Object Oriented Programming

Abstract Class, Abstract Method, Inheritence and Polymorphism

Abstract Class and Method

```
abstract class ProgrammingLanguage {
    abstract void describe();

    public void print() {
        System.out.println("Coding is fun!");
    }
}

public class Java extends ProgrammingLanguage {
    @Override
    public void describe() {
        System.out.println("Java use OOP concept.");
    }
}
```

Cont...

Abstract Class :

- Must be declared with `abstract` keyword
- Can have abstract and non-abstract method
- Can not be instantiated
- Can have *constructor* and static method also
- Can have final methods which will force the subclass not to change the body of the method

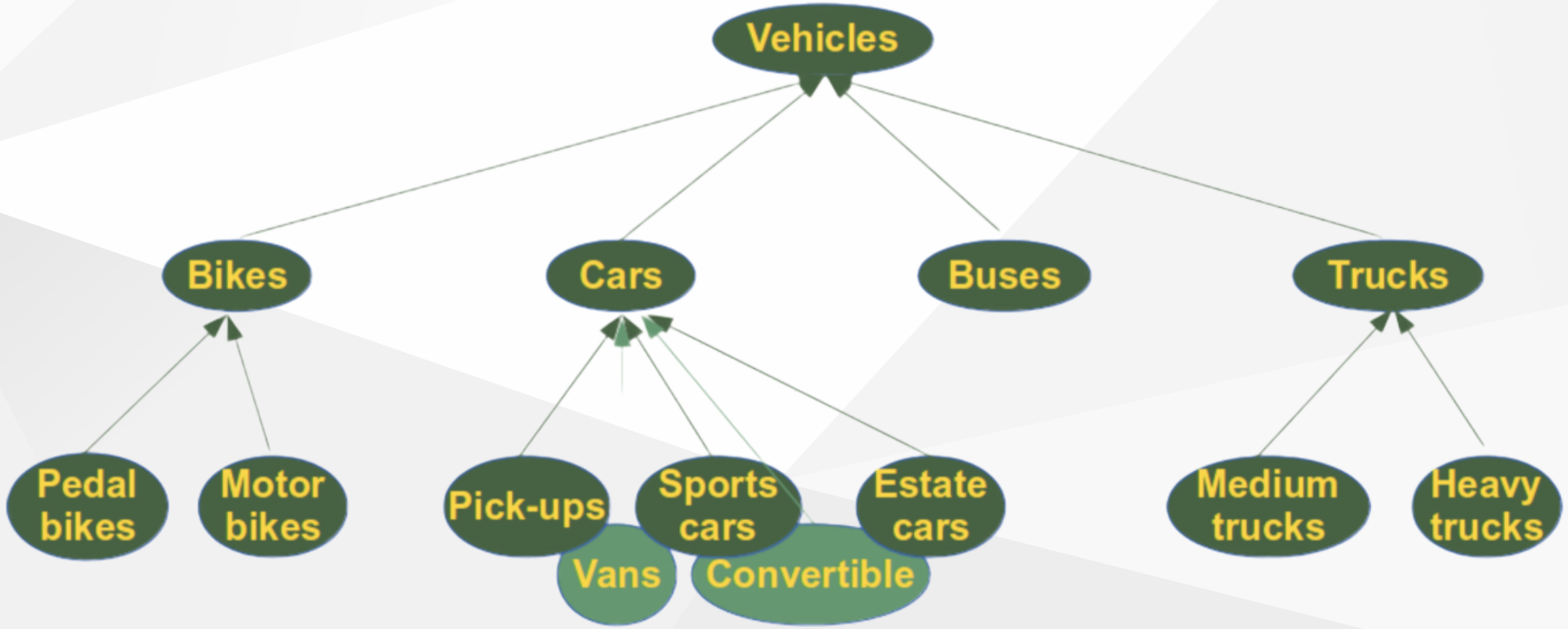
Inheritence

Object Fact :

- Object are often very similar. They share common logic.
- But, they're not entirely the same

> ***What if we put common logic in one class, and unique logic of every object in their own class? Is that save your life from creating bunch of code in one class***

Inheritence Analogy



Inheritence Example

```
public class Human {
    private String name;

    public Human(String name) {
        this.name = name;
    }
    // Setter getter block
}

public class Employee {
    private String nik;

    public Employee(String name, String nik) {
        super(name);
        this.nik = nik;
    }
}
```

Cont...

Pre save word SUPER

SUPER is used to returns a proxy object that delegates method all to a parent or sibling class of type.

