Laporan Praktikum

Soal Pertama

Toni ingin membuat satu Class yang dapat menghitung luas dari beberapa jenis bangun datar sekaligus, didalam Class BangunDatar terdapat method-method yang ber-overload satu sama lain, seperti berikut

Public double luas(double r) // lingkaran
Public double luas(double a, double t) //segitiga
Public int luas(int s) // persegi
Public int luas(int p, int l) // persegi Panjang

Buatlah ke-empat method diatas kemudian cobalah panggil di Main Class!

Source Code

Class Program.java

```
public class Program {

    public double luas(double r){ //lingkaran
        return 3.14 * r * r;
    }

    public double luas(double a, double t){ //segitiga
        return 0.5 * a * t;
    }

    public int luas(int s){ //persegi
        return s * s;
    }

    public int luas(int p, int l){ //persegi panjang
        return p * l;
    }
}
```

Class Aplikasi.java

```
public class Aplikasi {
    public static void main(String[] args){
        Program program = new Program();
        System.out.println("Luas Lingkaran =
"+program.luas(14.0));
        System.out.println("Luas Persegi = "+program.luas(5));
```

```
        System.out.println("Luas Segitiga =
"+program.luas(6.5, 2.5));
        System.out.println("Luas Persegi Panjang =
"+program.luas(7, 2));

    }
}
```

Output Program

```
"C:\Program Files\Java\jdk-16.0.1\bi
Luas Lingkaran = 615.44
Luas Persegi = 25
Luas Segitiga = 8.125
Luas Persegi Panjang = 14

Process finished with exit code 0
```

Laporan Praktikum

Soal Kedua

Perhatikan kode program berikut ini:

```
public class SuperClass {
    public static void cetak1(){
        System.out.println("Static method superclass");
    }

    public void cetak2(){
        System.out.println("Final method superclass");
    }

    public void cetak3(){
        System.out.println("Method dari superclass");
    }
}
```

```
public class AdaYangSalah extends SuperClass{
    public static void cetak1(){
        System.out.println("Static method superclass");
    }

    @Override
    public final void cetak2(){
        System.out.println("Final method superclass");
    }

    @Override
    public void cetak3(){
        System.out.println("Method dari superclass");
    }
}
```

Cobalah kode program diatas, jika ada error, jelaskan apa yang menyebabkan error dari kode program diatas, kemudian berikan kode program yang sudah benar!

Jawaban

Program tersebut mengalami error karena method final tidak dapat di override. Oleh karena itu, solusi penyelesaiannya dari error tersebut adalah mengganti final pada method dengan static atau tidak diberi apapun.

Source Code

```
public class Perbaikan extends SuperClass{
    public static void cetak1(){
        System.out.println("Static method superclass");
    }

    @Override
    public void cetak2(){
        System.out.println("Final method superclass");
    }

    @Override
    public void cetak3(){
        System.out.println("Method dari superclass");
    }
}
```

Output Program

```
"C:\Program Files\Java\jdk-16.0.1\
Static method superclass
Final method superclass
Method dari superclass

Process finished with exit code 0
```

Laporan Praktikum

Soal Ketiga

Buatlah sebuah abstract Class dengan nama Kucing beserta metod-methodnya kemudian buatlah bebrapa Class jenis-jenis Kucing yang meng-extends ke Abstract Class Kucing dan buatlah interface dengan nama Kaki, Ekor, dan Cakar ke Class Kucing!

Source Code

Class Cakar.java (interface)

```
package fisik;

public interface Cakar {
    String cakar1 = "tajam", cakar2 = "tumpul";

    public void kegiatan1();
    public void kegiatan2();
}
```

Class Ekor.java (interface)

```
package fisik;

public interface Ekor {
    String ekor1 = "panjang";

    public void sifat();
}
```

Class Kaki.java (interface)

```
package fisik;

public interface Kaki {
    String kaki1 = "besar", kaki2 = "kecil";

    public void gerakan1();

    public void gerakan2();

}
```

Class Kucing.java (Abstract)

```
package jenis.mother;

import fisik.Cakar;
import fisik.Ekor;
import fisik.Kaki;

public abstract class Kucing implements Cakar, Ekor, Kaki {
    protected String nama;

    public Kucing(){
    }

    public Kucing(String nama){
        this.nama = nama;
    }

    @Override
    public void kegiatan1() {
        System.out.print(", memiliki cakar "+cakar1+" sedang
mencakar");
    }

    @Override
    public void kegiatan2() {
        System.out.print(", memiliki cakar "+cakar2+" sedang
mencakar");
    }

    @Override
    public void sifat() {
        System.out.print(", dan memiliki ekor "+ekor1+" sedang
mengibas");
    }

    @Override
    public void gerakan1() {
        System.out.print(" dengan kaki "+kaki1+" sedang
berjalan");
    }

    @Override
    public void gerakan2() {
        System.out.print(" dengan kaki "+kaki2+" Sedang
berjalan");
```

```
    }
}
```

