

LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

PERTEMUAN KE-10



Disusun Oleh :

NAMA : Raden Isnawan Argi Aryasatya

NIM : 195410257

JURUSAN : Teknik Informatika

JENJANG : S1

KELAS : TI-5

Laboratorium Terpadu
Sekolah Tinggi Manajemen Informatika Komputer
AKAKOM
YOGYAKARTA

2021

PERTEMUAN KE-10 (INTERFACE)

TUJUAN

Dapat membuat dan menggunakan interface pada aplikasi

DASAR TEORI

Interface adalah kelas yang benar-benar abstrak, artinya hanya berisi deklarasi method dan (jika ada) konstanta saja. Method-method tersebut nantinya harus diimplementasikan pada real class. Interface dapat dianalogikan seperti menandatangani kontrak kerja. Misalnya seorang dosen wajib mengajar, membuat soal, dsb, akan tetapi cara mengajar dan membuat soalnya diserahkan ke masing-masing dosen (tidak ditentukan dalam kontrak kerja).

PRAKTIK 1: Interface Bangun2D

```
TestLingkaran.java ×
interface Bangun2D{
    public double hitungLuas();
    public double hitungKeliling();
}
class Lingkaran implements Bangun2D{
    private double jejari;
    public void setJejari(double jejari){
        this.jejari=jejari;
    }
    public double getJejari(){
        return this.jejari;
    }
    public double hitungLuas(){
        return (3.14 * this.jejari * this.jejari);
    }
    public double hitungKeliling(){
        return (2 * 3.14 * this.jejari);
    }
}

class TestLingkaran{
    public static void main(String[] arg){
        Lingkaran bunder = new Lingkaran();
        bunder.setJejari(10);
        double luas = bunder.hitungLuas();
        double keliling = bunder.hitungKeliling();
        System.out.println("Luas lingkaran dengan jejari "
            +bunder.getJejari()+ " adalah "+luas);
        System.out.println("Keliling lingkaran dengan jejari "
            +bunder.getJejari()+" adalah "+keliling);
    }
}
```

output:

```
C:\Windows\system32\cmd.exe
Luas lingkaran dengan jejari 10.0 adalah 314.0
Keliling lingkaran dengan jejari 10.0 adalah 62.800000000000004
Press any key to continue . . . _
```

penjelasan:

pertama-tama, untuk mendeklarasikan interface, kita perlu menulis sebuah keyword bernama "interface". Pada program di atas, kita mendeklarasikannya dengan menulis **interface Bangun2D**.

Setiap interface secara implisit sudah ditetapkan sebagai abstrak sehingga kita tadi tidak perlu menggunakan keyword abstract saat mendeklarasikan interface Bangun2D. Hal itu juga berlaku pada method-method di dalam interface, sehingga keyword abstract tidak diperlukan. Maka dari itu, method-method di dalam interface kita deklarasikan dengan menggunakan modifier public yaitu seperti baris-baris kode berikut:

```
public double hitungLuas();  
public double hitungKeliling();
```

Untuk menerapkan interface pada suatu class, kita menggunakan keyword "implements". Keyword tersebut harus berada di deklarasi class seperti berikut: **class Lingkaran implements Bangun2D**.

Di dalamnya kita buat variabel jejari dengan modifier private. Modifier private akan membuat member hanya bisa diakses oleh dari dalam class itu sendiri. Kemudian kita buat method mutator dengan parameter double jari dan sekaligus kita akses variabel jejari tersebut dengan this. Kita lakukan itu dengan menuliskan baris kode berikut:

```
public void setJejari(double jejari){  
    this.jejari=jejari;
```

kita buat method accessor milik jejari dengan mendeklarasikan method **public double getJejari()** yang memiliki return value **return this.jejari;**

setelah itu, kita terapkan kedua method milik interface Bangun2D di dalam class Lingkaran. Keduanya kita deklarasikan kemudian kita beri value nya masing-masing seperti berikut:

```
public double hitungLuas(){  
    return (3.14 * this.jejari * this.jejari);  
}  
public double hitungKeliling(){  
    return (2 * 3.14 * this.jejari);  
}
```

kita buat program main method nya bernama class TestLingkaran. Eksekusi nya kita awali dengan menuliskan **public static void main(String[] args){**.

