Praktikum Pemrograman Berbasis Objek

Pertemuan 3

Review Time 

1. Modularitas Java
2. Package
3. Array of Object
4. Access Modifier (package)
5. Enkapsulasi
6. Overloading

Materi Pertemuan 3

Gambaran Materi Pembelajaran Hari Ini

Materi Pertemuan 3

01

Inheritance

Mengenal konsep inheritance, access modifier untuk inheritance, dan keyword super

02

Method Overriding

Mengenal konsep dan penggunaan overriding

03

UML (Class Diagram)

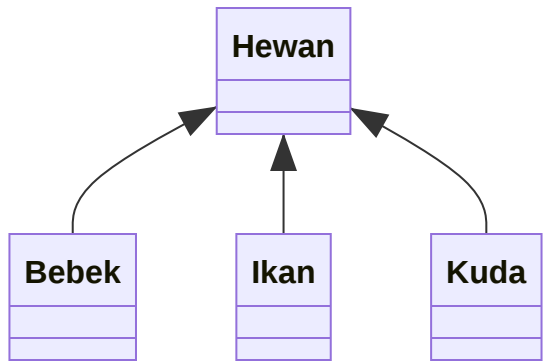
Mengenal fungsi dan implementasi UML (class diagram)

01

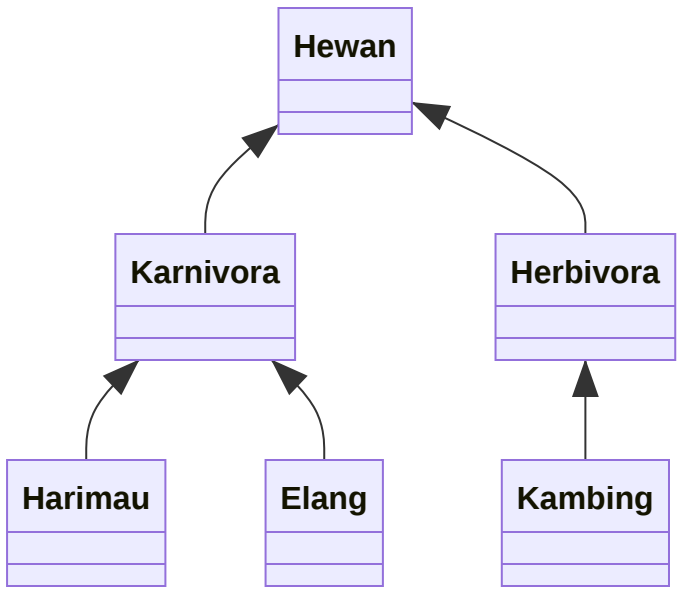
Inheritance

Inheritance

- Salah satu dari 4 pilar pada *Object Oriented Programming* (OOP), yakni *Inheritance*, *Encapsulation*, *Abstraction*, dan *Polymorphism*.
- Prinsip ini menyatakan bahwa suatu class dapat memiliki class turunan.
- Suatu class induk (*parent class*) akan dapat menurunkan **member** (properties dan methods) ke class anak (*child class*).
- Parent class dapat disebut juga **base class** atau **superclass**. Child class disebut juga **subclass**.



Inheritance (cont.)



- Java hanya mengizinkan **single inheritance**, yakni *subclass* hanya dapat memiliki **satu** *parent class*.
- Satu *parent class* dapat memiliki **banyak** *subclass* yang disebut **hierarchical inheritance**.
- Satu *subclass* dapat memiliki *subclass* lagi, disebut **multilevel inheritance**.
- Java tidak mendukung multiple inheritance, yakni satu *subclass* memiliki **lebih dari satu** *parent class*.

Inheritance (cont.)

```java
/**
 * Animal.java
 */
public class Animal {
    private int legCount;

    public Animal() { this.legCount = 2; }

    public Animal(int legCount) {
        this.legCount = legCount;
    }

    protected void setLegCount(int legCount) {
        this.legCount = legCount;
    }

    protected int getLegCount() {
        return this.legCount;
    }
}
```

```java
/**
 * Cat.java
 */
public class Cat extends Animal {
    public Cat() {
        super(4);
    }
}
```

```java
/**
 * Main.java
 */
public class Main {
    public static void main(String[] args) {
        Cat catus = new Cat();
        System.out.println(catus.getLegCount()); // 4
        catus.setLegCount(8);
        System.out.println(catus.getLegCount()); // 8
    }
}
```

Access Modifier (Inheritance)

Access modifier untuk pengaksesan member class

	class yang sama	package yang sama	subclass	class manapun
public	✅	✅	✅	✅
protected	✅	✅	✅	❌
default	✅	✅	❌	❌
private	✅	❌	❌	❌

