Praktikum Pemrograman Berbasis Objek

Pertemuan 5

Review Time 

1. Inheritance
2. Method Overriding
3. UML (Class Diagram)

Materi Pertemuan 5

Gambaran Materi Pembelajaran Hari Ini

Materi Pertemuan 5

01

Polimorfisme

Mengenal sifat polimorfisme objek dari PBO

02

Abstract Class

Mengenal class yang berbentuk abstrak

03

Interface

Mengenal interface sebagai bentuk lain dari abstract class

04

UML Interface dan Abstract Class

Mengenal Interface dan class abstrak dalam bentuk UML

05

Operator Instanceof

Mengenal penggunaan operator instanceof

06

Casting

Mengenal istilah lainnya yang terdapat dalam polimorfisme

01

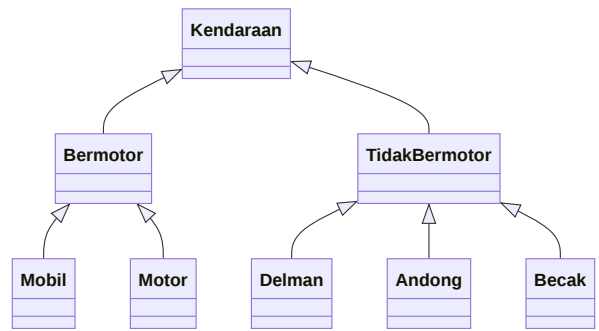
Polimorfisme

Polimorfisme

Polimorfisme adalah kemampuan untuk mempunyai beberapa bentuk class yang berbeda. Polimorfisme ini terjadi pada saat suatu objek bertipe **parent class**, akan tetapi **pemanggilan constructornya melalui child class/sub class.**

Contoh:

```
Kendaraan bmw = new Mobil();
```



Virtual Method Invocation (VMI)

Virtual Method Invocation (VMI) dapat terjadi ketika **adanya polimorfisme dan method yang di-override.** JVM secara otomatis akan **memanggil method yang sudah ter-override.**

```
public class Parent {
    int x = 5;

    public void showInfo() {
      System.out.println("Parent");
    }
}
```

```
public class Child extends Parent {
    int x = 10;

    @Override
    public void showInfo() {
      system.out.println("Child");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Parent testParent = new Child();
        System.out.println("Nilai x = " + testParent.x); /
        testParent.showInfo(); // Child
    }
}
```

Virtual Method Invocation (lanjutan)

Ketika melakukan polimorfisme pada suatu objek, **atribut dan method pada tipe data yang dideklarasikan**-lah yang diketahui oleh objek tersebut.

```
Parent test = new Child();
```

Object **test** hanya mengetahui atribut dan method pada **Parent** saja.

Tetapi ketika ada method parent yang di-override pada **Child** class, maka JVM akan memanggil method tersebut (konsep VMI)

02

Abstract Class

Abstract Class

Adalah class dengan kata kunci abstract. Aturan-aturan:

- Abstract class bisa berisi method abstract ataupun tidak.
- Method abstract hanya deklarasi saja tanpa body
- Jika ada **minimal** 1 method abstract pada suatu class, maka class tersebut **wajib** dinyatakan abstract.
- Abstract class **tidak dapat diinstansiasi**
- Abstract class bisa diakses dengan konsep **Inheritance**
- Sub-class yang meng-extend Abstract class **wajib meng-override method abstract** di dalam Abstract class

Abstract Class (lanjutan)

Contoh penggunaan abstract class

```
public abstract class Animal {
    private String name;

    public Animal(String name) {
      this.name = name;
    }

    public String getName() {
      return name;
    }

    public abstract String getSound();
}
```

```
public class Cat extends Animal {
    public Cat(String name) {
      super(name);
    }

    @Override
    public String getSound() {
      return "Miaw";
    }
}
```

Abstract Class (other)

Python

```python
from abc import ABC,abstractmethod

class Animal(ABC):
    @abstractmethod
    def getSound(self):
        pass

class Snake(Animal):
    def getSound(self):
        print("HISSS")
```

C++

```cpp
class Animal {
   public:
      virtual int getSound() = 0;
};

class Tiger: public Animal {
  public:
    void getSound() {
      cout << "Roar";
    }
};
```

03

Interface

Interface

Interface berbentuk abstract secara implisit, baik dari keyword Interface nya, maupun method-method didalamnya.