Method overriding is used to overriding parent method

Polymorphism

- **Poly** = Many, **Morphism** = Form, **Polymorphism** is ability objects of different types to respond to functions of the same name
- **User** does not have to know the exact object in advance
- **The behavior** of object can be implemented at runtime

Polymorphism Example

```
public interface Vehicle {
    void topSpeed();
}

public class Lamborghini implements Vehicle {
    @Override
    public void topSpeed() {
        System.out.println("350 km/h");
    }
}

public class Fuso implements Vehicle {
    @Override
    public void topSpeed() {
        System.out.println("80 km/h");
    }
}
```

Cont...

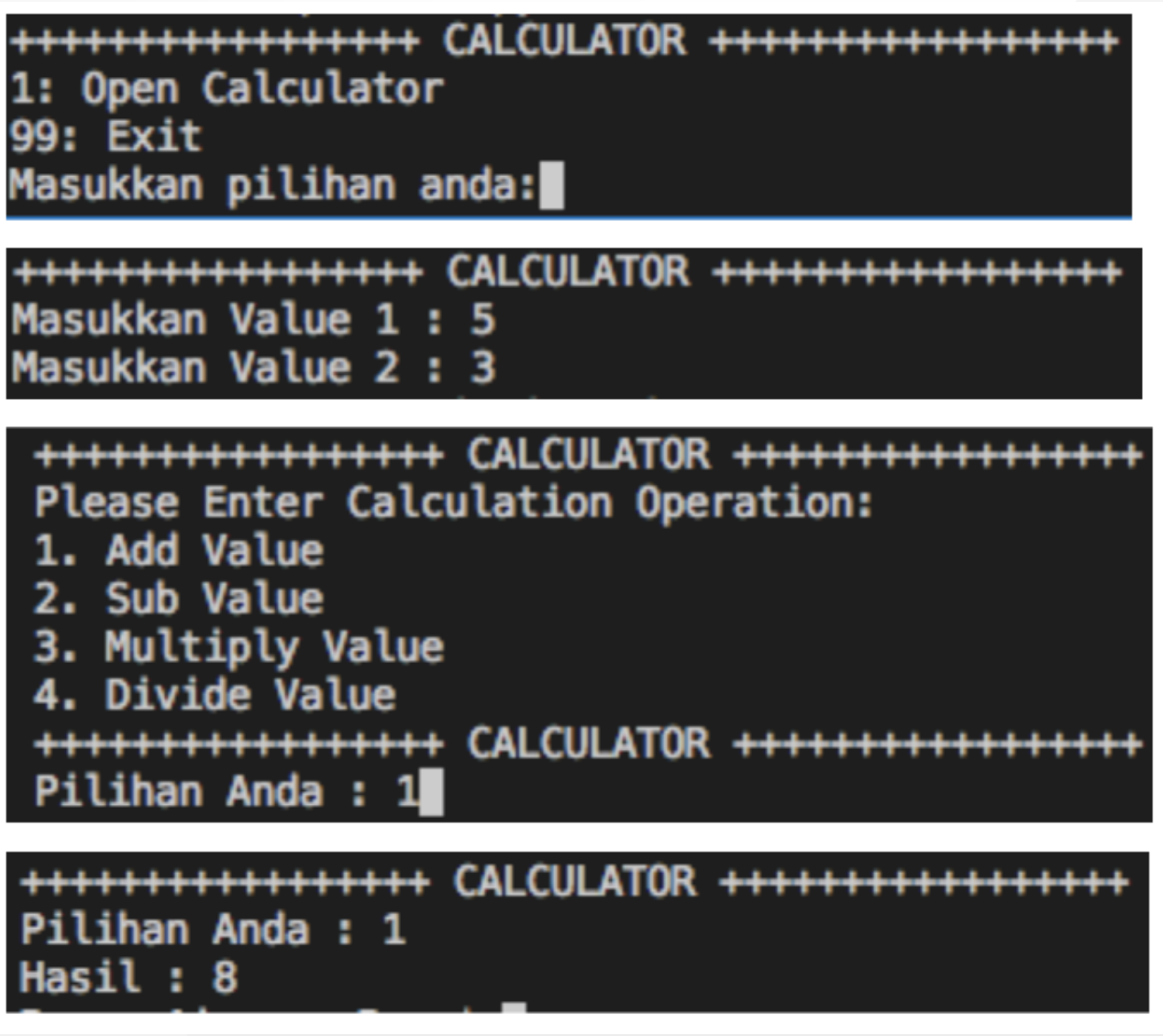
```
public class Avanza implements Vehicle {
    @Override
    public void topSpeed() {
        System.out.println("150 km/h");
    }
}

public class Skyline implements Vehicle {
    @Override
    public void topSpeed() {
        System.out.println("320 km/h");
    }
}
```

Task Abstract Class & Abstract Method

Create a simple calculator application with addition, subtraction, division and multiplication functions.

