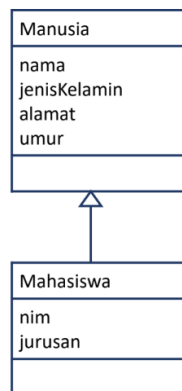


Nama : Faiza Muhammad Julianto
NIM : 312110009
Kelas : TI.21.C.2
Mata Kuliah : Pemograman Orientasi Objek

UAS PEMOGRAMAN ORIENTASI OBJEK

1. Inheritance

- Rancangan Class Diagram



- Code

🚦 *Mahasiswa.java*

```
public class Mahasiswa extends Manusia {
    String nim;
    String jurusan;

    public void nim(String nim) {
        this.nim = nim;
    }

    public String getNim() {
        return nim;
    }

    public void jurusan(String jurusan) {
        this.jurusan = jurusan;
    }

    public String getJurusan() {
        return jurusan;
    }

    public void cetakInfo() {
        super.getClass();
        System.out.println("Nama\t\t:" + nama);
        System.out.println("NIM\t\t: " + nim);
        System.out.println("Jurusan\t\t: " + jurusan);
    }
}
```

Manusia.java

```
public class Manusia {  
    String nama;  
    String jenisKelamin;  
    int umur;  
    String alamat;  
  
    public void cetakInfo() {  
        System.out.println("Nama\t\t: " + this.nama);  
        System.out.println("Jenis Kelamin\t: " + this.jenisKelamin);  
        System.out.println("Umur\t\t: " + this.umur);  
        System.out.println("Alamat\t\t: " + this.alamat);  
    }  
}
```

Main.java

```
public class Main {  
    public static void main(String[] args) {  
        Mahasiswa faiza = new Mahasiswa();  
  
        faiza.nim = "312110009";  
        faiza.nama= "Faiza Muhammad Julianto";  
        faiza.jenisKelamin= "Laki-laki";  
        faiza.umur= 20;  
        String alamat = "Bekasi Kota";  
        faiza.jurusan = "Informatika";  
  
        faiza.cetakInfo();  
    }  
}
```

- Hasil

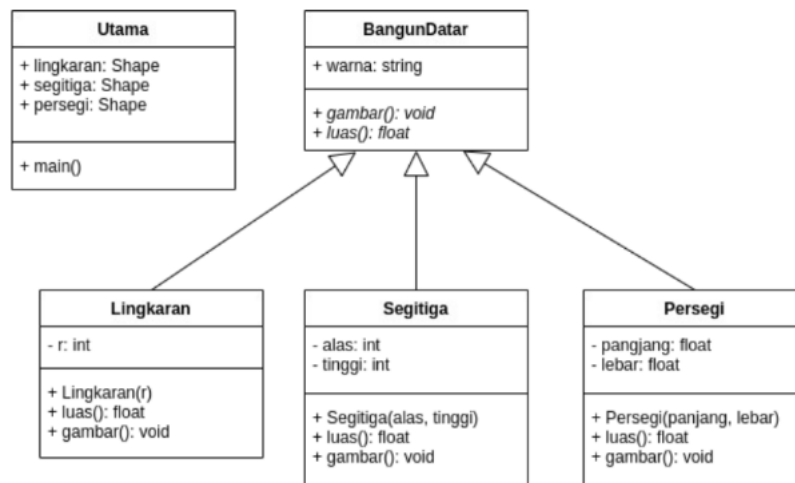
```
C:\Users\acer\.jdk\openjdk-19.0.1\bin\java.exe  
Nama      :Faiza Muhammad Julianto  
NIM       : 312110009  
Jurusan    : Informatika  
  
Process finished with exit code 0
```

- Fungsi

- ⇒ Tujuan dari penerapan inheritance ini adalah menggunakan kembali fungsi atau kode yang telah dibuat dan ditambah dengan atribut dan method baru.

2. Abstract Class atau Interface

- Rancangan Class Diagram



- Code

🚦 *Circle.java*

```
import java.lang.Math;
public class Circle extends Shape {
    private float radius;
    public Circle(float radius) {
        this.radius = radius;
    }
    @Override
    public void draw() {
        System.out.println("Drawing Circle");
    }
    @Override
    public float getAreas() {
        return (float) (Math.PI * radius * radius);
    }
}
```

🚦 *Rectangle.java*

```
public class Rectangle extends Shape {
    private float height;
    private float width;
    public Rectangle(float width, float height) {
        this.height = height;
        this.width = width;
    }
    @Override
    public void draw() {
        System.out.println("Drawing Rectangle");
    }
    @Override
    public float getAreas() {
        return this.width * this.height;
    }
}
```

```
}  
}
```

Triangle.java

```
public class Triangle extends Shape {  
    private float base;  
    private float height;  
    public Triangle(float base, float height) {  
        this.base = base;  
        this.height = height;  
    }  
    @Override  
    public void draw() {  
        System.out.println("Drawing Triangle");  
    }  
    @Override  
    public float getAreas() {  
        return 0.5f * base * height;  
    }  
}
```

Shape.java

```
public abstract class Shape {  
    String color;  
    public String getColor() {  
        return this.color;  
    }  
    public void setColor(String color) {  
        this.color = color;  
    }  
    public abstract void draw();  
    public abstract float getAreas();  
}
```

