Modul Praktikum

Pemrograman Berorientasi Objek

Jurusan Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Lampung

2022
Semester Ganjil

Modul 11. Abstract Class dan Interface

Abstract class merupakan sebuah class yang tidak dapat diinisialisasi menjadi objek dan hanya berguna menjadi sebuah “Template” untuk class turunannya. Sedangkan interface merupakan sebuah abstract class, yang membedakannya dengan abstract class adalah interface harus diimplementasikan ke dalam sebuah class terlebih dahulu agar dapat digunakan baik method maupun atributnya.

Capaian Pembelajaran

1. Mahasiswa mampu mengimplementasikan konsep abstract class ke dalam sebuah program
2. Mahasiswa mampu mengimplementasikan konsep interface dalam program

Materi

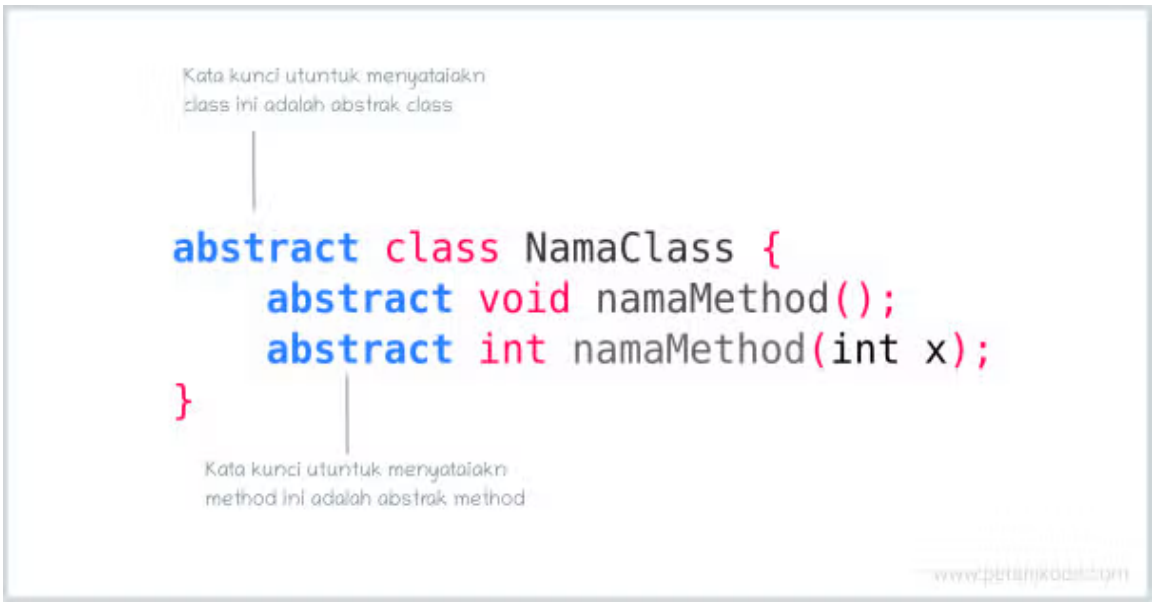
Abstract Class

Class abstrak adalah class yang masih dalam bentuk abstrak. Karena bentuknya masih abstrak, dia tidak bisa dibuat langsung menjadi objek. Sebuah class agar dapat disebut class abstrak setidaknya memiliki satu atau lebih **method abstrak**. **Method abstrak** adalah method yang tidak memiliki implementasi atau tidak ada bentuk konkritnya atau dalam kata lain **method abstract** adalah method yang tidak memiliki isi.

