

**LAPORAN PRAKTIKUM**  
**PEMROGRAMAN BERORIENTASI OBJEK**

**(Dosen Pengampu: *Dede Husen, M.Kom*)**



**NAMA : MUHAMMAD RIZAL NURFIRDAUS**

**NIM : 20230810088**

**KELAS : TINFC-2023-04**

**TEKNIK INFORMATIKA**  
**FAKULTAS ILMU KOMPUTER**  
**UNIVERSITAS KUNINGAN**

**2024**

## PRETEST

1. Gambarkan contoh diagram Polimorphism dalam kehidupan sehari-hari!

```
abstract
Peliharaan      name:
String          umur: int

              makan(): void

tidur(): void

suara(): void

      Anjing              Kucing

makan(): void  makan(): void |
tidur(): void  tidur(): void |
suara(): void  suara(): void |
```

## PRAKTIKUM 1

```
package com.example.m7;
```

```
abstract public class Pulau {    static final
```

```
String NEGARA = "Indonesia";    public
```

```

abstract String nama();    public String
ambilNegara(){    return NEGARA;}

} class Jawa extends Pulau{

String namap;    public

String nama(){    namap

= "Pulau Jawa";    return

namap;

    }

} class Kalimantan extends Pulau{

String namap;    public String

nama(){    namap = "Pulau

Kalimantan";    return namap;

    }

} class JawaBarat extends

Jawa{    void namaProv(){

    System.out.println("Ini Pulau Berada di "+ ambilNegara());

    System.out.println("Ini "+ nama());

    System.out.println("Ini Provinsi Jawa Barat");

    System.out.println("Jumlah Penduduk : 232342 Jiwa");

    }

} class KalimantanTimur extends

Kalimantan{    void namaProv(){

    System.out.println("Ini Pulau Berada di "+ ambilNegara());

    System.out.println("Ini "+ nama());

```

```

        System.out.println("Ini Provinsi Jawa Sumatera");

        System.out.println("Jumlah Penduduk : 27364 Jiwa");

    }

} class Utama {    public static void

main(String[] args) {

    JawaBarat JB = new JawaBarat();

    KalimantanTimur KT = new KalimantanTimur();

    JB.namaProv();

    System.out.println(".....:P");

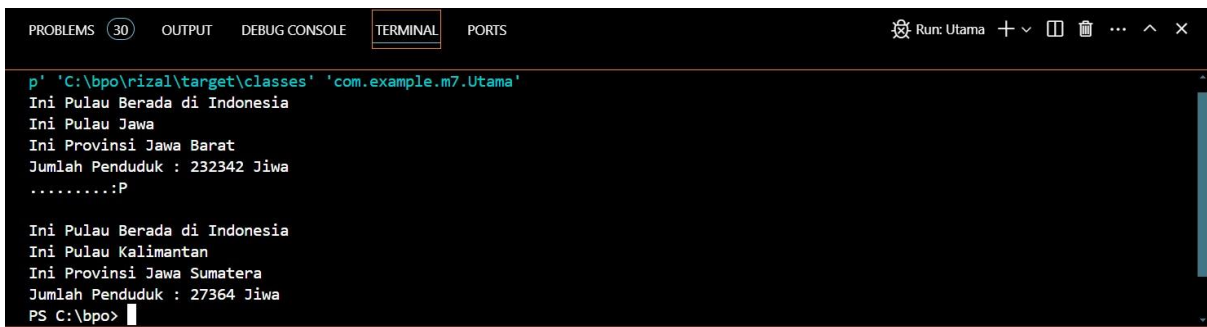
    System.out.println("");

    KT.namaProv();

}

}

```



```

p' 'C:\bpo\rizal\target\classes' 'com.example.m7.Utama'
Ini Pulau Berada di Indonesia
Ini Pulau Jawa
Ini Provinsi Jawa Barat
Jumlah Penduduk : 232342 Jiwa
.....:P

Ini Pulau Berada di Indonesia
Ini Pulau Kalimantan
Ini Provinsi Jawa Sumatera
Jumlah Penduduk : 27364 Jiwa
PS C:\bpo>

```

## PRAKTIKUM 2

```
package com.example.m7;
```

```

public abstract class MakhlukHidup {

    public abstract void berdiri ();    public

    void oksigen(){

```

```

        System.out.println("- butuh Makanan");

        System.out.println("- butuh Oksigen");

        System.out.println("- butuh air");

    }

} package

com.example.m7;

public class Manusia extends MakhlukHidup{

    private String Kaki;    public Manusia (String

    Kaki){        this.Kaki = Kaki;

        }    public void

    berdiri (){

        System.out.println("Manusia berdiri dengan : "+Kaki);

    }

} package

com.example.m7;

public class Hewan extends MakhlukHidup{

    private String Kaki, kaki1;

    public Hewan (String Kaki,String kaki1){

    this.Kaki = Kaki;        this.kaki1 = kaki1;

        }    public void berdiri

    (){

