Form Diskusi Pertemuan
POLIMORFISME

Tujuan diskusi:

- Mahasiswa mampu menerapkan polimorfisme statis
- Mahasiswa mampu menerapkan polimorfisme dinamis dengan menggunakan kelas dan interface.
- Mahasiswa mampu membuat kelas abstrak.

Diskusi 01: Polimorfisme Statik

Dengan menggunakan Netbeans buatlah:

1. Buat sebuah kelas **Bentuk2D** yang memiliki 3 buah fungsi, yang bernama sama yaitu **luas**,yang masing-masing digunakan untuk menghitung luas dari bentuk dua dimensi yang berbeda-bedaya itu **PersegiPanjang**, **BujurSangkar**, dan **Segitiga**.

```
public class Bentuk2D {
public double luas (intpanjang, intlebar) {
        return panjang*lebar;
    }

    public double luas (double sisi) {
        return sisi*sisi;
    }

public double luas (double alas, double tinggi) {
        return (alas*tinggi)/2;
    }
}
```

2. Buat kelas Uji (main program) dengan nama **Polimorfisme**. Isi kelas tersebut adalah:

```
public class Polimorfisme {
    public static void main(String[] args) {
        Bentuk2D b2d = new Bentuk2D();

System.out.println("LuasPersegiPanjang : " + b2d.luas(4,3));
System.out.println("LuasBujurSangkar : " + b2d.luas(4.0));
System.out.println("LuasSegitiga : " + b2d.luas(2.0, 5.0));
    }
}
```

3. Jelaskan hasil dari masing-masing eksekusi `b2d.luas()` dengan kata-kata Anda sendiri. Mengapa bisa menampilkan hasil yang berbeda?

Karena fungsi luas akann mendeteksi argument sesuai tipe data fungsi itu sendiri, kita bias melihat bahwa hasil diatas memanggil fungsi luas dengan argument yang berbeda. Oleh karena itu, hasilnya akan menyesuaikan dengan fungsi yang tipe data argumenya sesuai.

4. Mengapa ketiga fungsi dengan nama sama tersebut bias diterapkan di dalam sebuah kelas?Jelaskan menggunakan kata-kata Anda sendiri.

Karena pada fungsi yang sama tersebut terdapat parameter yang berbeda, walaupun fungsinya sama tetapi jika jika dipanggil berulang dengan struktur argument yang berbeda maka akan berbeda pula output yang dimunculkan.

Diskusi 02: Polimorfisme Dinamis dengan Kelas Konkret

Kelas Bentuk2D pada Diskusi 01 bukan merupakan perancangan kelas yang baik, maka akan kita ubah sehingga menerapkan konsep reusability, yaitu dengan pewarisan. Dengan menggunakan Netbeans buatlah:

1. Modifikasi kelas Bentuk2D yang memiliki 3 buah fungsi yang memiliki visibilitas public, yaitu:
   - **`Keliling`** dan **`luas`**, yang masing-masing digunakan untuk menghitung keliling dan luas dari bentuk dua dimensi (di sini dilakukan return 0),
   - **`display`**, yang digunakan untuk menampilkan luas dan keliling dengan format keluaran di bawah ini.

```
Luas     : 400.50
Keliling : 200.25
```

2. Buatlah 3 kelas yang ada dalam hirarki pewarisan pada gambar 1, yaitu kelas **`PersegiPanjang`**, **`Bujursangkar`**, **`Segitiga`**. Atribut, tipedata atribut, beserta dengan fungsi dan visibilitasnya yang ada pada masing-masing kelas dapat dilihat pada gambar 1 tersebut. Jangan lupa buat konstruktor untuk masing-masing kelas.

   - **`display`**, yang digunakan untuk menampilkan luas dan keliling dengan format keluaran di bawah ini.

```
LuasdanKeliling [NamaBangun] :
 Luas     : 400.50
Keliling : 200.25
```

3. Modifikasi kelas Uji, dengan nama Polimorfisme. Isi kelas tersebut adalah:

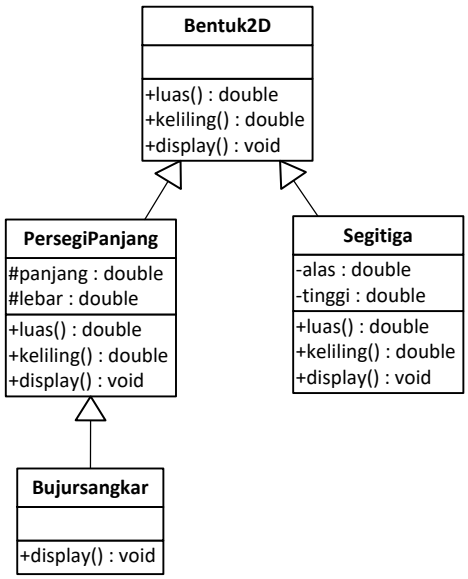
```
public class Polimorfisme{
    public static void main(String[] args) {
Bentuk2D b2d;

PersegiPanjangpp = new PersegiPanjang(4,5);
b2d = pp;
b2d.display();

Segitiga s3 = new Segitiga(6, 8);
```

```
b2d = s3;
        b2d.display();

b2d = new Bujursangkar(10);
b2d.display();
    }
}
```



Gambar 1. HirarkiPewarisan Bentuk2D

4. Jelaskan hasil dari masing-masing eksekusi **`b2d.display()`** dengan kata-kata Anda sendiri. Mengapa bisa menampilkan hasil yang berbeda? Apakah perbedaan polimorfisme pada diskusi 01 dan diskusi 02?