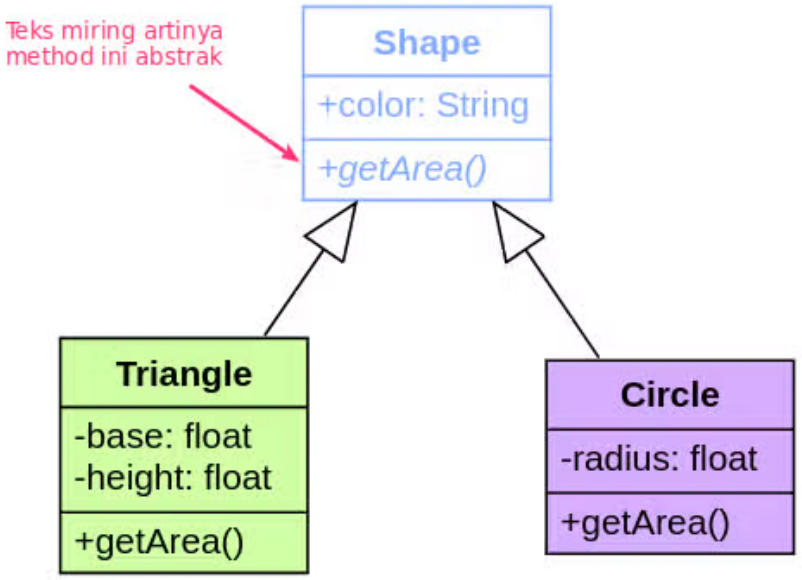
Deklarasi Abstract Class

Untuk dapat membuat sebuah Abstract Class maka dibutuhkan sebuah kata kunci yaitu **abstract** pada saat pembuatan sebuah Class. Begitupun dengan pembuatan **abstract method** yang membutuhkan sebuah keyword yaitu **abstract**. Namun perlu diingat, ketika ingin membuat sebuah **abstract method**, method tersebut tidak boleh memiliki body atau isi. Ilustrasi pendeklarasian **Abstract Class** dan **Abstract Method** dapat dilihat pada gambar dibawah ini.

Deklarasi Abstract Class :



Contoh Penggunaan Abstract Class



```
public abstract class Shape {

    String color;

    void setColor(String color){
        this.color = color;
    }

    String getColor(){
        return this.color;
    }

    abstract float getArea();
}
```

Pada gambar di atas, kita membuat sebuah Abstract Class bernama Shape. Di dalam class tersebut terdapat beberapa method biasa dan juga satu method abstract yaitu **getArea()**. Method inilah yang nantinya harus di override oleh anak dari class Shape ini.

```
public class Triangle extends Shape {

    private float base;
    private float height;

    public Triangle(int base, int height) {
        this.base = base;
        this.height = height;
    }


    @Override
    float getArea() {
        return 0.5f * base * height;
    }

}
```

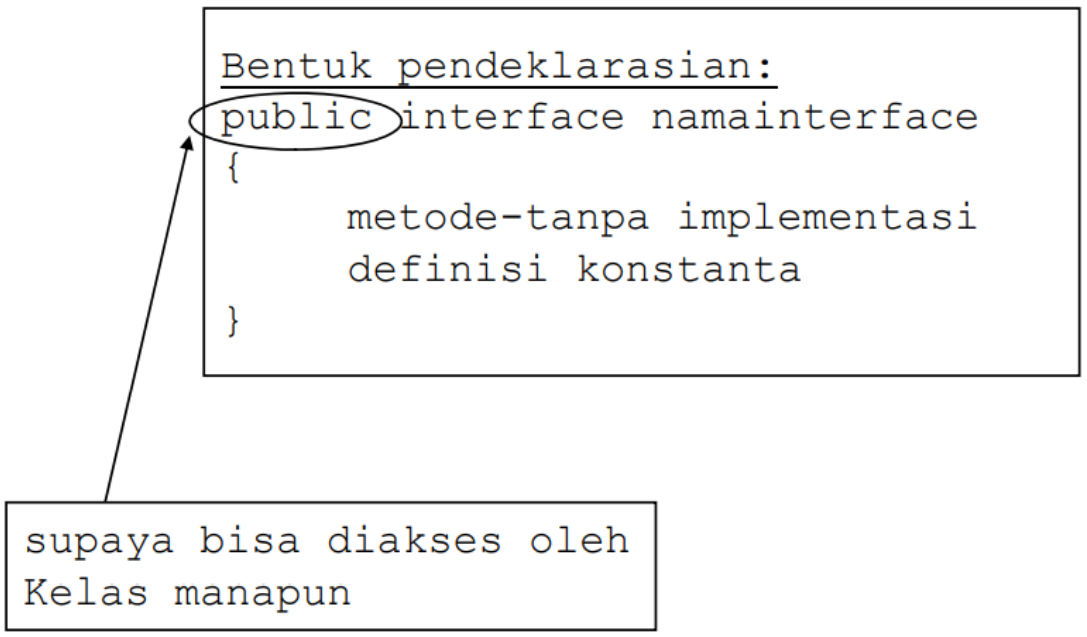
Cara untuk menggunakan sebuah Abstract Class pada Class tertentu, maka kita cukup menambahkan keyword **extends** seperti pada saat kita ingin membuat anak class pada bab Inheritance sebelumnya. **Class Triangle** di atas juga harus meng-override sebuah method abstract milik Class Shape yaitu **getArea()**. Namun bedanya, pada saat kita meng-override **method abstrak** tersebut, Kita bisa mengisi body dari method nya.

Class Interface

Pada dasarnya, Class Interface merupakan sebuah abstrak class. Namun bedanya didalam Class Interface hanya boleh berisi sebuah konstanta dan method.