'super' Keyword

Keyword 'super' dipakai untuk merujuk member dari parent class.

```
super.variable; // merujuk pada property milik parent class

super.function(); // merujuk pada method milik parent class

super(); // merujuk pada konstruktor parent class
```

'super' Keyword (cont.)

```java
/**
 * Animal.java
 */
public class Animal {
    public int legCount;

    public Animal() { this.legCount = 2; }
}
```

```java
/**
 * Main.java
 */
public class Main {
    public static void main(String[] args) {
        Cat catus = new Cat();
        catus.info();
    }
}
```

```java
/**
 * Cat.java
 */
public class Cat extends Animal {
    public int legCount = 4;
    public Cat() { super(); }

    public void info() {
        System.out.println(legCount); // 4
        System.out.println(this.legCount); // 4
        System.out.println(super.legCount); // 2
    }
}
```

Constructor pada Inheritance

- Hanya member class *properties* dan *methods* yang diwariskan, **konstruktor tidak diwariskan**.
- Ketika object child dibuat (konstruktornya dieksekusi), maka konstruktor parent class dijalankan terlebih dahulu kemudian menyelesaikan konstruktor child.

1. Coba compile class Cat di samping, apa yang terjadi?
2. Kemudian pindahkan super() sebelum `this.legCount = 8` pada class Cat di samping, compile ulang, dan lihat apa yang terjadi.

```java
/**
 * Animal.java
 */
public class Animal { }
```

```java
/**
 * Cat.java
 */
public class Cat extends Animal {
    private int legCount = 4;
    public Cat() {
        this.legCount = 8;
        super();
    }

    public void info() {
        System.out.println(legCount);
    }
}
```

02

Method Overriding

Method Overriding

Menulis kembali method dari superclass pada subclassnya untuk memperoleh implementasi yang lebih spesifik.

```java
/**
 * Animal.java
 */
class Animal {
    public void walk() {
        System.out.println(getClass().getSimpleName() + " walks!");
    }
}
```

```java
/**
 * Cat.java
 */
class Cat extends Animal {
    @Override
    public void walk() {
        super.walk();
        System.out.println("This cute " + getClass().getSimpleName() + " walks!");
    }
}
```

Method Overriding (Other)

Python

```python
class Animal:
   def walk(self):
       print(self.__name__ + " walks!")

class Cat(Animal):
   def walk(self):
       print("This cute " + self.__name__ + " walks!")
```

C++

```cpp
class Animal {
  public:
    void walk() {
      cout << "Animal walks!";
    }
};

class Cat: public Animal {
  public:
    void walk() {
      cout << "This cute cat walks!";
    }
};
```

Method Overriding (cont.)

Syarat atau aturan overriding meliputi:

- Nama method child dan parent harus sama.
- Daftar parameter method child dan parent harus sama.
- Return type method child dan parent harus sama.
- Access modifier pada method overriding di subclass **tidak boleh lebih ketat** dibandingkan dengan access modifier method pada parent class.

Method overriding merupakan contoh dari *runtime polymorphism*.

03

UML

UML (*Unified Modeling Language*)

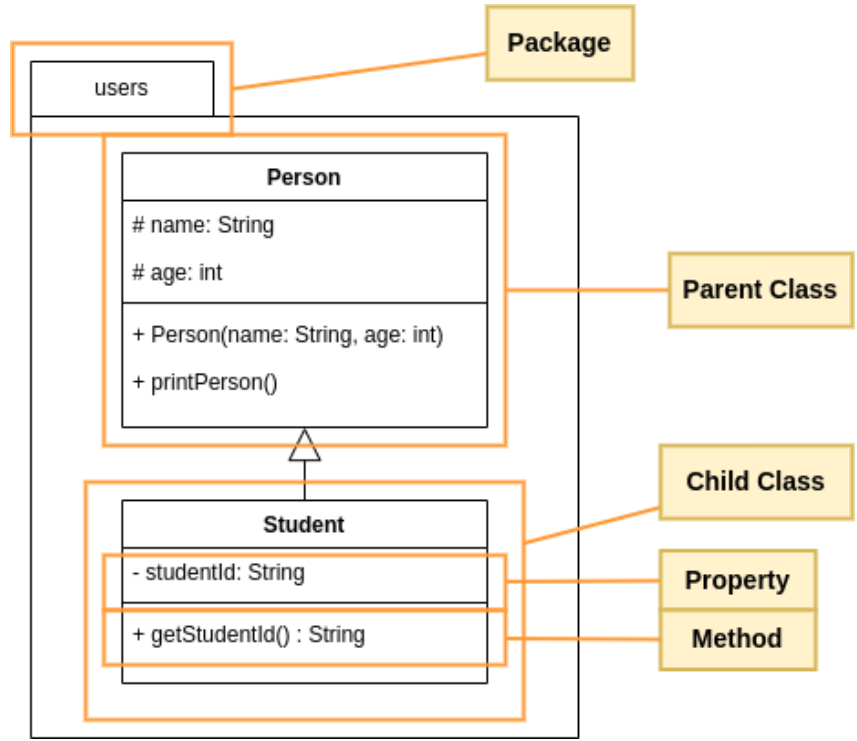
- "Bahasa" yg telah menjadi standar dalam industri untuk visualisasi, perancangan, dan pendokumentasian aplikasi.
- Model UML digunakan untuk semua jenis aplikasi, tanpa batasan hardware, sistem operasi, jaringan, serta ditulis dalam bahasa pemrograman apapun.

