VIDYASHILP UNIVERSITY

Lab Worksheet 7
Object Oriented Programming

19th March 2025

Objectives

Abstraction

- Abstract Class
- Interface
- Multiple Inheritance

Case Study 1: Vehicle that Fly, Sail and Autonomous.

```
abstract class Vehicle {
    String model;

    Vehicle(String model) {
        this.model = model;
    }

    void startEngine() {
        System.out.println(model + " engine started.");
    }

    void stopEngine() {
        System.out.println(model + " engine stopped.");
    }

        abstract void move();
}

interface Flyable {
    void fly();
}

interface Sailable {
    void sail();
}

interface Autonomous {
    void autoPilot();
}

class ElectricCar extends Vehicle implements Autonomous {
    ElectricCar(String model) {
        super(model);
```

```
    }

    @Override
    void move() {
        System.out.println(model + " is driving on the road.");
    }

    @Override
    public void autoPilot() {
        System.out.println(model + " is driving autonomously.");
    }
}

class Drone extends Vehicle implements Flyable, Autonomous {
    Drone(String model) {
        super(model);
    }

    @Override
    void move() {
        System.out.println(model + " is flying in the air.");
    }

    @Override
    public void fly() {
        System.out.println(model + " is taking off.");
    }

    @Override
    public void autoPilot() {
        System.out.println(model + " is flying autonomously.");
    }
}

class Boat extends Vehicle implements Sailable {
    Boat(String model) {
        super(model);
    }

    @Override
    void move() {
        System.out.println(model + " is moving on water.");
    }

    @Override
    public void sail() {
        System.out.println(model + " is sailing.");
    }
}

public class jSmartVehicleSystem {
    public static void main(String[] args) {
```

```
        ElectricCar tesla = new ElectricCar("Tesla Model X");
        Drone dji = new Drone("DJI Phantom");
        Boat yacht = new Boat("Luxury Yacht");

        System.out.println("\n--- Electric Car ---");
        tesla.startEngine();
        tesla.move();
        tesla.autoPilot();
        tesla.stopEngine();

        System.out.println("\n--- Drone ---");
        dji.startEngine();
        dji.move();
        dji.fly();
        dji.autoPilot();
        dji.stopEngine();

        System.out.println("\n--- Boat ---");
        yacht.startEngine();
        yacht.move();
        yacht.sail();
        yacht.stopEngine();
    }
}
```

Case Study 2: Second-Hand Vehicle Showroom

The **Second-Hand Vehicle Showroom System** is designed to manage the buying, selling, registration, servicing, and payment processes of second-hand vehicles. The system will differentiate between **Petrol, Diesel, and Electric** vehicles and demonstrate **abstract classes, interfaces, and multiple inheritance**.

The system consists of different **vehicles** that can be purchased, sold, serviced, and paid for using multiple payment methods. It follows an **inheritance-based structure** to classify vehicles and their behaviors.

User Functions

- **Register Vehicle**: Users can register their second-hand vehicle in the showroom system.
- **Purchase Vehicle**: Users can buy a second-hand vehicle.
- **Sell Vehicle**: Users can sell their registered vehicle.
- **Service Vehicle**: Vehicles can be serviced.
- **Payment Processing**: Payments can be made through different methods like **Card, UPI, Cash, or Net Banking**.

Vehicle Types

- **Petrol Vehicle**: Requires fuel-based servicing and regular oil changes.
- **Diesel Vehicle**: Requires fuel-based servicing with a focus on fuel injectors.
- **Electric Vehicle**: Requires battery servicing and software updates.

Abstraction

- **Abstract Class: Vehicle**
  - Common properties: brand, model, price.
  - Common method: registerVehicle().
  - Common method getDetails();
  - Abstract method: sell().
    Selling of different type of vehicle has different criteria. Petrol Vehicle will have 60% depreciation, Diesel vehicle will have 70% depreciation. Electric Vehicle will have 80% Depreciation.
- **Interface: ForPurchase**
  - Method: purchase().
    Petrol vehicle will have no discount nor subsidy, the road tax will be 18% of the vehicle cost.
    Diesel vehicle can be Personal or Commercial vehicle type. The road tax for personal vehicle is 18% and for Commercial Vehicle is 22%.
    Electric vehicles will get subsidy of 5% and zero road tax.
- **Interface: ForService**
  - Method: service().
    Petrol Vehicle will have following service: Engine oil & spark plug check.
    Diesel Vehicle will have following service: Engine oil, Fuel injector & turbo maintenance.
    Electric Vehicle will have following service: Battery check & software update
- **Interface: PaymentMethod**
  - Method: pay(String method).
    Petrol Vehicle can be purchased only by Cash and Card payments.
    Diesel Vehicle can be purchased only by Cash.
    Electric Vehicle can be purchased using only by UPI payment method.

Complete the following classes:

- **class DieselVehicle** extends **Vehicle** implements **ForPurchase, ForService, PaymentMethod**

- **class ElectricVehicle** extends **Vehicle** implements **ForPurchase, ForService, PaymentMethod**

- **class PetrolVehicle** extends Vehicle implements **ForPurchase, ForService, PaymentMethod**

Create appropriate objects such as Scorpio, a Diesel vehicle, Alto, a Petrol Vehicle, and MGComet, an Electric Vehicle.