# TUGAS : ABSTRACT CLASS & INTERFACE

## PERTEMUAN 7

## PEMROGRAMAN BERORIENTASI OBJEK (TE)


Disusun oleh :

Muhammad Wildan Gumilang (231511087)

**PROGRAM DIPLOMA III TEKNIK INFORMATIKA**

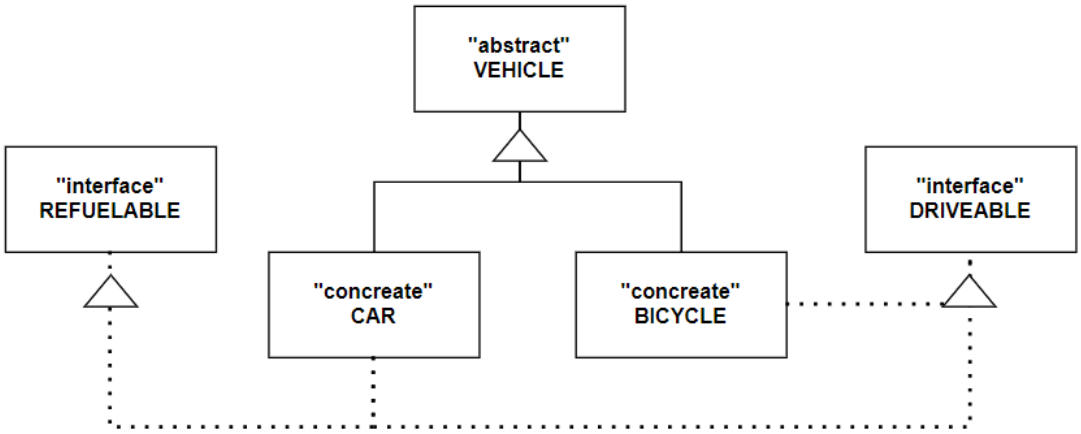
**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

# DAFTAR ISI

# Program yang mengimplementasikan konsep abstract class, interface dan implementation

## Class Diagram




## Abstract Class

**Vehicle.Java**

```java
public abstract class Vehicle {
    protected String brand;
    protected int fuel;

    public Vehicle(String brand, int fuel) {
        this.brand = brand;
        this.fuel = fuel;
    }

    public void showFuelLevel() {
        System.out.println(brand + " has " + fuel + " liters of fuel.");
    }
}
```

Abstract class adalah class yang tidak bisa diinstansiasi secara langsung, melainkan harus di-extend oleh class lain. Class ini dapat berisi abstract method (method yang tidak memiliki implementasi) dan juga concrete method (method yang memiliki implementasi).

Dalam kasus yang saya gunakan, Vehicle adalah abstract class yang mendefinisikan atribut umum dari kendaraan, seperti brand dan fuel. Selain itu, Vehicle juga memiliki concrete method showFuelLevel(), untuk menampilkan sisa bahan bakar.

## Interface

**Refuelable.java**

```
public interface Refuelable {
    void refuel(int liters);
}
```

```
public interface Driveable {
    void drive();
}
```

Interface adalah kontrak yang mendefinisikan method-method yang harus diimplementasikan oleh class yang menggunakannya. Dalam Kasus ini interface mendefinisikanmethod refuel() yang berfungsi untuk mengisi bahan bakar dan drive() yang berfungsi untuk cara berkendara. Yang nantinya class Car mengimplementasikan kedua interface ini, sedangkan class Bicycle hanya mengimplementasikan interface Driveable karena sepeda tidak membutuhkan bahan bakar.

## Implementation

Sebuah subclass yang mewarisi abstract class harus memberikan implementasi dari method abstract yang ada di dalam class tersebut. Begitu juga dengan interface; class yang mengimplementasikan interface wajib memberikan implementasi dari method yang didefinisikan dalam interface tersebut.

Class Car adalah implementasi konkret dari abstract class Vehicle dan juga interface Refuelable. Class ini mengimplementasikan interface Driveable yang mengurangi bahan bakar setiap kali mobil dikendarai.

```
@Override
    public void drive() {
        if (fuel > 0) {
            fuel -= 10;
            System.out.println(brand + " is driving.");
        } else {
            System.out.println(brand + " has no fuel to drive.");
        }
    }
```

Class Bicycle juga adalah subclass dari Vehicle, tetapi sepeda tidak membutuhkan bahan bakar sehingga tidak mengimplementasikan interface Refuelable. Bicycle ini hanya mengimplementasikan interface Driveable.

**Bicyle.java**

```
public class Bicycle extends Vehicle implements Driveable {

    public Bicycle(String brand) {
        super(brand, 0);
    }

    @Override
    public void drive() {
        System.out.println(brand + " is being pedaled.");
    }
}
```

Pada class Main, dibuat objek dari Car dan Bicycle, dan menggunakan method-method yang diimplementasikan di masing-masing class.

**Main.java**

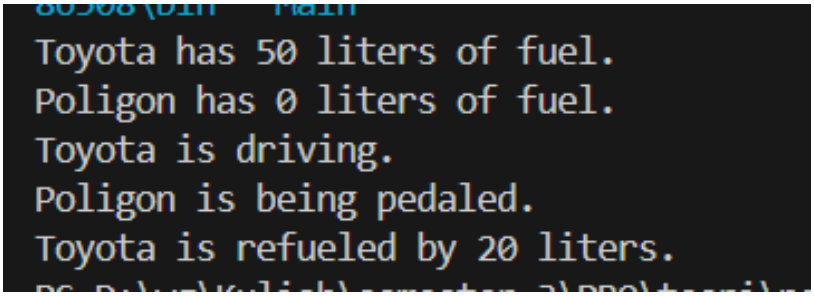
```
public class Main {
    public static void main(String[] args) {
        Vehicle car = new Car("Toyota", 50);
        Vehicle bike = new Bicycle("Poligon");

        car.showFuelLevel();
        bike.showFuelLevel();

        ((Driveable) car).drive();
        ((Driveable) bike).drive();
        car.showFuelLevel();

        ((Refuelable) car).refuel(20);
        car.showFuelLevel();
    }
}
```

**Output :**




# Kesimpulan

Jadi, dari kasus ini dapat disimpulkan bahwa :

1. Abstract class (Vehicle) adalah kelas dasar yang menyediakan kerangka umum untuk kelas lain.
2. Interface (Refuelable dan Driveable) adalah kontrak yang memaksa kelas yang menggunakannya, seperti Car, untuk mengimplementasikan method refuel() dan Bicycle, untuk mengimplementasikan drive(). Interface tidak berisi detail, hanya menjelaskan bahwa method tersebut harus ada di kelas yang menggunakannya.
3. Implementation adalah bagaimana kelas turunan seperti Car dan Bicycle memberikan implementasi konkret dari method-method yang dijelaskan oleh abstract class dan interface. Car mengimplementasikan method drive() dari Driveable dan refuel() dari Refuelable. Sementara Bicycle hanya mengimplementasikan drive() dari Driveable.