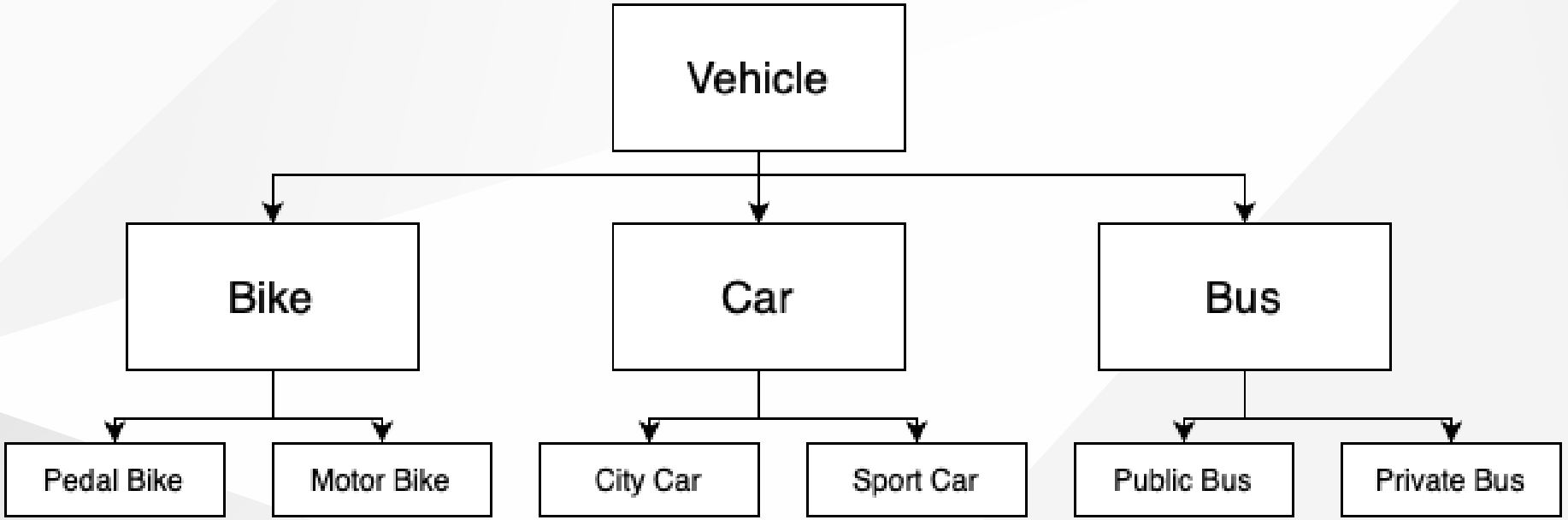
Take advantage of the `input()` function in Java to `enter the desired 2 numbers` and **1 number in the form of an operation choice**.

Print the result of the operation at the end of the section like demo on the next slide.


++++++++++++++++++ CALCULATOR ++++++++++++++++++
1: Open Calculator
99: Exit
Masukkan pilihan anda:
++++++++++++++++++ CALCULATOR ++++++++++++++++++
Masukkan Value 1 : 5
Masukkan Value 2 : 3
++++++++++++++++++ CALCULATOR ++++++++++++++++++
Please Enter Calculation Operation:
1. Add Value
2. Sub Value
3. Multiply Value
4. Divide Value
++++++++++++++++++ CALCULATOR ++++++++++++++++++
Pilihan Anda : 1
++++++++++++++++++ CALCULATOR ++++++++++++++++++
Pilihan Anda : 1
Hasil : 8

Task Inheritence & Polymorphism (Vehicles)

