VERSION 2.0

FEBRUARI, 2021

PEMOGRAMAN BERORIENTASI OBJEK

ABSTRACT CLASS, INTERFACE, POLYMORPHISM. MODUL 4

TIM PENYUSUN:

- GALIH WASIS W S.KOM,.M.CS.
- FARLI NAHRUL JAVIER
- MUHAMMAD NUR ICHSAN

PRESENTED BY: LAB. INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

PEMOGRAMAN BERORIENTASI OBJEK

CAPAIAN PEMBELAJARAN MATA KULIAH

Mahasiswa mampu memahami dan menerapkan :
1. Abstract method.
2. Abstract class.
3. Konsep interface.
4. Method overriding.
5. Konsep polymorphism.

SUB CAPAIAN PEMBELAJARAN MATA KULIAH

Mahasiswa mampu mengimplementasikan :
1. Method overriding.
2. Multiple interface.
3. Perbedaan abstract class dan interface.
4. Interface, anstract class, dan inheritance dalam konsep polymorphism.

KEBUTUHAN HARDWARE & SOFTWARE

1. Compiler java (JDK), JRE.
2. Editor Java (NetBeans, Gel, Eclipse, Jcreator, dll).

MATERI POKOK

1. Abstract method

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(;).

```
public abstract int myMethod(int n1, int n2);
```

2. Abstract class

Adalah class yang sifatnya abstrak yang biasanya digunakan untuk mendefinisikan konsep class yang sifatnya umum. Sebagai contoh jika ada class hewan, kucing dan kelinci. karena hewan sifatnya sangat umum, maka bisa dideklarasikan sebagai abstract class. Dalam implementasinya, abstract class tidak bisa dibuat objectnya, namun bisa dimanfaatkan dalam konsep polimorfisme. Di dalam abstract class bisa dideklarasikan variabel class, concrete method dan abstract method. Concrete method adalah method yang memiliki kode program / body program, artinya sudah diimplementasi. Sedangkan abstract method adalah method yang hanya berupa deklarasi method saja, yang terdiri dari nama method, jenis luaran, modifier dan input parameter tanpa kode program. Selanjutnya, untuk class yang merupakan turunan dari abstract class, jika itu adalah concrete class maka harus mengimplementasi abstract class (bentuknya seperti overriding), sedangkan jika absract class juga maka bisa tidak mengimplementasi abstract class.

```
//abstract class
public abstract class Person {

	private String name;
	private String gender;

	public Person(String nm, String gen){
		this.name=nm;
		this.gender=gen;
	}

	//abstract method
	public abstract void work();

	@Override
	public String toString(){
		return "Name="+this.name+"::Gender="+this.gender;
	}

	public void changeName(String newName) {
		this.name = newName;
	}
}
```

3. Interface

Interface bukanlah class, meskipun deklarasinya menyerupai. Penamaan interface biasanya menggunakan kata sifat seperti Fireable, Rideable, Movable, dan lain-lain. Hal ini berhubungan dengan konsep interface yang digunakan untuk mendeklarasikan sifat-sifat tertentu. Sifat-sifat ini yang nantinya akan diimplementasi oleh class sesuai dengan kebutuhan. Sebagai contoh jika ada interface Fireable dan di dalamnya ada method shot() – menembak, Maka interface ini bisa diimplementasi oleh class Pistol, Tank, Bazooka, PesawatTemput yang memang memiliki kemampuan untuk menembak. Interface hanya memiliki abstract method saja, tidak ada yang concrete, sehingga pada saat menulis deklarasi method bisa tidak menggunakan keyword “abstract”, meskipun boleh juga menggunakan keyword “abstract”. Interface bisa juga memiliki variable, namun variable itu harus diberi nilai serta otomatis menjadi konstanta dengan modifier public, final dan static. Keyword yang digunakan untuk mengimplementasikan interface adalah “implements”.

```
public interface Bentuk{
	void tampil();
}
```

4. Method overriding

Method Overloading adalah sebuah kemampuan yang membolehkan sebuah class mempunyai 2 atau lebih method dengan nama yang sama, yang membedakan adalah parameternya.