Diskusi 03: Polimorfisme Dinamis dengan Kelas Abstrak

1. Kelas **`Bentuk2D`** pada diskusi 02 mendefinisikan fungsi **`luas`** dan **`keliling`** dengan mengembalikan bilangan nol (return 0.0). Fungsi-fungsi ini diwariskan ke kelas turunan, dan kelas turunan mendapatkan fungsi warisan yang kurang sesuai, sehingga kelas turunan melakukan redefinisi (override) terhadap fungsi-fungsi tersebut. Tentu mengembalikan suatu angka nol untuk mencari luas dan keliling bukanlah suatu langkah yang elegan.

2. Satucara yang lebih baik adalah menyerahkan kepada kelas turunan untuk mendefinisikan

fungsi yang sesuai. Dengan demikian, kelas induk (**Bentuk2D**) tidak perlu membuat definisi fungsi 'return 0.0', melainkan cukup dengan mendeklarasikan fungsi luas dan keliling tanpa perlu membuat definisinya (definisi = membuat badan fungsi). Mendeklarasikan dapat diwujudkan dengan membuat fungsi abstrak, dengan deklarasi berikut ini pada kelas **Bentuk2D**:

```
public abstract double luas();
public abstract double keliling();
```

3. Apabila sebuah kelas memiliki setidaknya satu fungsi abstrak, maka kelas tersebut berubah menjadi kelas abstract. Dengan demikian modifikasilah kelas **Bentuk2D** menjadi kelas abstrak.Deklarasi kelas**Bentuk2D**menjadi :

```
public abstract class Bentuk2D {
public abstract double luas();

    public abstract double keliling();

    public void display() {
System.out.println("Luas     : " + luas());
System.out.println("Keliling : " + keliling() + "\n");
    }
}
```

4. Lakukan eksekusi kembali dan perhatikan hasilnya. Apa perbedaan polimorfisme dinamis dengan kelas konkret dan dengan kelas abstract? Jelaskan menggunakan kata-kata Anda sendiri.

- kelas konkret objejk yang di definisikan sudah pasti atau nyata, dan kita sudah tahu method yang akan dipanggil akan di implementasikan seperti apa.
- Sedangkan, kelas abstrak adalah kelas yang tidak ada isinya(tidak ada arahnya), kelas abstrak mendefinisikan pada kelas induknya, yang bertujuan ketika methodnya memiliki implementasi yang berbeda, maka kelas abstrak akan menjadi bentuk konkrit ketika mengakses kelas induknya

Demo 04: Polimorfisme Dinamis dengan Interface

1. Kelas **Bentuk2D** pada diskusi 03 mendeklarasikan fungsi **luas** dan **keliling** saja, dan menyerahkan kepada kelas turunan untuk mendefinisikan badan fungsi yang sesuai. Kelas turunan melakukan redefinisi (override) terhadap fungsi-fungsi tersebut. Pada diskusi 03, terdapat dua fungsi abstrak (**luas** dan **keliling**)dan sebuah fungsi konkrit (**display**).

2. Apabila semua fungsi yang ada dalam kelas **Bentuk2D** diubah menjadi fungsi abstrak, maka kelas **Bentuk2D** akan berubah menjadi

```
public abstract double luas();
public abstract double keliling();
public abstract void display();
```

3. Apabila keadaan kelas **Bentuk2D** pada langkah2 di atas terjadi, maka kelas abstrak **Bentuk2D** dapat diubah dengan menggunakan interface. Buatlah sebuah interface **IBentuk2D** dengan deklarasi berikut (mengunakan New→Java Interface):

```
public interface IBentuk2D {
    public double luas();
    public double keliling();
    public void display();
}
```

4. Modifikasilah Kelas **PersegiPanjang** dan **Segitiga** supaya mengimplementasikan interface **IBentuk2D** tersebut. Contoh:

```
public class PersegiPanjang implements IBentuk2D {
      // …… berisi sama dengan diskusi 02 &03
// modifikasi method display
        }
    }

    public class Segitiga implements IBentuk2D {
              // …… berisi sama dengan diskusi 02 &03
// modifikasi method display
        }
    }
```

5. Modifikasi kelas Uji, dengan nama Polimorfisme. Isi kelas tersebut adalah:

```
public class Polimorfisme {
        public static void main(String[] args) {
IBentuk2D b2d;

PersegiPanjangpp = new PersegiPanjang(4,5);
            b2d = pp;
b2d.display();

Segitiga s3 = new Segitiga(6, 8);
            b2d = s3;
b2d.display();
```

```
        b2d = new Bujursangkar(10);
        b2d.display();
                }
        }
```

6. Lakukan eksekusi kembali dan perhatikan hasilnya. Apa perbedaan polimorfisme dinamis dengan kelas abstrak dan dengan interface? Jelaskan menggunakan kata-kata Anda sendiri.

- **Abstrak**, ada satu atau lebih method yang bersifat abstrak, walaupun method abstrak tidak ada isinya(kosong), namun kelas abstrak dapat mengimplementasikan method tersebut.
- **Interface**, memiliki kemampuan yang sama dengan abstrak, *interface* memungkinkan kita mengimplementasikan method yang sama terhadap class yang tidak ada hubungan sama sekali (tidak dalam satu hirarki)

No Kelompok : 8
Anggota :

NIM Anggota	Nama Anggota
20102165	Satya prakash
20102192	Khusnul Khatimah
20102196	Tiara Humaira Putri
20102205	Yoga Naden

*Anggota yang tidak berkontribusi sama sekali dalam diskusi boleh tidak dituliskan.

Cukup 1 orang yang mengumpulkan

Berinama file :
X_Y_Polimorfisme.docx

X = Kelas
Y = Nomor Kelompok