```

```

        System.out.println("Hewan berdiri dengan : "+Kaki+" "+kaki1);

    }

} package

com.example.m7;

public class Tumbuhan extends MakhlukHidup{

    private String Akar;    public

    Tumbuhan    (String    Akar){

        this.Akar = Akar;

    }    public void

    berdiri (){

        System.out.println("Tumbuhan berdiri dengan : "+Akar);

    }

} package

com.example.m7;

public class MainMakhlukHidup {    public void

cekMakhlukHidup (MakhlukHidup mHidup){

    mHidup.berdiri();    mHidup.oksigen();

}

    public static void main(String[] args) {

```

```

MainMakhlukHidup tMakhlukHidup = new MainMakhlukHidup();

tMakhlukHidup.cekMakhlukHidup(new Manusia ("Dua Kaki"));

System.out.println("-----");

tMakhlukHidup.cekMakhlukHidup(new Hewan ("Empat Kaki ", "Dua Kaki"));

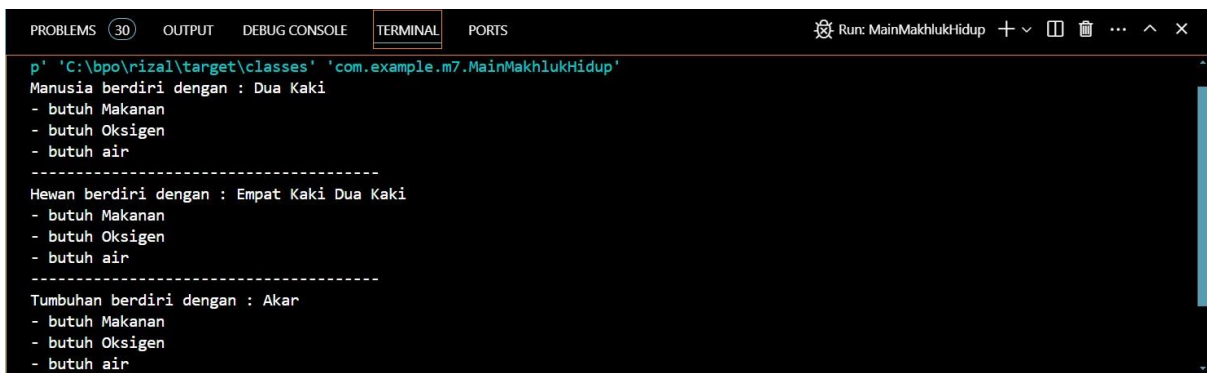
System.out.println("-----");

tMakhlukHidup.cekMakhlukHidup(new Tumbuhan ("Akar"));

}

}

```



```

p' 'C:\bpo\rizal\target\classes' 'com.example.m7.MainMakhlukHidup'
Manusia berdiri dengan : Dua Kaki
- butuh Makanan
- butuh Oksigen
- butuh air
-----
Hewan berdiri dengan : Empat Kaki Dua Kaki
- butuh Makanan
- butuh Oksigen
- butuh air
-----
Tumbuhan berdiri dengan : Akar
- butuh Makanan
- butuh Oksigen
- butuh air

```

## PRAKTIKUM 3

```
package com.example.m7;
```

```
abstract class BangunDatar {
```

```
    abstract double luas();    abstract
```

```
    double keliling();    void
```

```
    tampilLuas( double l) {
```

```
        System.out.println("Luasnya sebesar "+ l);
```

```

    }

    static void staticMethod()

    {

        System.out.println("Static Method dapat dipanggil");

    }

} package

com.example.m7;


class Lingkaran extends BangunDatar {

    double jari;

    Lingkaran (double jari){

        this.jari=jari;

    }    double

    luas()

    {        return Math.PI * jari *

    jari;

    }    double

    keliling()

    {        return Math.PI * 2.0

    *jari;

    }

}

package com.example.m7;

```



```

class Test {    public static void

main(String[] s)

    {

        Lingkaran li;    li =

new Lingkaran(7.0);

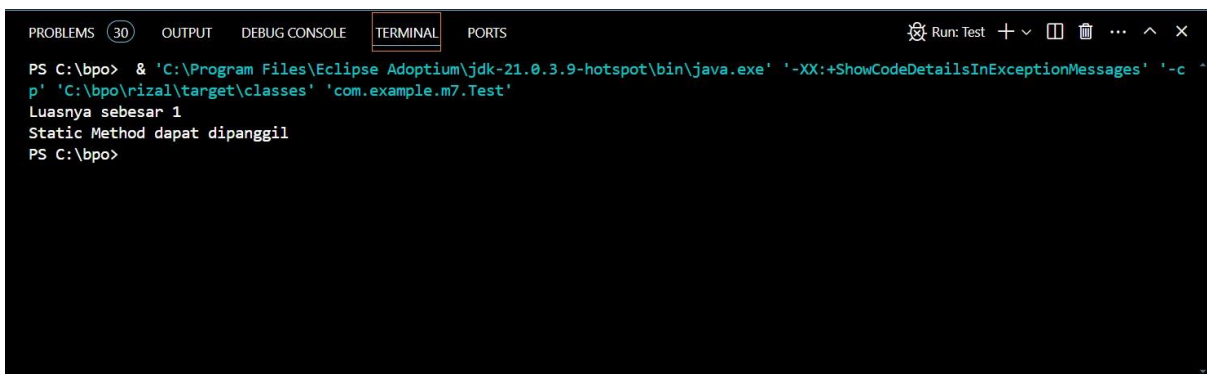
li.tampilLuas(li.luas());

        BangunDatar.staticMethod();

    }

}