Pada method overloading perbedaan parameter mencakup :

a. Jumlah parameter
b. Tipe data dari parameter
c. Urutan dari tipe data parameter

Method Overloading juga dikenal dengan sebutan Static Polymorphism. Berikut ini contoh Class yang melakukan Overloading.

```
public class Sum {

    // Overloaded sum(). This sum takes two int parameters
    public int sum(int x, int y)
    {
        return (x + y);
    }

    // Overloaded sum(). This sum takes three int parameters
    public int sum(int x, int y, int z)
    {
        return (x + y + z);
    }

    // Overloaded sum(). This sum takes two double parameters
    public double sum(double x, double y)
    {
        return (x + y);
    }

    // Driver code
    public static void main(String args[])
    {
        Sum s = new Sum();
        System.out.println(s.sum(10, 20));
        System.out.println(s.sum(10, 20, 30));
        System.out.println(s.sum(10.5, 20.5));
    }
}
```

5. Polymorphism

Polimorfisme merupakan kemampuan objek-objek yang berbeda kelas namun terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama. Polimarfisme juga dapat dikatakan kemampuan sebuah objek untuk memutuskan method mana yang akan diterapkan padanya, tergantung letak objek tersebut pada jenjang pewarisan.

```
public class Employee extends Person {

      private int empId;

      public Employee(String nm, String gen, int id) {
            super(nm, gen);
            this.empId=id;
      }

      @Override
      public void work() {
```

```
            if(empId == 0){
                System.out.println("Not working");
            }else{
                System.out.println("Working as employee!!");
            }
        }

        public static void main(String args[]){
            //coding in terms of abstract classes
            Person student = new Employee("Dove","Female",0);
            Person employee = new Employee("Pankaj","Male",123);
            student.work();
            employee.work();
            //using method implemented in abstract class - inheritance
            employee.changeName("Pankaj Kumar");
            System.out.println(employee.toString());
        }
    }
```

MATERI PRAKTIKUM

1. **Abstract**

   Buatlah class-class berikut dalam satu package.

   Class abstract hewan

```
package Abstract;

public abstract class hewan {
    abstract void setNama();
    abstract void setMakanan();
}
```

   Class kucing

```
package Abstract;

public class kucing extends hewan{

    @Override
    void setNama() {
        System.out.println("Nama hewan adalah Kucing");
    }

    @Override
    void setMakanan() {
        System.out.println("Makanan kucing adalah ikan");
    }

    void setWarna(){
        System.out.println("Warana putih");
    }
}
```

Class sapi

```
package Abstract;

public class sapi extends hewan{

    @Override
    void setNama() {
        System.out.println("Nama hewan adalah Sapi");
    }

    @Override
    void setMakanan() {
        System.out.println("Makanan sapi adalah rumput");
    }

    void setUkuran(){
        System.out.println("ukuran hewan besar");
    }
}
```

Class main

```
package Abstract;

public class main {
    public static void main(String[] args) {
        kucing k = new kucing();
        k.setNama();
        k.setMakanan();
        k.setWarna();

        sapi s = new sapi();
        s.setNama();
        s.setMakanan();
        s.setUkuran();
    }
}
```

2. Interface

Buatlah class-class berikut dalam satu package.

Interface fireable

```
package Interface;

public interface fireable {
    public void shot();
}
```

Interface moveable

```
package Interface;

public interface moveable {
    public void move();
}
```

Class tank

```
package Interface;

public class tank implements fireable, moveable{

    @Override
    public void shot() {
        System.out.println("Tank menembak");
    }

    @Override
    public void move() {
        System.out.println("Tank bergerak menyerang");
    }

}
```

Class pistol

```
package Interface;

public class pistol implements fireable{

    @Override
    public void shot() {
        System.out.println("Pistol menembak");
    }

}
```

Class main

```
package Interface;

public class main {
    public static void main(String[] args) {
        moveable m = new tank();
        fireable f = new pistol();

        m.move();
        f.shot();
        f = (fireable) m;
        f.shot();
    }
}
```