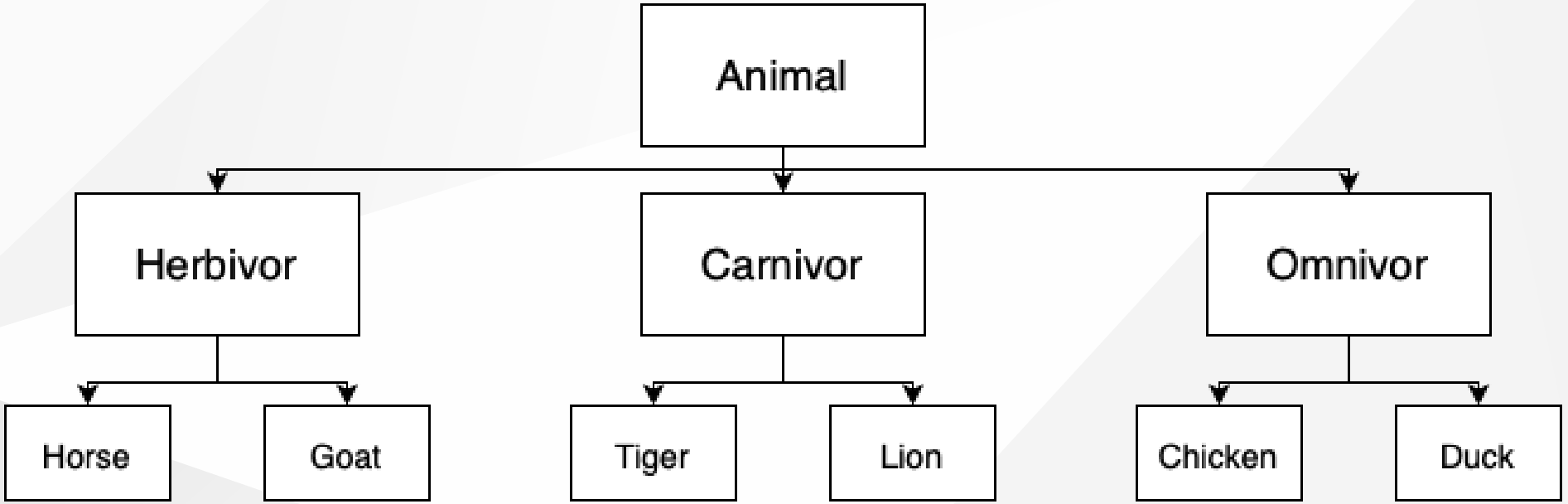
- `Vehicle` is a parent of all existing classes. And have property:
  - `name` : for object name
  - `isUseEngine` : flag object if has engine or not
- `Bike`, `Car` and `Bus` is a child from `Vehicle`
- Class `Bike`
  - `wheelCount` : number of wheels owned
- Class `Car`
  - `wheelCount`
  - `engineType` : type of engine
- Class `Bus`
  - `wheelCount`
  - `isPrivateBus` : flag bus is private or public
- Every class have method `identifyMySelf()` that **overrides** from `Vehicle` to print out like demo on the next slide




```
/media/anton/aa5d38f0-c9e2-4761-9df6-fd94422f4892/ATA/ata-venv/bin/python /media/anton/aa5d38f0-c9e2-4761-9df6-fd944
Hi I'm Parent of All Vehicles, My Name is Gerobak, My Engine Status is 'no engine'

Hi I'm Bike , My Name is Pedal Bikes, My Engine Status is 'no engine', I have 2 Wheel(s)
Hi I'm Bike , My Name is Motor Bikes, My Engine Status is 'with engine', I have 2 Wheel(s)

Hi I'm Car , My Name is Sport Cars, My Engine Status is 'with engine', I have 4 Wheel(s), My Engine Type = V8
Hi I'm Car , My Name is Pickup Cars, My Engine Status is 'with engine', I have 4 Wheel(s), My Engine Type = Solar

Hi I'm Bus [Public Bus] , My Name is Trans Jakarta, My Engine Status is 'with engine', I have 4 Wheel(s)
Hi I'm Bus [Private engine] , My Name is School Bus, My Engine Status is 'with engine', I have 4 Wheel(s)
```

Task Inheritence & Polymorphism (Animal)

- `Animal` is a parent of all existing classes. And have property:
  - `name` : object name
  - `foodType` : type of food
  - `isSharpTeeth` : flag teeth is sharp or blunt
- `Herbivor`, `Carnivor` and `Omnivor` is a child from `Animal`
- Class `Herbivor`
  - Should eat plants
  - Should have blunt teeth
- Class `Carnivor`
  - Should eat meat
  - Should have sharp teeth
- Every class have method `identifyMySelf()` that **overrides** from `Animal` to print out like demo on the next slide




```
Hi I'm Parent of All Animal, My Name is Binatang

Hi I'm Herbivor , My Name is Kambing, My Food is 'tumbuhan', I have tumpul teeth
Hi I'm Carnivor , My Name is Singa, My Food is 'daging', I have tajam teeth

Hi I'm Omnivor , My Name is Ayam, My Food is 'semua', I have tajam dan tumpul teeth
```