```



```

PROBLEMS (30) OUTPUT DEBUG CONSOLE TERMINAL PORTS
Run: Test + - [ ] [ ] ... ^ x
PS C:\bpo> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.3.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-c '
p' 'C:\bpo\rizal\target\classes' 'com.example.m7.Test'
Luasnya sebesar 1
Static Method dapat dipanggil
PS C:\bpo>

```

## POSTTEST

1. Buatlah Program Abstract class dengan case studi yang berbeda dari praktikum diatas.

```
package com.example.m7;
```

```

abstract public class Kendaraan {

static final String MOBIL = "Bugatti";

public abstract String nama();    public

```

```

String ambilKendaraan(){    return

MOBIL;}

} class Chiron extends

Kendaraan{    String namap;

public String nama(){

namap = "Bugatti Chiron";

return namap;

    }

} class Veyron extends

Kendaraan{    String namap;

public String nama(){

namap = "Bugatti Veyron";

return namap;

    }

} class    ChironSuperSport    extends

Chiron{    void namaMerk(){

        System.out.println("Ini adalah mobil "+ ambilKendaraan());

        System.out.println("Ini Merk "+ nama());

        System.out.println("Ini Jenis Hypercar");

        System.out.println("Ini ditenagai oleh mesin 8.000 cc");

        System.out.println("Ini menghasilkan kecepatan sampai 200 mhp(321 km/jam)");

    }

```

```

} class VeyronGrandSport extends

Veyron{    void namaMerk(){

    System.out.println("Ini adalah mobil "+ ambilKendaraan());

    System.out.println("Ini Merk "+ nama());

    System.out.println("Ini Jenis Hypercar");

    System.out.println("Ini ditenagai oleh mesin 8.000 cc");

    System.out.println("Ini menghasilkan kecepatan sampai 267 mhp(429,69 km/jam)");

    }

} class Main {    public static void

main(String[] args) {

    ChironSuperSport CSP = new ChironSuperSport();

    VeyronGrandSport VGS = new VeyronGrandSport();

    CSP.namaMerk();

    System.out.println("=====");

    System.out.println("");

    VGS.namaMerk();

    }

}

```



```

PROBLEMS 30 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\bpo> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.3.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-c '
p' 'C:\bpo\rizal\target\classes' 'com.example.m7.Main'
Ini adalah mobil Bugatti
Ini Merk Bugatti Chiron
Ini Jenis Hypercar
Ini ditenagai oleh mesin 8.000 cc
Ini menghasilkan kecepatan sampai 200 mhp(321 km/jam)
=====

Ini adalah mobil Bugatti
Ini Merk Bugatti Veyron
Ini Jenis Hypercar
Ini ditenagai oleh mesin 8.000 cc
Ini menghasilkan kecepatan sampai 267 mhp(429,69 km/jam)
PS C:\bpo>

```

## TUGAS INDIVIDU UNTUK MINGGU DEPAN

### 1. Apa perbedaan Abstract dengan Interface, Jelaskan!

Abstract class dan interface adalah dua konsep penting dalam pemrograman berorientasi objek yang digunakan untuk mencapai abstraksi dan mendefinisikan kontrak untuk class lain. Abstract class dapat memiliki metode dengan implementasi konkret serta metode abstrak yang harus diimplementasikan oleh subclass. Abstract class juga bisa memiliki properti dengan atau tanpa nilai default dan constructor. Namun, sebuah class hanya bisa mewarisi satu abstract class, sehingga tidak mendukung multiple inheritance. Abstract class cocok digunakan ketika ingin menyediakan beberapa implementasi dasar dan properti yang bisa digunakan oleh class turunan.

Interface, di sisi lain, digunakan untuk mendefinisikan kontrak atau perilaku yang harus diikuti oleh class yang mengimplementasikannya. Semua metode dalam interface adalah abstrak secara default (sebelum Java 8) dan harus diimplementasikan oleh class yang mengimplementasikannya, meskipun sejak Java 8, interface bisa memiliki metode default dengan implementasi konkret. Interface tidak bisa memiliki properti dengan nilai default, hanya deklarasi konstanta, dan tidak memiliki constructor. Interface mendukung multiple inheritance, memungkinkan sebuah class untuk mengimplementasikan banyak interface. Interface ideal untuk mendefinisikan perilaku umum yang dapat diimplementasikan oleh berbagai class tanpa memperhatikan hirarki class.