UML mendefinisikan diagram-diagram sebagai berikut:

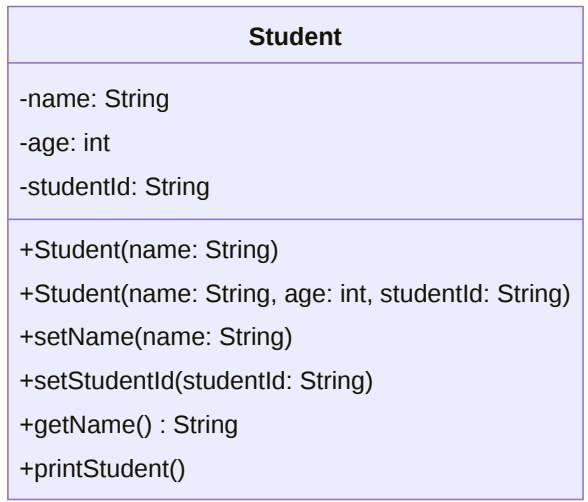
- use case diagram
- **class diagram**
- statechart diagram
- activity diagram
- sequence diagram
- collaboration diagram
- component diagram
- deployment diagram

Class Diagram

Class Diagram adalah salah satu jenis diagram di UML untuk memetakan struktur sistem tertentu dengan memodelkan kelas, atribut, operasi serta hubungan antar objek.



Class Diagram (cont.)