Class Himalaya.java

```
package jenis;
import jenis.mother.Kucing;

public class Himalaya extends Kucing {

    public Himalaya(String nama){
        super(nama);
    }

    public void kucing(){
        System.out.print("Ini adalah kucing berjenis Himalaya
yang bernama "+nama);
    }
}
```

Class Mainecoon.java

```
package jenis;
import jenis.mother.Kucing;

public class MaineCoon extends Kucing {

    public MaineCoon(String nama){
        super(nama);
    }

    public void kucing(){
        System.out.print("Ini adalah kucing berjenis Maincone
yang bernama "+nama);
    }
}
```

Class Main.java

```
import jenis.Himalaya;
import jenis.MaineCoon;

public class Main {
    public static void main(String[] args) {
        MaineCoon maineCoon = new MaineCoon("Ciko");
```

```
        Himalaya himalaya = new Himalaya("Deno");

        maineCoon.kucing();
        maineCoon.gerakan1();
        maineCoon.kegiatan1();
        maineCoon.sifat();
        System.out.println();
        System.out.println("============================");
        himalaya.kucing();
        himalaya.gerakan2();
        himalaya.kegiatan2();
        himalaya.sifat();
    }
}
```

Output Program

```
Ini adalah kucing berjenis Maincone yang bernama Ciko dengan kaki besar sedang berjalan, memiliki
 cakar tajam sedang mencakar, dan memiliki ekor panjang sedang mengibas
============================
Ini adalah kucing berjenis Himalaya yang bernama Deno dengan kaki kecil Sedang berjalan, memiliki
 cakar tumpul sedang mencakar, dan memiliki ekor panjang sedang mengibas
Process finished with exit code 0
```

Laporan Praktikum

Soal Keempat

Perhatikan baris kode interface berikut.

```
public interface AdaYangSalah {
    void iniMethod(int iniParam){
        System.out.println(iniParam);
    }
}
```

Apa yang salah dari penerapan Interface diatas? Coba tuliskan baris kode diatas!

Jawaban

Yang salah dari program interface tersebut adalah tidak boleh ada isi di dalamnya (dalam badan method “iniMethod()”), melainkan hanya nama method dan parameter yang diperbolehkan.

Source Code

```
public interface AdaYangSalah {
    void iniMethod(int iniParam);
}
```

Laporan Praktikum

Soal Kelima

PROGRESS STUDI KASUS PROJECT AKHIR:

Berdasarkan studi kasus kalian cobalah untuk buat/ terapkan penggunaan Compile-Time Polymorphism Abstraction atau Interface ke dalam project Studi Kasus kalian!

Source Code

Class Transaksi.java

```
package entity.Mother;
import utils.DateString;

public abstract class Transaksi{

    private String tanggalTransaksi;

    public Transaksi(){
        tanggalTransaksi = DateString.now();
    }

    public String getTanggalTransaksi(){
        return tanggalTransaksi;
    }

    public abstract int getNilaiNominal();
}
```

Class SetorTunai.java

```
package entity;

import entity.Mother.Transaksi;

public class SetorTunai extends Transaksi {

    private int nilaiNominalSetor;

    public SetorTunai(int nominalSetor){
        this.nilaiNominalSetor = nominalSetor;
    }

    @Override
```

```
    public int getNilaiNominal(){
        return nilaiNominalSetor;
    }
}
```

Class TarikTunai.java

```
package entity;

import entity.Mother.Transaksi;

public class TarikTunai extends Transaksi {

    private int nilaiNominalTarik;

    public TarikTunai(int nominalTarik){
        this.nilaiNominalTarik = nominalTarik;
    }

    @Override
    public int getNilaiNominal(){
        return nilaiNominalTarik;
    }
}
```

Class Transfer.java

```
package entity;

import entity.Mother.Transaksi;

public class Transfer extends Transaksi {

    private int nilaiNominalTransfer;
    private User userAsal, userTujuan;

    public Transfer(int nominal, User asal, User tujuan){
        this.nilaiNominalTransfer = nominal;
        this.userAsal = asal;
        this.userTujuan = tujuan;
    }

    @Override
    public int getNilaiNominal(){
        return nilaiNominalTransfer;
```

```
    }

    public User getUserAsal(){
        return userAsal;
    }

    public User getUserTujuan(){
        return userTujuan;
    }
}
```