Deklarasi Class Interface

Untuk membuat sebuah Class Interface, kita perlu menambahkan **keyword interface** sebelum nama class tersebut.



Untuk dapat menggunakan Class Interface pada Class lainnya, kita perlu menambahkan keyword **implements** yang mana ini berbeda dengan ketika kita ingin menggunakan sebuah Class Abstract yang perlu menggunakan keyword **extends**. Contoh penggunaan dari sebuah Class Interface pada Class lain dapat dilihat pada gambar dibawah ini.

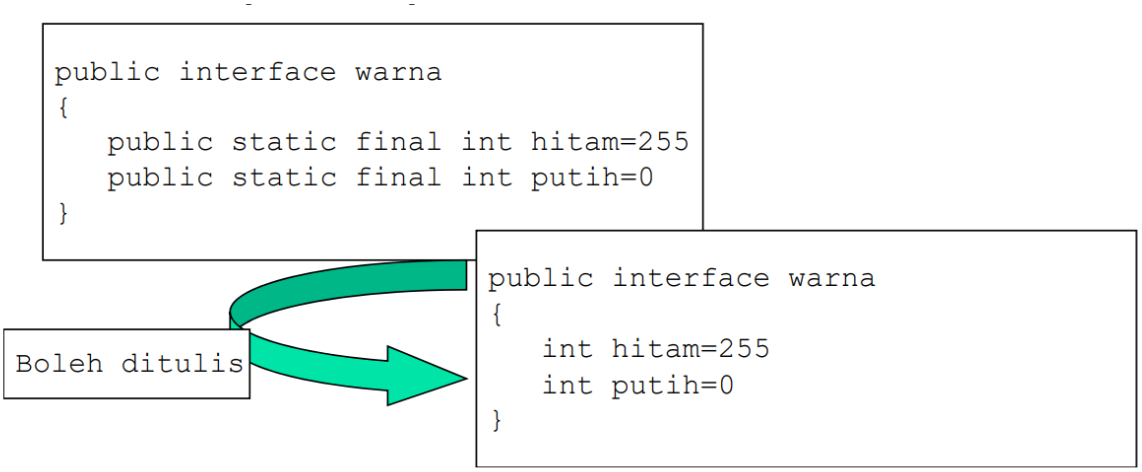
```java
public class Xiaomi implements Phone {

    private int volume;
    private boolean isPowerOn;

    public Xiaomi() {
        // set volume awal
        this.volume = 50;
    }
```

Variable Pada Class Interface

Variable di dalam Class Interface hanya diperkenankan untuk menggunakan akses modifier **public static final**. Seandainya jika akses modifier tersebut tidak ditulis, maka secara implisit akan diberlakukan akses modifier tersebut pada sebuah variabel.



Perbedaan Abstract Class dan Interface Class

Berikut adalah perbedaan antara Abstract Class dan Interface Class.

Abstract	Interface
Bisa berisi abstract dan non-abstract method.	Hanya boleh berisi abstract method.
Kita harus menuliskan sendiri modifiernya.	Kita tidak perlu menulis public abstract di depan nama method. Karena secara implisit, modifier untuk method di interface adalah public dan abstract.
Bisa mendeklarasikan constant dan instance variable.	Hanya bisa mendeklarasikan constant. Secara implisit variable yang dideklarasikan di interface bersifat public, static dan final.
Method boleh bersifat static.	Method tidak boleh bersifat static.
Method boleh bersifat final.	Method tidak boleh bersifat final.
Suatu abstact class hanya bisa meng-extend satu abstract class lainnya.	Suatu interface bisa meng-extend satu atau lebih interface lainnya.
Abstract class hanya bisa meng-extend satu abstract class dan meng-implement beberapa interface.	Suatu interface hanya bisa meng-extend interface lainnya. Dan tidak bisa meng-implement class atau interface lainnya.