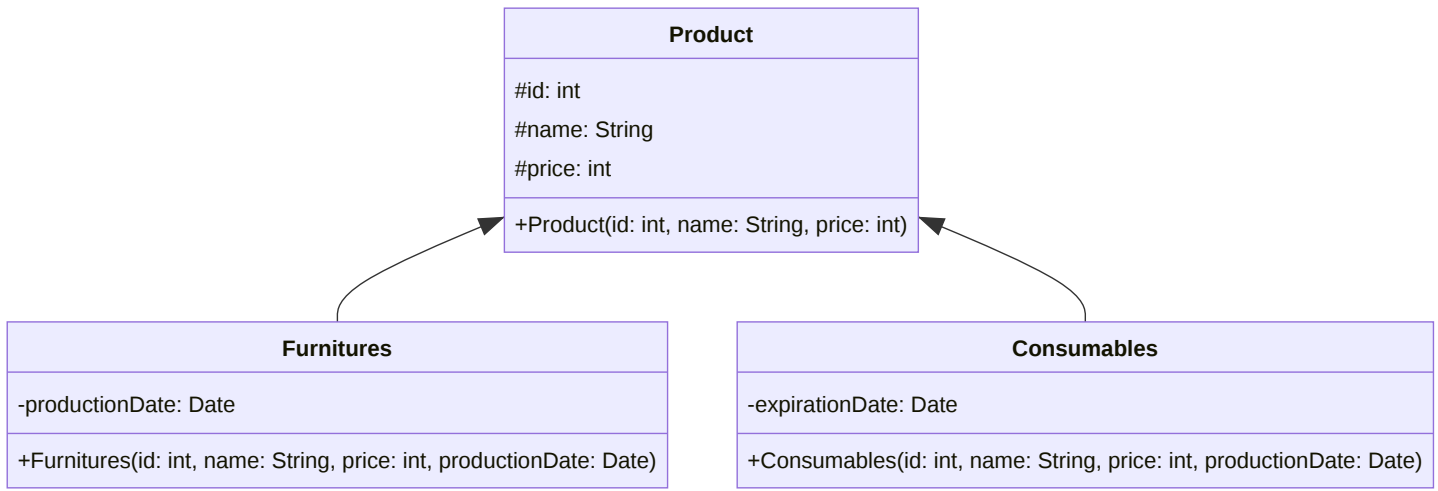


Visibilitas

Marker	Visibility
+	public
-	private
#	protected
~	package

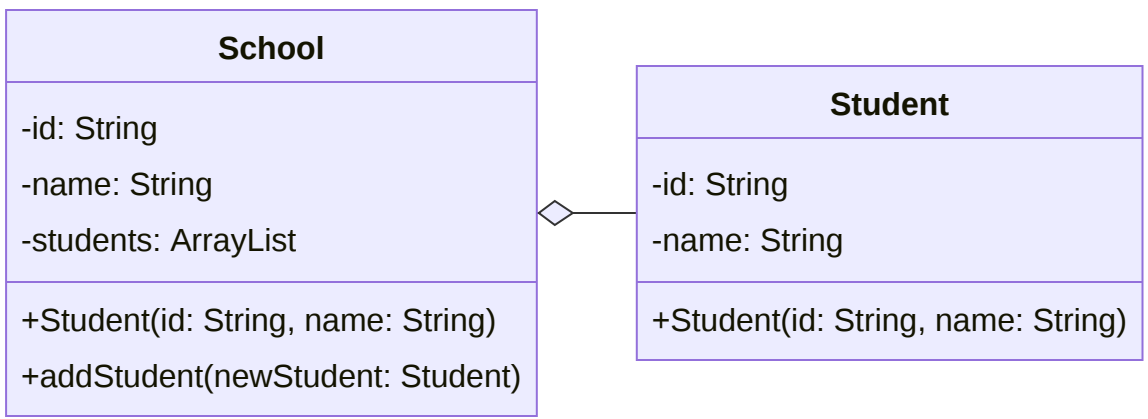
Hubungan Antar Kelas

Inheritance/ Pewarisan yakni membuat sebuah subclass dari class yang sudah ada. Hubungan ini disebut juga hubungan *spesialisasi* dan berkebalikan dengan hubungan *generalisasi*.



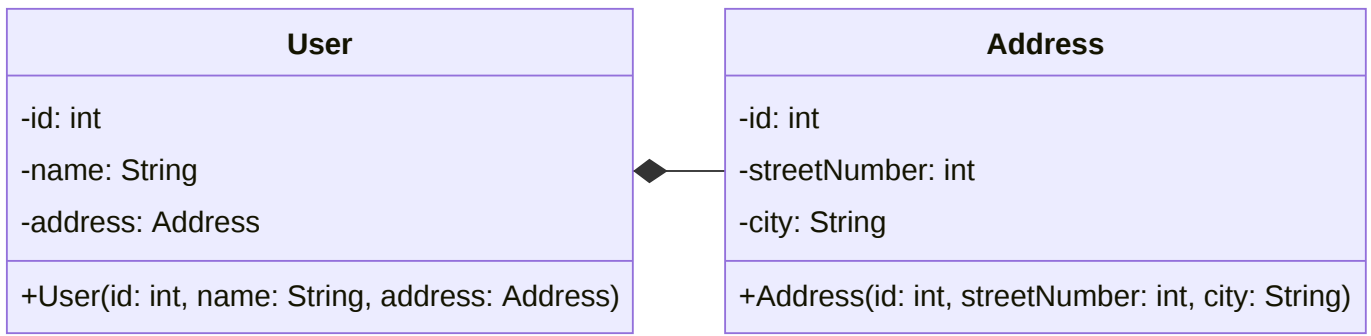
Hubungan Antar Kelas (cont.)

Aggregation merupakan hubungan antara class (entity) dengan dependency yang memiliki lifespan berbeda. Jika 1 class telah mati, tetapi dependency **tidak akan ikut mati.**



Hubungan Antar Kelas (cont.)

Composition merupakan hubungan antara class (entity) dengan dependency yang memiliki lifespan sama. Jika 1 class telah mati, maka dependency pun **akan ikut mati**.



