke 12 nim anda

Lakukan instruksi berikut ini:

a. **Implementasikan dalam bahasa java** diagram kelas diatas sesuai ketentuan yang diberikan.
b. Implementasi dari cetakInfo() mencetak semua atribut yang dimiliki oleh suatu kelas (baik atributnya sendiri maupun atribut dari pewarisan).
c. **Lengkapi segala sesuatunya** sehingga main program berikut dapat berjalan:

```
public class MManusia {
  public static void main(String[] args) {
    PNS p1 = new PNS('Satriyo',01-04-2006,'Jl. Seroja',15000000,'198302032006041002');
    Pengusaha pe1 = new Pengusaha('Adhy',01-01-2000,'Jl.Air',55000000,'000-556-773-212-000-5');
    Petani pt1 = new Petani('Nugraha',09-01-1977,'Jl. Bunga 9 Tembalang',5000000,'wonogiri');
    PNS p2 = new PNS('Panji',01-04-2010,10000000,'1980042120100041002');

    p2.setAlamat('Jl. Panorama 111 Tembalang');

    System.out.println("Jumlah Manusia = " + Manusia.getCounterMns());
    System.out.println("Jumlah PNS = " + PNS.getCounterPNS());
    System.out.println("Jumlah Pengusaha = " + Pengusaha.getCounterPengusaha());
    System.out.println("Jumlah Petani = " + Petani.getCounterPetani());

    System.out.println("Pajak PNS p1 = " + p1.hitungPajak());
```

```
    System.out.println("Pajak Pengusaha pe1 = " + pe1.hitungPajak());
    System.out.println("Pajak Petani pt1 = " + pt1.hitungPajak());

    System.out.println("Masa Kerja p1 = " + p1.hitungMasaKerja());
    System.out.println("Masa Kerja pe1 = " + pe1.hitungMasaKerja());
    System.out.println("Masa Kerja pt1 = " + pt1.hitungMasaKerja());

    p1.cetakInfo();
    pe1.cetakInfo();
    pt1.cetakInfo();
  }
}
```

PELAPORAN

Selama sesi praktikum, laporkan hasil praktikum pada link http://tiny.cc/pbo25.
Lengkapi semua file program yang harus dikerjakan dalam modul ini dan kumpulkan hasil akhirnya di kulon maksimal H+3 setelah pelaksanaan praktikum.

**********Selamat Mengerjakan dan Berlatih **********