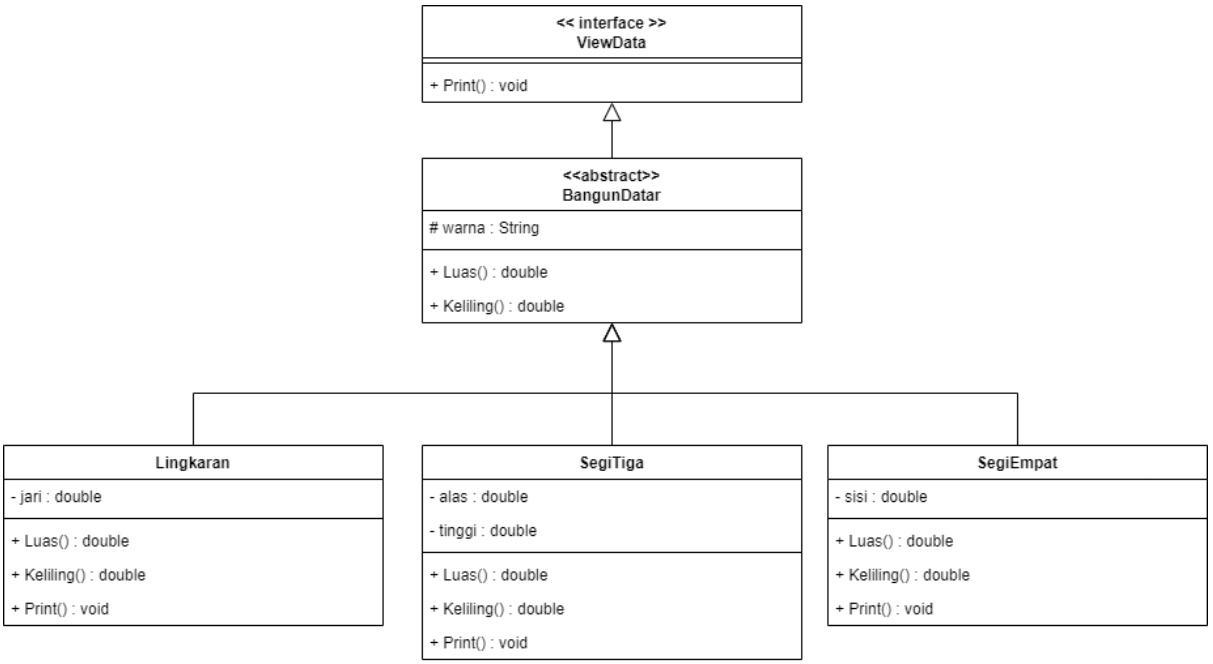
Langkah Praktikum

Membuat Project

1. Buka Netbeans
2. Buat project baru dengan nama **Praktikum11_nama**

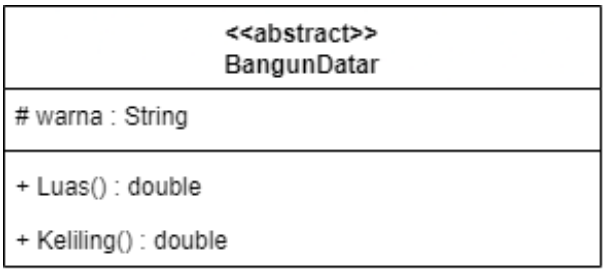
Studi Kasus

Anda akan diminta untuk membuat beberapa class yang saling terhubung menggunakan konsep abstract class dan interface. Class yang akan dibuat dapat dilihat pada class diagram berikut:



Membuat Abstract Class

1. Buatlah sebuah **abstract class** dengan nama **BangunDatar**. Sesuaikan isinya dengan class diagram berikut:



2. Pastikan untuk menambahkan kata kunci **abstract** sebelum pendeklarasian class dan method yang bersifat abstract.

3. Tidak perlu membuat Constructor, Setter, dan Getter karena class ini adalah class Abstract.
4. Keseluruhan code untuk class ini akan terlihat sebagai berikut:

```
public abstract class BangunDatar {
    protected String warna;

    public abstract double luas();

    public abstract double keliling();
}
```

Membuat Child Class

1. Buatlah class baru dengan nama **Lingkaran** yang merupakan turunan dari class **BangunDatar**. Sesuaikan isinya dengan class Diagram berikut:

Lingkaran
- jari : double
+ Luas() : double + Keliling() : double

2. Buatlah Constructor, Setter, dan Getter untuk class Lingkaran.
3. Karena merupakan class turunan dari sebuah abstract class, maka class Lingkaran harus melakukan override terhadap method yang abstract pada class parentnya.
4. Sesuaikan isi dari method luas dan keliling agar mengembalikan nilai luas dan keliling dari sebuah lingkaran berdasarkan atributnya.