Kemudian kita lakukan instansiasi (pembuatan objek) bernama bunder yang kita ambil dari class Lingkaran. Untuk melakukannya, kita menuliskan **Lingkaran bunder = new Lingkaran();**. Di dalam objek tersebut, kita set jari-jari lingkaran dengan **bunder.setJejari(10);** yang artinya jari-jari lingkaran adalah 10. Kemudian kita buat dua atribut yaitu double luas dan double keliling yang di dalamnya kita inisialisasi objek bunder dengan method hitungLuas() dan hitungKeliling(). Untuk melakukan itu kita perlu menuliskan baris kode berikut:

```
double luas = bunder.hitungLuas();  
double keliling = bunder.hitungKeliling();
```

terakhir, kita cetak output nya dengan menuliskan baris-baris kode berikut:

```
System.out.println("Luas lingkaran dengan jejari "  
    +bunder.getJejari()+ " adalah "+luas);  
System.out.println("Keliling lingkaran dengan jejari "  
    +bunder.getJejari()+ " adalah "+keliling);
```

fungsi dari bunder.getJejari adalah untuk mendapatkan value jari-jari lingkaran yang sudah di set tadi yaitu 10.

PRAKTIK 2: Kombinasi antara turunan satu kelas dan interface

```
TestInherInterface.java × TestLingkaran.java
1  interface MProvides{
2  void func();
3  }
4  interface MRequires{
5  int getValue();
6  }
7
8  class Mixin implements MProvides
9  {
10     private final MRequires parent;
11     public Mixin(MRequires parent) {
12         this.parent = parent;
13     }
14     public void func() {
15         System.out.println("Nilai dari method func: "
16                             + parent.getValue());
17     }
18 }
19
20 class Parent
21 {
22     private int value;
23     public Parent(int value ) {
24         this.value = value;
25     }
26     public int getValue() {
27         return this.value;
28     }
29 }
30
31 class Child extends Parent implements MRequires, MProvides{
32     private final MProvides mixin;
33     public Child(int value){
34
35         super(value);
36         this.mixin = new Mixin(this);
37     }
38     public void func(){
39         mixin.func();
40     }
41 }
42
43 class TestInherInterface{
44     public static void main(String[] arg){
45         Child anak = new Child(5);
46         anak.func();
47         System.out.println("nilai dari method getValue:"
48                             +anak.getValue());
49     }
50 }
```

output:

```
C:\Windows\system32\cmd.exe
Nilai dari method func: 5
nilai dari method getValue:5
Press any key to continue . . . _
```

penjelasan:

Pada program kali ini, intinya kita membuat dua interface yaitu MProvides (interface yang menyediakan suatu method) dan MRequires (interface yang memerlukan suatu method) lalu membuat class Mixin beserta class Parent yang akan mewarisi method dan atributnya kepada class Child.

Untuk membuat kedua interface, kita ketik baris-baris kode berikut:

```
interface MProvides{  
void func();  
}  
interface MRequires{  
int getValue();  
}
```

MProvides memiliki method func() yang tidak memiliki modifier, sedangkan MRequires sebuah method accessor bernama getValue().

Kemudian kita buat sebuah class bernama Mixin yang mengimplement interface MProvides. Di dalamnya ada pendeklarasian atribut bernama parent dengan **private final MRequires parent** lalu ada konstruktor yang memiliki sebuah parameter yaitu **public Mixin(MRequires parent)** dan di dalamnya kita akses variabel parent tersebut dengan menuliskan **this.parent = parent;**

di dalam class Mixin ini, kita deklarasikan kembali method func() milik interface MProvides, kemudian di dalamnya kita buat sebuah kalimat beserta inisialisasinya dengan atribut parent + method getValue. Kita lakukan itu dengan menuliskan baris kode ini:

```
public void func() {  
System.out.println("Nilai dari method func: "+ parent.getValue());
```

Kita buat class Parent yang di dalamnya ada variabel int value yang bermodifier private. Lalu kita buat konstruktor berparameter int value kemudian kita akses variabel value dengan cara **this.value = value**. Kemudian kita deklarasikan method getValue() yang di dalamnya kita buat return value nya **return this.value;**

Selanjutnya kita buat class Child yang diwarisi method dan atribut dari class Parent dan implement dari MRequires dan MProvided. Kita tulis **class Child extends Parent implements MRequires, MProvides**. Di dalamnya kita buat atribut mixin dan konstruktor Child berparameter int value dan sekaligus mengakses atribut value dengan super (karena dari class parent) kemudian mixin dengan this. Untuk melakukan itu kita tulis:

```
private final MProvides mixin;  
public Child(int value){  
    super(value);  
    this.mixin = new Mixin(this);  
}  
public void func(){  
    mixin.func();
```