Jika Class diibaratkan sebagai sebuah Blueprint, Interface di sisi lain diibaratkan sebagai sebuah Contract yang wajib dipatuhi (dalam arti diimplementasikan).

Method-method pada interface secara implisit memiliki akses modifier public.

Perhatikan penulisan interface serta pengimplementasiannya pada gambar disamping

```java
interface Animal {
  public void eat();
  public void travel();
}
```

```java
public class Fish implements Animal {
    private String name;

    public Fish(String name) {
      this.name = name;
    }

    @Override
    public void eat() {
      System.out.println("Eat");
    }

    @Override
    public void travel() {
      System.out.println("Swim");
    }
}
```

Interface (lanjutan)

Suatu class dapat mengimplementasi lebih dari 1 Interface.

```
public class Fish implements Animal, Mammals {
  private String name;

  public Fish(String name) {
    this.name = name;
  }
}
```

Interface dapat dapat meng-extend lebih dari 1 Interface lainnya.

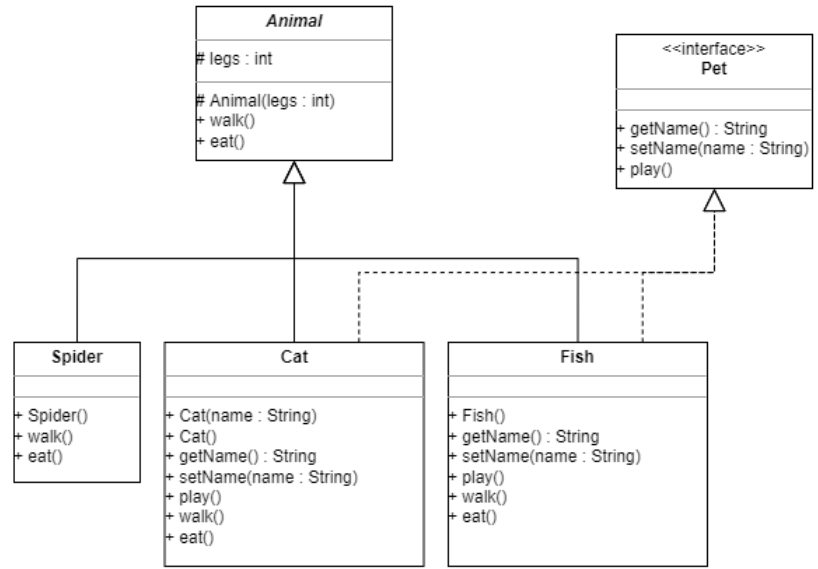
```
public interface Hockey extends Event, Sports {
  // Methods here
}
```

04

UML Interface & Abstract Class

UML Interface & Abstract Class

- Abstract class dituliskan dengan Italic seperti pada contoh Animal disamping.
- Interface dituliskan seperti pada contoh disamping.
- Garis penyambung Interface menggunakan garis putus-putus.



05

Operator Instanceof

Instance Of

Operator instanceof digunakan untuk mengecek apakah objek adalah sebuah instance dari sebuah Object type (class / subclass / interface). Biasa digunakan ketika menggunakan konsep polimorfisme.

Contoh sederhana:

```
public class TestClass{
  public static void main(String[] args) {
    TestClass test = new TestClass();
    System.out.println(test instanceof TestClass); // True
  }
}
```

Instance Of

Contoh lain ketika ada sebuah Class Cat extend Class Animal:

```
public class TestClass{
  public static void main(String[] args) {
    Cat kucing = new Cat();
    System.out.println(kucing instanceof Animal); // True
  }
}
```

06

Casting

Casting

Casting merupakan aksi ketika kita mengubah suatu tipe data menjadi tipe data yang lain.

Misal hirarki **Food → Fruit → Apple**

Downcasting: Mengubah tipe class ke yang lebih rendah hirarkinya

```
Fruit fruit = new Apple();
Apple castedApple = (Apple) fruit;
```

Apa output dari **fruit instanceof Fruit?**

Apa output dari **castedApple instanceof Fruit?**

Exercise!