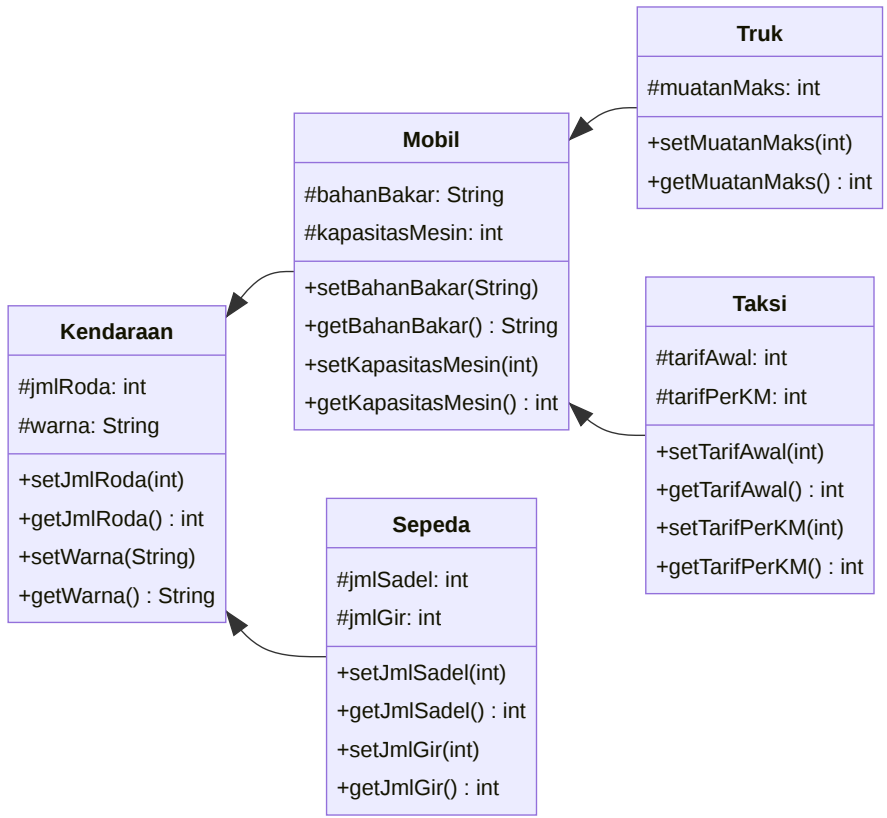
Mengimplementasikan UML Diagram

Perhatikan tabel di bawah, apa yang dapat kamu simpulkan?

Objek	Jml Roda	Warna	Bhn Bakar	Kaps Mesin	Muatan Maks	Tarif Awal	Tarif per KM	Jml Sadel	Jml Gir
truk1	4	kuning	solar	1500	1000	-	-	-	-
truk2	6	merah	solar	2000	5000	-	-	-	-
taksi1	4	oranye	bensin	1500	-	10000	5000	-	-
taksi2	4	biru	bensin	1300	-	7000	3500	-	-
sepeda1	3	hitam	-	-	-	-	-	1	2
sepeda2	2	putih	-	-	-	-	-	2	5

Mengimplementasikan UML Diagram

Class diagram dari tabel sebelumnya sebagai berikut.



Exercise!

Semua bisa karena terbiasa

Exercise `Test.java`

Buatlah kelas yang dibutuhkan dan kelas `Test.java` yang membuat obyek-obyek serta menyimpan nilai variabel seperti pada tabel/ class diagram slide sebelumnya. Tampilkan data seluruh objek!

INPUT -

OUTPUT data objek seperti contoh di bawah

CONTOH

```
truk1
jmlRoda: 4
warna: kuning
bahanBakar: solar
kapasitasMesin: 1500
muatanMask: 1000

...
```

Assignment!

Seperti biasa akan setiap selesai praktikum pasti ada tugas

Assignment 3 Soal 1

Buat UML Class Diagram dengan **minimal 3 class** (1 Parent dan 2 Child). Gunakan konsep enkapsulasi yang telah dipelajari, terapkan konsep *overriding* minimal dalam satu class child dan berikan panah untuk setiap hubungan antar kelasnya.

Kriteria UML dibebaskan (selain UML kendaraan dari slide exercise). Elemen yang perlu ada adalah:

- Nama class
- Nama dan tipe variabel
- Nama, parameter, dan return type method
- Access Modifier/ Marker

Tools yang digunakan dibebaskan, jika kebingungan dapat menggunakan website diagrams.net dengan output berupa file bernama `UML.png`

Assignment 3 Soal 2

Setelah membuat UML, silahkan implementasikan kedalam bahasa pemrograman Java. Buat filenya sesuai dengan class yang ada dalam UML.

Berikut adalah contoh isi foldernya:

```
git-repo-kalian
├── ContohChild1.java
├── ContohChild2.java
├── ContohParent.java
├── Main.java
└── UML.png
```

`Main.java` digunakan sebagai program untuk memeriksa instansiasi objek setiap class yang dibuat.

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:

26 September 2022, 23:59 WIB

Kelas B:

27 September 2022, 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.

Selalu semangat menggapai impian!! 🔥🔥🔥