MODUL 8

MULTI INHERITANCE

A. Tujuan

1. Mahasiswa memahami dan bisa mengimplementasikan konsep penurunan sifat dari beberapa class induk dengan mengguanakan *Interface*.
2. Memahami jenis Multi *Inheritance*

B. Dasar Teori

B.1 *Abstract Method*

Abstract Method adalah sebuah '*method* dasar' yang harus diimplementasikan ulang di dalam class anak (*child class*). *Abstract method* ditulis tanpa isi dari *method*, melainkan hanya '*signature'*-nya saja. *Signature* dari sebuah *method* adalah bagian *method* yang terdiri dari nama *method* dan parameternya (jika ada). *Abstract method* merupakan sebuah *method* yang dideklarasikan dengan menambahkan *keyword abstract* pada deklarasinya, dan tanpa ada implementasi dari *method* tersebut. Dalam arti, hanya pendeklarasian saja, tanpa tanda sepasang kurung kurawal. Tetapi diakhiri dengan tanda titik koma(;), seperti contoh sederhana berikut ini :

```
abstract void
setName(); abstract void
setMakanan();
```

Kode program diatas artinya : Mendeklarasikan *abstract method* dengan nama `setName` dan `setMakanan`. Jadi intinya *method* abstrak itu adalah sebuah *method* yang tidak tahu mau kita apakan nantinya, sebuah *class* dan *method* abstrak dibuat oleh seorang programmer sebagai acuan atau gambaran dari program yang ingin mereka buat.

B.2 *Interface*

Interface memiliki pengertian dan fungsi yang hampir sama dengan class abstract, walalupun fungsi dari keduanya sama, tetapi ada beberapa perbedaan

yang perlu diketahui, berikut ini adalah tabel yang menjelaskan perbedaan diantara *abstract* dan *interface*:

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 *int*

B.3 *Multiple Inheritance*

Multiple inheritance adalah fitur dari konsep berorientasi object, dimana sebuah class dapat mewariskan propertinya kepada lebih dari satu parent class. Permasalahan yang terjadi ketika terdapat method dengan signature sama dalam kedua super class dan subclass tersebut. Ketika method dipanggil maka kompilator tidak dapat menentukan method class mana yang akan dipanggil, bahkan ketika memanggil method class mana yang harus diproritaskan terlebih dahulu.

Satu-satunya cara untuk mengimplementasikan multiple inheritance pada Java adalah dengan mengimplementasikan banyak interface dalam sebuah class. Pada Java, satu class dapat mengimplementasikan dua atau lebih interface, dimana hal ini tidak akan menyebabkan ambiguitas karena semua method yang dideklarasikan dalam interface diimplementasikan pada class.

C. Studi Kasus

Kelompok Ganjil : Buatlah aplikasi kalkulator menggunakan konsep multiple inheritance menggunakan interface. Kalkulator tersebut terdiri dari 5 fitur yaitu menghitung penjumlahan, pengurangan, perkalian, pembagian, perpangkatan dan akar kuadrat. Program dibuat secara dinamis dengan menu pilihan dapat menggunakan condition dan if statement maupun swith case.