LEMBAR KERJA

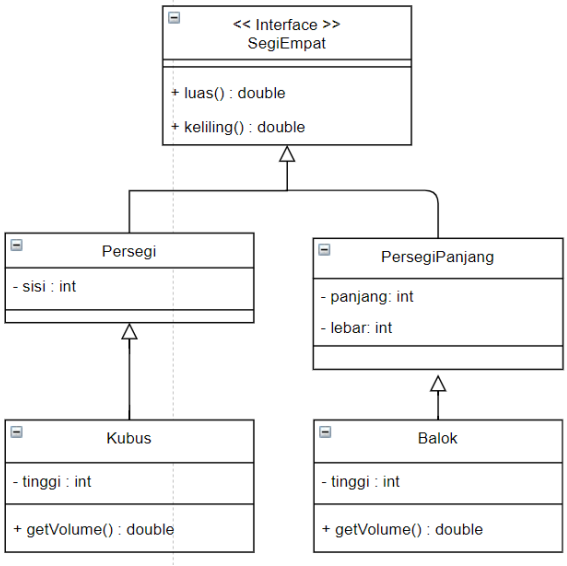
KEGIATAN 1

Buatlah 2 abstract method yaitu getLuasPermukaan dan getVolume di dalam kelas Abstract BangunRuang. Kemudian turunkan dua method ini kedalam kelas Tabung dan Balok (ingat bahwa masing masing menerapkan rumus matematika sendiri). Kemudian buatlah 1 buah driver class untuk outputannya

KEGIATAN 2

Pilihlah salahsatu soal di bawah ini. Nilai menyesuaikan bobot dari soal.

- Soal 1 (easy)

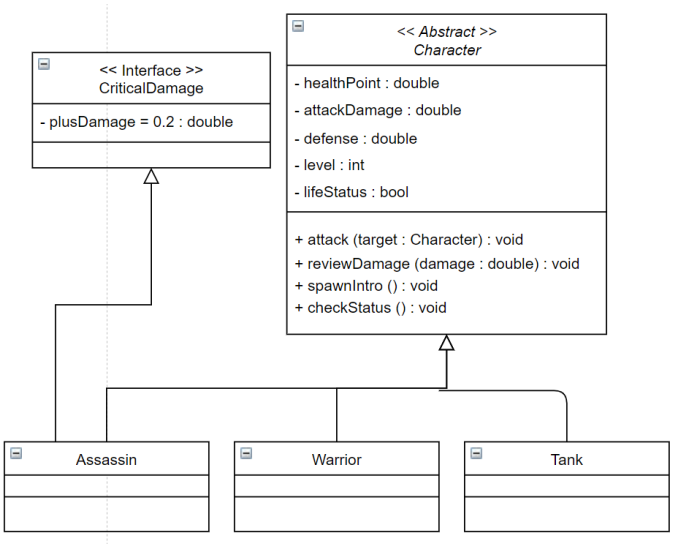


Buatlah sebuah kerangka program penghitungan keliling, luas dan volume sesuai dengan gambar yang telah diberikan di atas dengan syarat :

1. Class Persegi dan PersegiPanjang mengimplementasikan method yang ada pada interface SegiEmpat
2. Class Kubus merupakan subclass dari class Persegi, sedangkan class Balok merupakan subclass dari PersegiPanjang

Tambahkan sebuah class Driver untuk mengimplementasikan kerangka program diatas

- Soal 2 (hard)



Buatlah sebuah Fighting Game sederhana sesuai dengan class diagram diatas dengan syarat dan ketentuan :

- Character memiliki healthpoint, attackDamage, defense, level, dan lifeStatus
  - Healthpoint : status yang menunjukkan jumlah satuan kesehatan karakter game.
  - attackDamage : nilai serangan yang akan diberikan untuk mengurangi healthpoint lawan
  - Defense : ketahanan tubuh character. Digunakan ketika mendapat serangan dari lawan
  - Level : tingkat kemahiran / kehebatan character
  - lifeStatus : status character (hidup / mati)
- method attack(..) ,reviewDamage(..), dan checkStatus() merupakan method concrete, sedangkan spawnIntro() merupakan abstract method.
- attack(target : Character) : menyerang target.
- reviewDamage(damage : int) : mengurangi healthPoint sebesar damage yang telah dikurangi defense (realDamage)

