

# 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

03

## Interface

Mengenal interface sebagai bentuk lain dari abstract class

05

## Operator Instanceof

Mengenal penggunaan operator instanceof

02

## Abstract Class

Mengenal class yang berbentuk abstrak

04

## UML Interface dan Abstract Class

Mengenal Interface dan class abstrak dalam bentuk UML

06

## Casting

Mengenal istilah lainnya yang terdapat dalam polimorfisme

01

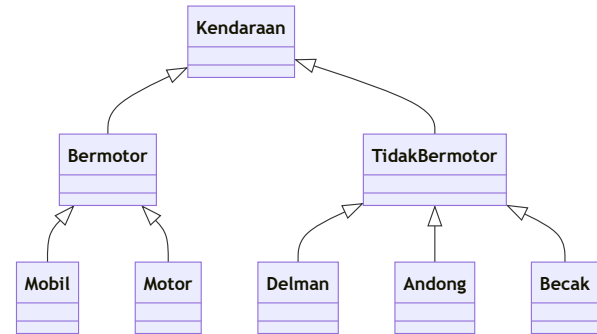
# Polimorfisme

# 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

```
from abc import ABC, abstractmethod

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

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

## C++

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

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



# 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

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

```
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  
}
```

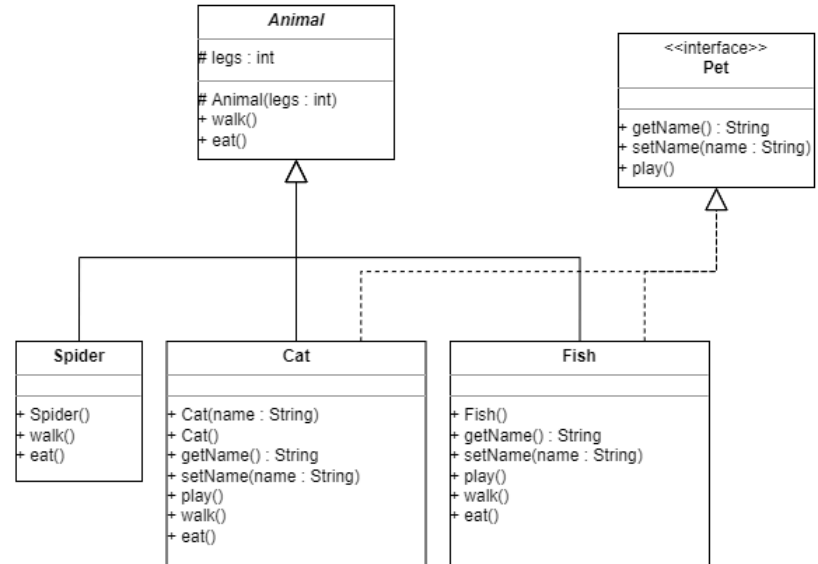
04

## 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

RaceLap.java

Buatlah 7 buah class :

- `RaceCar``, class interface dengan 1 buah method:
  - `double getLapTime();`
- `Car``, class abstract yang mengimplementasikan `RaceLap.java` dengan atribut:
  - `String name` (nama mobil)
  - `double lapTime` (lap time dalam detik)
  - abstract method `getLapTime();`
- `Mobil1, Mobil2, Mobil3, Mobil4, Mobil5``, class turunan dari `Car`:
  - Mengimplementasikan method `getLapTime()` untuk mengembalikan nilai `lapTime`.
- `RaceResult``, class dengan 1 method:
  - `showResults(Car[] cars)`
  - Menentukan mobil tercepat.
  - Mengurutkan mobil berdasarkan lap time.
  - Menampilkan hasil balapan dengan format:
    - Mobil tercepat ditampilkan dengan waktu lap.
    - Mobil selanjutnya ditampilkan dengan selisih waktu terhadap mobil sebelumnya.

Buat kelas `RaceLap.java`` untuk:

1. Input nama & waktu lap (detik) dari 5 mobil melalui terminal.
2. Membuat objek dari lima mobil tersebut.
3. Memanggil `RaceResult.showResults()` untuk menampilkan hasil perlombaan.

**INPUT** Nama mobil & lap time (detik) dari 5 mobil balap

**OUTPUT** Urutan mobil tercepat beserta selisih waktu (delta time)

Contoh Input & Output

```
Masukkan nama mobil 1: Red Bull
Masukkan lap time mobil 1 (detik): 72.53
```

```
Masukkan nama mobil 2: Ferrari
Masukkan lap time mobil 2 (detik): 73.21
```

```
Masukkan nama mobil 3: Mercedes
Masukkan lap time mobil 3 (detik): 72.88
```

```
Masukkan nama mobil 4: McLaren
Masukkan lap time mobil 4 (detik): 74.02
```

```
Masukkan nama mobil 5: Aston Martin
Masukkan lap time mobil 5 (detik): 73.77
```

**CONTOH**

```
Hasil Balapan F1:
1. Red Bull - 72.53 detik
2. Mercedes - +0.35
3. Ferrari - +0.68
4. Aston Martin - +1.24
5. McLaren - +1.49
```



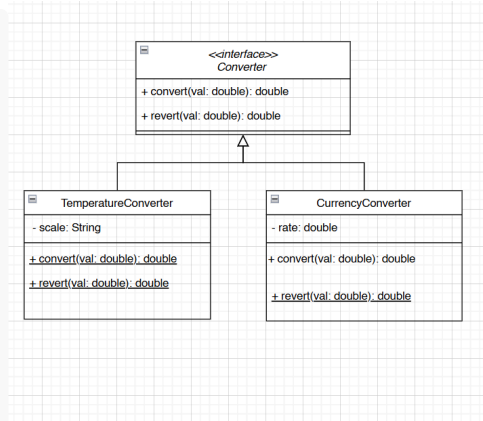
# Assignment!

Kurang seru praktikum tanpa tugas :D

# Assignment 5 Soal 1

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

```
class Test {  
    public static void main(String[] args) {  
        Converter conv1 = new TemperatureConverter();  
        System.out.println("Celsius ke Fahrenheit: " + conv1.convert(25));  
        System.out.println("Fahrenheit ke Celsius: " + conv1.revert(77));  
  
        Converter conv2 = new CurrencyConverter(15000);  
        // Change line of code below to fix the problem without change declaration code!  
        conv2.convert(100);  
    }  
}
```



Dengan ketentuan :

- `Converter` adalah class interface
- `TemperatureConverter` adalah class implement `Converter` (konversi suhu)
- `CurrencyConverter` adalah class implement `Converter` (konversi uang, kurs Rupiah-USD)

# Assignment 5 Soal 2

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

```
class Test {  
    public static void main(String[] args) {  
        TemperatureConverter conv1 = new TemperatureConverter();  
        System.out.println("Celsius ke Fahrenheit: " + conv1.convert(30));  
        System.out.println("Fahrenheit ke Celsius: " + conv1.revert(86));  
    }  
}
```

# 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:

2 September 2025, 23:59 WIB

Kelas B:

01 Oktober 2025, 23:59 WIB

**Waktu yang dilihat adalah waktu last commit.**

Jika ada yang commit melewati deadline walaupun sudah commit sebelumnya akan dianggap telat raghrrrrrrrr  
aku ketua dpa

# Terima Kasih!

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

Key of success adalah kunci kesuksesan 🔥 🔥 🔥