D. Data Hasil

a. *Flowchart*

b. *Script* Program

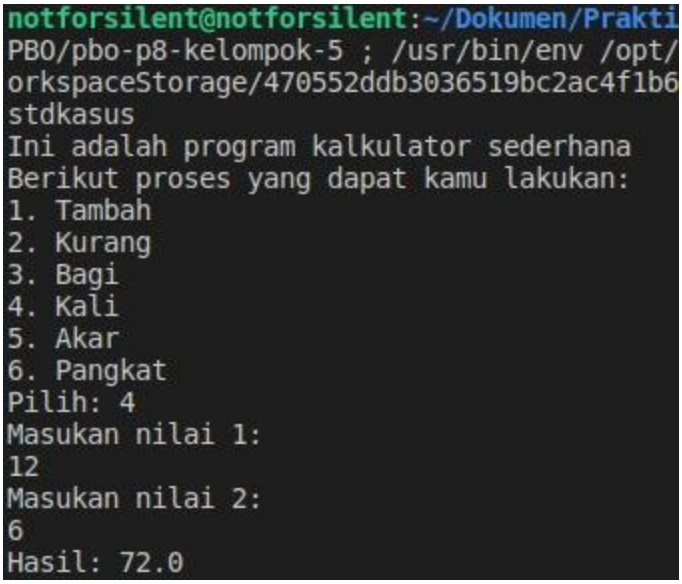
```
package studi_kasus;

/*Buatlah aplikasi kalkulator menggunakan konsep multiple
inheritance menggunakan
interface. Kalkulator tersebut terdiri dari 5 fitur yaitu
menghitung penjumlahan,
pengurangan, perkalian, pembagian, perpangkatan dan akar kuadrat.
Program dibuat
secara dinamis dengan menu pilihan dapat menggunakan condition dan
if statement
maupun swith case */

import java.util.Scanner;

public class stdkasus extends proces{

    public static void main(String[] args) {
        stdkasus data = new stdkasus();
        Scanner in = new Scanner(System.in);

        double a,b;

        System.out.println("Ini adalah program kalkulator
sederhana");

        System.out.println("Berikut proses yang dapat kamu
lakukan:");
        System.out.println("1. Tambah");
        System.out.println("2. Kurang");
        System.out.println("3. Bagi");
        System.out.println("4. Kali");
        System.out.println("5. Akar");
        System.out.println("6. Pangkat");
        System.out.print("Pilih: ");
        int pilih = in.nextInt();


        switch (pilih) {
            case 1:
                System.out.println("Masukan nilai 1: ");
                a = in.nextDouble();
                System.out.println("Masukan nilai 2: ");
                b = in.nextDouble();
                System.out.println("Hasil: "+data.tambah(a, b));
                break;
            case 2:
                System.out.println("Masukan nilai 1: ");
                a = in.nextDouble();
                System.out.println("Masukan nilai 2: ");
                b = in.nextDouble();
                System.out.println("Hasil: "+data.kurang(a, b));
                break;
            case 3:
```

```
                System.out.println("Masukan nilai 1: ");
                a = in.nextDouble();
                System.out.println("Masukan nilai 2: ");
                b = in.nextDouble();
                System.out.println("Hasil: "+data.bagi(a, b));
                break;
            case 4:
                System.out.println("Masukan nilai 1: ");
                a = in.nextDouble();
                System.out.println("Masukan nilai 2: ");
                b = in.nextDouble();
                System.out.println("Hasil: "+data.kali(a, b));
                break;
            case 5:
                System.out.println("Masukan nilai 1: ");
                a = in.nextDouble();
                System.out.println("Hasil: "+data.akar(a));
                break;
            case 6:
                System.out.println("Masukan nilai 1: ");
                a = in.nextDouble();
                System.out.println("Masukan nilai 2: ");
                b = in.nextDouble();
                System.out.println("Hasil: "+data.pangkat(a, b));
                break;

            default:
                break;
        }
    }
}
```

```
package studi_kasus;

interface tambah{
    public double tambah(double a, double b);
}

interface kali{
    public double kali(double a, double b);
}

interface kurang{
    public double kurang(double a, double b);
```

```
}

interface bagi{
    public double bagi(double a, double b);
}

interface pangkat{
    public double pangkat(double a, double b);
}

interface akar{
    public double akar(double a);
}

class proces implements tambah,kali,kurang,akar,bagi,pangkat{
    @Override
    public double tambah(double a, double b){
        double hasil = a + b;
        return hasil;
    }

    @Override
    public double kurang(double a, double b){
        double hasil = a - b;
        return hasil;
    }

    @Override
    public double kali(double a, double b){
        double hasil = a * b;
        return hasil;
    }

    @Override
    public double bagi(double a, double b){
        double hasil = a / b;
        return hasil;
    }

    @Override
    public double pangkat(double a, double b){
        double hasil=1;

        for (int i = 0; i<b;i++){
            hasil = hasil * a;
        }
        return hasil;
    }
    @Override
    public double akar(double a){
        double hasil = Math.sqrt(a);
        return hasil;
    }


}
```

c. Hasil *Running*



DAFTAR PUSTAKA

Anonim. 2023. *Modul Praktikum Pemrograman Berorientasi Objek*. Laboratorium Jaringan dan Komputer. Jurusan Teknik Elektro. Fakultas Teknik. Universitas Mataram

Andre. (2014, October 10). *Tutorial Belajar OOP PHP Part 15: Pengertian Abstract Class dan Abstract Method PHP*. Retrieved June 5, 2021, from Duniailkom website: https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-abstract-class-danabstract-method-php/

Athoillah, W. M. (2017, November 5). *Penggunaan Class Abstract dan Interface pada Java.* Retrieved June 5, 2021, from WILDAN TECHNO ART website: https://www.wildantechnoart.net/2017/11/class-abstract-dan-interface-pada-java.html

Elfan mauludi. (2021). *Java dan Multiple Inheritance*. Retrieved June 5, 2021, from Elfanmauludi.id website: https://www.elfanmauludi.id/2019/05/java-dan-multipleinheritance.h