Terakhir kita buat program main method bernama TestInherInterface yaitu dengan menuliskan **class TestInherInterface**. Di dalamnya kita melakukan instansiasi atau pembuatan objek bernama anak sekaligus memberikan value 5 dengan menuliskan baris kode **Child anak = new Child(5);**. Lalu kita akses func dengan objek anak dengan **anak.func()**, yang akan memunculkan kalimat yang tadi sudah ditulis di class Mixin. Terakhir, kita cetak satu kalimat untuk getValue yaitu:

```
System.out.println("nilai dari method getValue:"  
    +anak.getValue());
```

PRAKTIK 3: Multiple interface

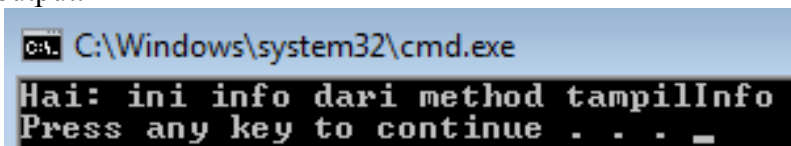
```
MultipleInterface.java x TestInherInterface.java TestLingkaran.java
interface Interface1 {
    public void tampilInfo();
    public void setInfo(String info);
}
    interface Interface2 {
        public void cetakInfo();
    }

class MultiInterfaces implements Interface1, Interface2 {
    private String info;
    public void setInfo(String info) {
        this.info = info;
    }
    public void tampilInfo(){
        System.out.println(this.info+": ini info dari method tampilInfo");
    }

    public void cetakInfo(){
        System.out.println(this.info+": ini info dari method cetakInfo");
    }
}

public class MultipleInterface{
    public static void main(String[] a) {
        MultiInterfaces t = new MultiInterfaces();
        t.setInfo("Hai");
        t.tampilInfo();
    }
}
```

output:



```
C:\Windows\system32\cmd.exe
Hai: ini info dari method tampilInfo
Press any key to continue . . . _
```

penjelasan:

pada program di atas, kita menerapkan multiple interface atau bisa dibilang ada beberapa interface di dalam suatu program. Khusus pada program tersebut kita mengimplementasikan multiple interface pada satu class saja. Java tidak memperkenankan multiple inheritance pada suatu class tapi mengizinkan implementasi multiple interface.

Pertama-tama, kita mencetak Interface1 yang berisi

```
public void tampilInfo();
public void setInfo(String info);
```

lalu kita buat Interface2 yang berisi:

```
interface Interface2 {
    public void cetakInfo();
```

kemudian kita buat satu-satunya class bernama MultiInterfaces yang implement dari Interface1 dan Interface2 yaitu dengan menuliskan **class MultiInterfaces implements Interface1, Interface2**.

Di dalamnya kita buat atribut dan konstruktor seperti berikut:

```
private String info;
public void setInfo(String info) {
    this.info = info;
```

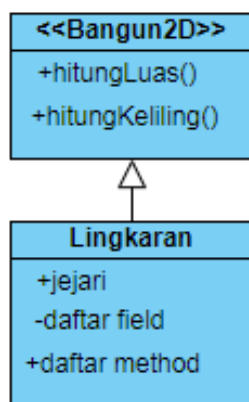
lalu kita deklarasikan kembali method tampilInfo() milik Interface1 dengan menuliskan **System.out.println(this.info+": ini info dari method tampilInfo");**

kita deklarasikan juga method cetakInfo() milik Interface2 dengan menuliskan **System.out.println(this.info+": ini info dari method cetakInfo");**

terakhir kita buat main method bernama MultipleInterface yang di dalamnya kita melakukan pembuatan objek bernama t dengan menuliskan **MultiInterfaces t = new MultiInterfaces.** Kemudian kita cetak output dengan **t.setInfo("Hai");** dan **t.tampilInfo();**

LATIHAN

1. Gambarkan class diagram untuk program pada praktek 1!



penjelasan:

untuk diagram di atas adalah interface Bangun2D yang memiliki method hitungLuas() dan hitungKeliling(). Kemudian diagram class dibawah adalah class Lingkaran yang mengimplementasi interface bangun2D. Di dalamnya kita tambah atribut jejari dan method-method milik Bangun2D kita deklarasikan lagi di dalamnya.