Semua bisa karena terbiasa

Exercise `Test.java`

Buat abstract class 'Shape' dengan tiga method abstrak yaitu 'rectangleArea' yang memiliki dua parameter, 'squareArea' dan 'circleArea' yang masing-masing memiliki satu parameter. Parameter dalam 'rectangleArea' adalah panjang dan lebar persegi panjang, 'squareArea' adalah sisi persegi, dan 'circleArea' adalah jari-jari lingkaran. Kemudian buat kelas 'Area' yang meng-extend kelas 'Shape' untuk mencetak luas persegi panjang, persegi dan lingkaran.

Buat kelas `Test.java` untuk membuat objek dari kelas 'Area' dan panggil ketiga method tersebut.

INPUT panjang & lebar persegi panjang, sisi persegi, jari-jari lingkaran

OUTPUT luas persegi panjang, persegi, dan lingkaran

CONTOH Contoh Input & Output

```
Masukkan panjang persegi panjang: 2
Masukkan lebar persegi panjang: 3
Masukkan sisi persegi: 4
Masukkan jari-jari lingkaran: 5
```

```
Luas persegi panjang: 6.00
Luas persegi: 16.00
Luas lingkaran: 78.54
```

Assignment!

Seperti biasa, setiap selesai praktikum pasti ada tugas

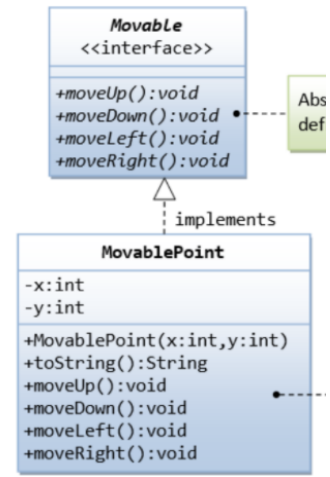
Assignment 5 Soal 1

implementasikan UML berikut dengan class Test dan perbaiki class Test tersebut:

```
class Test {
  public static void printMovablePoint(MovablePoint point) {
    System.out.println(point);
  }

  public static void main(String[] args) {
    MovablePoint point1 = new MovablePoint(0,0);
    point1.moveUp();
    point1.moveUp();
    point1.moveDown();
    point1.moveRight();
    point1.moveRight();
    point1.moveRight();
    point1.moveLeft();
    System.out.println(point1);

    Movable point2 = new MovablePoint(3,2);
    // Change line of code below to fix the problem without change declaration code!
    printMovablePoint(point2);
  }
}
```



Assignment 5 Soal 2

Buatlah tugas ke-1 tadi dengan menggunakan **abstract class** (tanpa interface).
Nama abstract class : **AbstractMovablePoint**

```
class Test {
  public static void main(String[] args) {
    MovablePoint point1 = new MovablePoint(0,0);
    point1.moveUp();
    point1.moveUp();
    point1.moveDown();
    point1.moveRight();
    point1.moveRight();
    point1.moveRight();
    point1.moveLeft();
    System.out.println(point1);
  }
}
```

Snippets 👨‍💻

Berikut snippets program materi kali ini

snippets-PBO-05

Silahkan untuk kalian mencoba menjalankan dan mempelajari snippets yang disediakan pada repository tersebut. Selamat belajar!

Teknis Pengumpulan

Pengerjaan dan pengumpulan tugas akan dilakukan di **Github Classroom**

Kelas A:

Link Tugas Kelas A

Kelas B:

Link Tugas Kelas B

Accept assignment terlebih dahulu lalu link akun Github dengan slot nama yang sesuai di Github Classroom

Teknis Pengumpulan

Format setiap file `.java` didahulukan dengan Nama, NPM, Kelas, Tanggal, dan Deskripsi

Cara menambah comment di java

```
// untuk single line
/* untuk multiple line */
```

Contoh Format

```
/*
  Nama  : Jane Doe
  NPM       : 99
  Kelas     : A
  Tanggal   : 1 September 2021
  Deskripsi : Class jawaban exercise-01 soal-01
*/
```

Deadline Pengumpulan ⌚

Kelas A:

8 Oktober 2023, 23:59 WIB

Kelas B:

10 Oktober 2023, 23:59 WIB

Waktu yang dilihat adalah waktu last commit.
Jika ada yang commit melewati deadline walaupun sudah commit sebelumnya akan dianggap telat

Terima Kasih!

Do you have any questions? Please use respective class discussion channel on Discord.

Key of success adalah kunci kesuksesan 🔥🔥🔥