5. Keseluruhan code untuk class ini akan tampak seperti berikut:

```
public class Lingkaran extends BangunDatar {
    private double jari;

    public Lingkaran(double jari, String warna) {
        this.jari = jari;
        this.warna = warna;
    }

    public double getJari() {
        return jari;
    }

    public void setJari(double jari) {
        this.jari = jari;
    }

    public String getWarna() {
        return this.warna;
    }

    public void setWarna(String warna) {
        this.warna = warna;
    }

    @Override
    public double keliling() {
        return 2 * Math.PI * jari;
    }

    @Override
    public double luas() {
        return Math.PI * jari * jari;
    }
}
```

6. Lakukan hal yang sama untuk 2 class turunan lainnya, sesuai class diagram berikut:

SegiTiga
- alas : double - tinggi : double
+ Luas() : double + Keliling() : double

SegiEmpat
- sisi : double
+ Luas() : double + Keliling() : double

Mengisi fungsi main

1. Ubah isi dari fungsi main menjadi seperti berikut :

```
public static void main(String[] args) {
    BangunDatar lingkaran = new Lingkaran(10, "Merah");
    BangunDatar segiTiga = new SegiTiga(13, 5, "Biru");
    BangunDatar segiEmpat = new SegiEmpat(5, "Hitam");

    BangunDatar[] bangunDatar = new BangunDatar[3];
    bangunDatar[0] = lingkaran;
    bangunDatar[1] = segiTiga;
    bangunDatar[2] = segiEmpat;

    for (int i = 0; i < 3; i++) {
        System.out.println("Bangun Datar " + (i + 1));
        System.out.println("Warna: " + bangunDatar[i].getWarna());
        System.out.println("Luas: " + bangunDatar[i].luas());
        System.out.println("Keliling: " + bangunDatar[i].keliling());
        System.out.println();
    }
}
```

2. Apabila dijalankan maka hasil outputnya akan menjadi seperti berikut:

Run:
Bangun Datar 1
Warna: Merah
Luas: 314.1592653589793
Keliling: 62.83185307179586

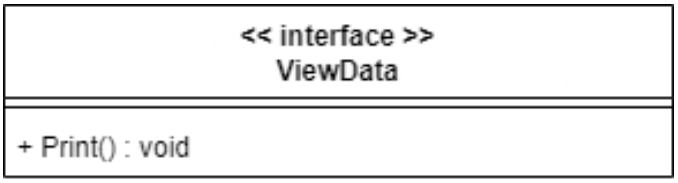
Bangun Datar 2
Warna: Biru
Luas: 32.5
Keliling: 39.0

Bangun Datar 3
Warna: Hitam
Luas: 25.0

Keliling: 20.0

Membuat Interface

1. Buatlah sebuah interface dengan nama **ViewData**.
2. Caranya sama dengan membuat class. Hanya saja kata kunci **class** diganti menjadi **interface**.
3. Buat sebuah method **Print()** dengan tipe data **void** di dalam interface tersebut, sesuai dengan class diagram berikut:



4. Keseluruhan code pada interface ViewData akan tampak sebagai berikut:

```
public interface ViewData {
    void print();
}
```

Mengimplementasikan Interface

1. Implementasikan interface **ViewData** ke dalam class **BangunDatar**. Anda dapat menggunakan kata kunci **implements**.
2. Code pada class bangun ruang akan tampak seperti ini:

```
public abstract class BangunDatar implements ViewData {
    protected String warna;

    public abstract double luas();

    public abstract double keliling();
}
```

Membuat Method Override

1. Selanjutnya buatlah method **Print()** pada masing-masing class **Lingkaran, SegiTiga,** dan **SegiEmpat**. Method ini akan meng-override method pada interface **ViewData**.

2. Isi dari method **Print()** akan menampilkan data dari setiap class dalam format sebagai berikut :

```
Lingkaran
Warna : Merah
Jari-jari : 10
Luas : 314
Keliling : 62.8
```

Mengubah isi fungsi main

1. Ubah isi dari fungsi main menjadi seperti berikut :

```java
public static void main(String[] args) {
    BangunDatar lingkaran = new Lingkaran(10, "Merah");
    BangunDatar segiTiga = new SegiTiga(13, 5, "Biru");
    BangunDatar segiEmpat = new SegiEmpat(5, "Hitam");

    BangunDatar[] bangunDatar = new BangunDatar[3];
    bangunDatar[0] = lingkaran;
    bangunDatar[1] = segiTiga;
    bangunDatar[2] = segiEmpat;

    for (int i = 0; i < 3; i++) {
        bangunDatar[i].print();
    }
}
```

2. Apabila dijalankan maka anda akan mendapatkan hasil seperti berikut:

```
Run:
Lingkaran
Warna : Merah
Jari-jari : 10.0
Luas : 314.1592653589793
Keliling : 62.83185307179586

Segi Tiga
Warna : Biru
Alas : 13.0
Tinggi : 5.0
Luas : 32.5
Keliling : 39.0

Segi Empat
Warna : Hitam
Sisi : 5.0
```

```
Luas : 25.0
Keliling : 20.0
```