  realDamage = attackDamage – defense

  healthPoint tidak boleh kurang dari 0 dan jika healthPoint menyentuh nilai 0 maka lifeStatus akan berubah menjadi false atau Character tersebut telah mati dan tidak dapat melakukan apapun.
- spawnIntro() : dialog awal karakter ketika telah memasuki giliran Character tersebut.
  Ex. Character Tank : System.out.println("\'U can't defeat me kid!!\'");
- checkStatus() : menghasilkan data-data attribut karakter
- spesifikasi dasar Character adalah lifeStatus:true dan menyesuaikan jenis Character dibawah:
  - Assassin : healthPoint:2500, defense:200, attackDamage:750
  - Warrior : healthPoint:3000, defense:250, attackDamage:800
  - Tank : healthPoint:5000, defense:400, attackDamage:400
- Tiap jenis Character memiliki constructor dengan 1 parameter (level : int) yang digunakan untuk menginisialisasi nilai-nilai attribut
- level pada parameter constructor digunakan untuk menambahkan nilai attribut Character jika level > 0, dengan nilai tiap 1 levelnya :
  - Assassin : +80 healthPoint, +10defense, +25attackDamage
  - Warrior : +100 healthPoint, +10defense, +15attackDamage

        - Tank : +150 healthPoint, +15defense, +10attackDamage
    - Khusus Character Assassin, mendapatkan Critical Damage atau tambahan attackPoint selain dari level. Pada Constructor menambahkan attackDamage sebesar attackDamage * plusDamage(0.2) atau sebesar 20% pada interface CriticalDamage.

        attackDamage += attackDamage * plusDamage
    - Boleh menambahkan getter and setter jika perlu.

Tambahkan sebuah class Driver untuk mengimplementasikan objek game yang telah dibuat. Game tersebut nantinya akan berlangsung dengan mode 1v1 dengan giliran attack bergantian dan akan berhenti ketika salah satu character mati. Character yang masih hidup adalah pemenang. Contoh P1(Assassin lvl.2) vs P2(Tank lvl.0) outputnya adalah sebagai berikut :

Output :

```
===== PLAYER 1 ====
level : 2
healthPoint : 2660.0        defense : 220.0
attackDamage : 960.0        lifeStatus : true

==== PLAYER 2 ====
level : 0
healthPoint : 5000.0        defense : 400.0
attackDamage : 400.0        lifeStatus : true

############ ROUND START ############

=== TURN 1 ===
- Player 1 Move -
'I need ur blood'
Player 2 Remaining HP : 4440.0
- Player 2 Move -
'U Can't defeat me kid!!'
Player 1 Remaining HP : 2480.0

=== TURN 2 ===
- Player 1 Move -
'I need ur blood'
Player 2 Remaining HP : 3880.0
- Player 2 Move -
'U Can't defeat me kid!!'
Player 1 Remaining HP : 2300.0

=== TURN 3 ===
- Player 1 Move -
'I need ur blood'
Player 2 Remaining HP : 3320.0
- Player 2 Move -
'U Can't defeat me kid!!'
Player 1 Remaining HP : 2120.0

=== TURN 4 ===
```

```
- Player 1 Move -
'I need ur blood'
Player 2 Remaining HP : 2760.0
- Player 2 Move -
'U Can't defeat me kid!!'
Player 1 Remaining HP : 1940.0

=== TURN 5 ===
- Player 1 Move -
'I need ur blood'
Player 2 Remaining HP : 2200.0
- Player 2 Move -
'U Can't defeat me kid!!'
Player 1 Remaining HP : 1760.0

=== TURN 6 ===
- Player 1 Move -
'I need ur blood'
Player 2 Remaining HP : 1640.0
- Player 2 Move -
'U Can't defeat me kid!!'
Player 1 Remaining HP : 1580.0

=== TURN 7 ===
- Player 1 Move -
'I need ur blood'
Player 2 Remaining HP : 1080.0
- Player 2 Move -
'U Can't defeat me kid!!'
Player 1 Remaining HP : 1400.0

=== TURN 8 ===
- Player 1 Move -
'I need ur blood'
Player 2 Remaining HP : 520.0
- Player 2 Move -
'U Can't defeat me kid!!'
Player 1 Remaining HP : 1220.0

=== TURN 9 ===
- Player 1 Move -
'I need ur blood'
Player 2 Remaining HP : 0.0
- Player 2 Move -
CAN'T ATTACK, ALREADY DIED
Player 1 Remaining HP : 1220.0

PLAYER 1 WINN
```

RUBRIK PENILAIAN

Pemahaman : 40

Kegiatan 1 : 20

Kegiatan 2 : Opsional

Soal 1 : 20; atau

Soal 2 : 40