2. Buatlah program Java yang mengimplementasikan multiple interface seperti diagram kelas berikut,

```
TestBangun.java x TestInherInterface.java MultipleInterface.java
1 interface Bangun2D{
2     public double hitungLuas();
3     public double hitungKeliling();
4 }
5
6 interface Bangun3D{
7     public double hitungVolume();
8     public void tampilInfo();
9     public void setInfo(String info);
10 }
11
12 interface Pola{
13     public void Warna();
14     public void setWarna(String warna);
15 }
16
17 class Lingkaran implements Bangun2D{
18     private double jejari;
19     public void setJejari(double jejari){
20         this.jejari=jejari;
```

```

20         this.jejari=jejari;
21     }
22     public double getJejari(){
23         return this.jejari;
24     }
25     public double hitungLuas(){
26         return (3.14 * this.jejari * this.jejari);
27     }
28     public double hitungKeliling(){
29         return (2 * 3.14 * this.jejari);
30     }
31 }
32
33 class Tabung implements Bangun3D, Pola{
34     private double tinggi;
35     private double radius;
36     private String info;
37     private String warna;
38
39     public void setInfo(String info) {
40         this.info = info;
41     }
42     public void setWarna(String warna) {
43         this.warna = warna;
44     }
45
46     public void setTinggi(double tinggi){
47         this.tinggi = tinggi;
48     }
49     public void setRadius(double radius){
50         this.radius = radius;
51     }
52     public double getTinggi(){
53         return this.tinggi;
54     }
55     public double getRadius(){
56         return this.radius;
57     }
58     public double hitungVolume(){
59         return (3.14 * this.radius * this.radius * this.tinggi);
60     }
61
62     public void tampilInfo(){
63         System.out.println(this.info+": tabung keren ");
64     }
65     public void Warna(){
66         System.out.println("warna tabung adalah "+ this.warna);
67     }}
68
69 public class TestBangun{
70     public static void main(String[] args){
71         Lingkaran bunder = new Lingkaran();
72         bunder.setJejari(10);
73         double luas = bunder.hitungLuas();
74         double keliling = bunder.hitungKeliling();
75         System.out.println("Luas lingkaran dengan jejari "
76             +bunder.getJejari()+ " adalah "+luas);
77         System.out.println("Keliling lingkaran dengan jejari "
78             +bunder.getJejari()+ " adalah "+keliling);
79
80         Tabung tab = new Tabung();
81         tab.setRadius(12);
82         tab.setTinggi(24);
83         double volume = tab.hitungVolume();
84         System.out.println("Volume tabung dengan radius "+tab.getRadius() +
85             " adalah " + volume);
86         tab.setInfo("info tabung");
87         tab.setWarna("merah");
88         tab.tampilInfo();
89         tab.Warna();
90     }
91 }

```


output:

```
C:\Windows\system32\cmd.exe
Luas lingkaran dengan jejari 10.0 adalah 314.0
Keliling lingkaran dengan jejari 10.0 adalah 62.800000000000004
Volume tabung dengan radius 12.0 adalah 10851.84
info tabung: tabung keren
warna tabung adalah merah
Press any key to continue . . . _
```

penjelasan:

Jadi inti dari program ini adalah kita membuat dan meneruskan praktik 1 yang mempunyai class Lingkaran. Sekarang kita membuat class baru yaitu class Tabung yang implementasi dari interface Bangun3D dan Pola. isi Bangun 3D ada 3 method yaitu hitungVolume(); tampilInfo(); dan setInfo(String info);. isi Pola adalah method Warna() dan setWarna(). Pada class tabung kita deklarasi 4 atribut yaitu tinggi radius info dan warna. Kemudian kita proses dengan method accessor dan mutator. Sementara hitungVolume berisi rumus volume tabung dan tampilInfo serta Warna berisi kalimat yang dikehendaki. Terakhir kita buat main method. Untuk class Lingkaran sama saja seperti praktik 1. untuk Tabung, kita setRadius 12 dan setTinggi 24 dan kita munculkan beberapa output seperti berikut:

```
tab.setInfo("info tabung");
tab.setWarna("merah");
tab.tampilInfo();
tab.Warna();
```

TUGAS

Jelaskan perbedaan antara kelas abstrak dan interface!

KELAS ABSTRAK	INTERFACE
Bisa berisi abstract dan non-abstract method.	Hanya boleh berisi abstract method.
Modifiersnya harus dituliskan sendiri.	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 abstract class hanya bisa meng-extend satu abstract class lainnya.	Suatu interface bisa meng-extend satu atau lebih interface lainnya.
Suatu 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.