Utama.java

```
public class Utama {  
    public static void main(String[] args) {  
        // membuat objek Shape dari class Rectangle  
        Shape rect = new Rectangle(25, 40);  
        // membuat objek Shape dari class Circle  
        Shape circ = new Circle(21);  
        // membuat objek Shape dari class Triangle  
        Shape tria = new Triangle(14, 20);  
        /* memanggil method draw */  
        rect.draw();  
        circ.draw();  
        tria.draw();  
        System.out.println("Luas Lingkaran: " + circ.getAreas());  
        System.out.println("Luas Persegi Panjang: " + rect.getAreas());  
        System.out.println("Luas Segitiga: " + tria.getAreas());  
    }  
}
```

- Hasil

```
C:\Users\acer\.jdk\openjdk-19.0.1\bin\java.exe
Drawing Rectangle
Drawing Circle
Drawing Triangle
Luas Lingkaran: 1385.4424
Luas Persegi Panjang: 1000.0
Luas Segitiga: 140.0

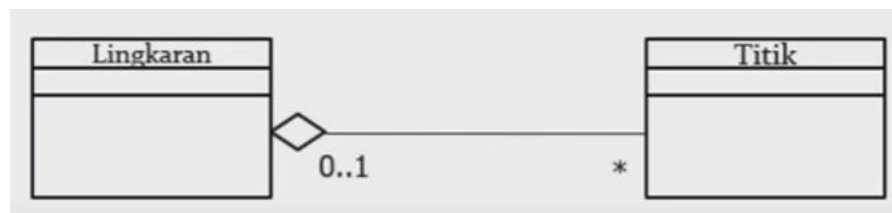
Process finished with exit code 0
```

- Fungsi


Fungsi *interface* jika dibandingkan dengan fungsi *abstract class*. Pada dasarnya *Interface* lebih berperan untuk *menyeragamkan method*. Interface tidak masuk kedalam struktur class seperti *abstract class*.

3. Class Relationship

- Rancangan Agregasi



- Code

 *Lingkaran.java*

```
public class Lingkaran
{
    private float jari;
    private Titik titikPusat;
    public Lingkaran (Titik T, float j)
    {
        jari=j;
        titikPusat=T;
    }

    public void cetakLingkaran()
    {
        System.out.println("Posisi Titik :");
        titikPusat.posisiTitik();
        System.out.println("Jari jari : " + jari + "\n" );
    }
}
```

Titik.java

```
public class Titik {  
    private float X,Y;  
    public Titik(float x, float y)  
    {  
        X=x;  
        Y=y;  
    }  
  
    public void posisiTitik()  
    {  
        System.out.println "[" + X + "," + "Y" + "]" ;  
    }  
}
```

Main.java

```
public class Main {  
    public static void main(String[] args)  
    {  
        Titik T = new Titik(3, 5);  
        Lingkaran L = new Lingkaran(T, 8);  
        L.cetakLingkaran();  
    }  
}
```

- Hasil

```
C:\Users\acer\.jdk\openjdk-19.0.1\bin\java.exe  
Posisi Titik :  
[3.0,Y]  
Jari jari :8.0  
  
Process finished with exit code 0
```


- Fungsi

Dalam hubungan asosiasi antara 2 objek, terdapat 2 relasi yaitu:

- ⇒ *Aggregation*, hubungan yang tidak terikat, hanya sekedar memiliki. Setiap class yang berhubungan dapat berdiri sendiri jika salah satunya dihancurkan atau tidak ada.
- ⇒ *Composition*, hubungan yang saling terikat, yang saling memiliki satu sama lain. Apabila salah satunya dihancurkan, maka class yang terikat lainnya akan hancur juga, atau tidak bisa digunakan lagi.

4. GUI

- Code

 *Show.java*

```
import javax.swing.*;

class Show {

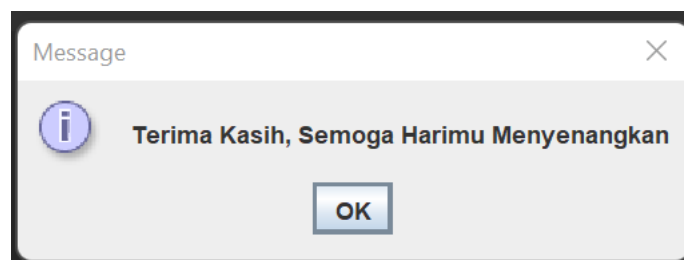
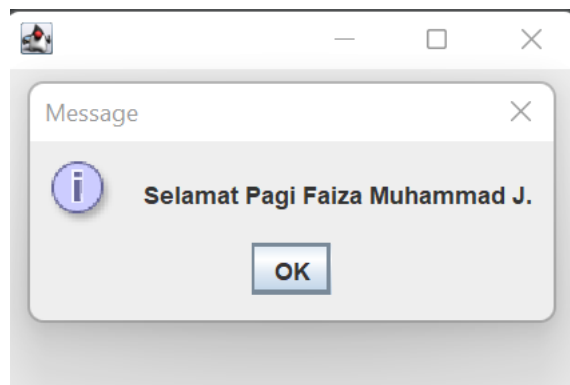
    public static void main(String[] args) {

        JFrame jFrame;

        jFrame = new JFrame();
        jFrame.setSize(300,200);
        jFrame.setVisible(true);

        JOptionPane.showMessageDialog(jFrame, "Selamat Pagi Faiza
Muhammad J.");
        JOptionPane.showMessageDialog(null, "Terima Kasih, Semoga
Harimu Menyenangkan");
    }
}
```

- Hasil



- Fungsi

⇒ Penggunaan dari GUI ini diimplementasikan pada perangkat elektronik seperti komputer, laptop dan juga smartphone. GUI biasanya digunakan untuk representasi visual untuk melakukan perintah dan fungsi pada operating system Anda juga pada software.