



**S. B. JAIN INSTITUTE OF TECHNOLOGY,  
MANAGEMENT & RESEARCH, NAGPUR.**

**Practical No. 9(Post Lab)**

**Aim:** Write a Java program to create a vehicle class hierarchy. The base class should be Vehicle, with subclasses Truck, Car and Motorcycle. Each subclass should have properties such as make, model, year, and fuel type. Implement methods for calculating fuel efficiency, distance travelled, and maximum speed.

**Name of Student:** Mohit Jamaiwar

**Roll No.:** CS23002

**Semester/Year:** IV/ 2nd

**Academic Session:** 2024-25

**Date of Performance:**

**Date of Submission:**

## ***Object Oriented Programming (N-PCCCS401P)***

**AIM:** Write a Java program to create a vehicle class hierarchy. The base class should be Vehicle, with subclasses Truck, Car and Motorcycle. Each subclass should have properties such as make, model, year, and fuel type. Implement methods for calculating fuel efficiency, distance travelled, and maximum speed.

### **OBJECTIVES:**

1. Establish a Vehicle Class Hierarchy:  
Create a base Vehicle class and subclasses (Car, Truck, Motorcycle) to represent different types of vehicles with shared and specific properties.
2. Implement Core Vehicle Properties:  
Include common attributes for vehicles such as make, model, year, and fuelType to ensure each subclass has a basic identity.
3. Override Methods for Specific Behaviors:  
Implement methods like calculateFuelEfficiency() and getMaxSpeed() in the base class and override them in the subclasses to provide specific behavior for different types of vehicles.
4. Simplified Code Structure:  
Present the simplest possible version of the code that still adheres to object-oriented principles and provides basic functionality for vehicle efficiency and speed.
5. Efficient Output:  
Display basic output for each vehicle type, demonstrating that the overridden methods for fuel efficiency work for each subclass.

### **THEORY:**

The program demonstrates fundamental Object-Oriented Programming (OOP) concepts in Java, which include classes, inheritance, method overriding, and polymorphism. Here's an explanation of the key topics and concepts used:

#### **1. Class and Object**

- **Class:** A class is a blueprint for creating objects in Java. It defines the properties (attributes) and behaviors (methods) that objects of the class will have.
- **Object:** An object is an instance of a class. In this program, objects like car, truck, and motorcycle are instances of the classes Car, Truck, and Motorcycle.

In the program, Vehicle is a class that serves as a template for other classes (Car, Truck, Motorcycle), and each vehicle type is instantiated as an object.

#### **2. Inheritance**

- Inheritance is a key feature of OOP that allows a new class to inherit properties and behaviors from an existing class. The new class is called a subclass (or derived class), and the class being inherited from is called the superclass (or base class).

In the program:

- Car, Truck, and Motorcycle inherit from the base class Vehicle.
- This allows Car, Truck, and Motorcycle to inherit common attributes like make, model, year, and fuelType from Vehicle.

#### **3. Method Overriding**

- Method Overriding occurs when a subclass provides its own implementation of a method that

## ***Object Oriented Programming (N-PCCCS401P)***

is already defined in its superclass. This allows for different behaviors for the same method in different classes.

In the program:

- The method `calculateFuelEfficiency()` is defined in the base class `Vehicle` but is overridden in each subclass (`Car`, `Truck`, `Motorcycle`) to provide specific implementations for each vehicle type.

For example:

- The fuel efficiency for a car is 15 km/l, for a truck it is 8 km/l, and for a motorcycle, it is 40 km/l, even though the method name is the same in all classes.

### **4. Polymorphism**

- Polymorphism is an OOP concept that allows a single method or function to take on many forms. In Java, polymorphism is typically achieved through method overriding and interfaces.

In the program:

- Polymorphism is demonstrated through the overridden `calculateFuelEfficiency()` method. Despite calling the method on different objects (`car`, `truck`, `motorcycle`), the correct version of the method is invoked based on the object's actual type (whether it's a `Car`, `Truck`, or `Motorcycle`).

### **5. Encapsulation**

- Encapsulation refers to bundling the data (variables) and methods (functions) that operate on the data into a single unit or class, and restricting access to some of the object's components. In Java, we use access modifiers like `private`, `public`, and `protected` to achieve encapsulation.

In the program:

- The `Vehicle` class encapsulates common properties (`make`, `model`, `year`, `fuelType`) and methods (`calculateFuelEfficiency()`, `getMaxSpeed()`), which are then reused and modified in subclasses.

### **6. Constructors**

- A constructor is a special type of method that is used to initialize objects. The constructor has the same name as the class and is invoked when a new object of the class is created.

In the program:

- Each class (`Vehicle`, `Car`, `Truck`, `Motorcycle`) has a constructor. The constructor is used to set values for the properties like `make`, `model`, `year`, and `fuelType` when an object is created.

For example:

```
public Car(String make, String model, int year, String fuelType) {  
    super(make, model, year, fuelType);  
}
```

Here, the `Car` constructor calls the constructor of the `Vehicle` superclass using `super()` to initialize inherited properties.

### **7. Access Modifiers**

- Access modifiers determine the visibility and accessibility of classes, methods, and variables. In this program, `public` is used, which means the class, method, or variable is accessible from any other class.

In the program:

- Methods like `calculateFuelEfficiency()` and `getMaxSpeed()` are declared as `public`, meaning they can be called from outside their class.

## *Object Oriented Programming (N-PCCCS401P)*

### 8. Method Invocation

- Method invocation refers to calling or executing a method in Java. A method can be invoked on an object to perform certain actions.

In the program:

```
car.calculateFuelEfficiency();  
truck.calculateFuelEfficiency();
```

These statements invoke the calculateFuelEfficiency() method on the respective objects, demonstrating polymorphism where the specific method is chosen based on the object type.

### 9. Object Instantiation

- Object Instantiation is the process of creating a new object from a class. In Java, this is done using the new keyword.

In the program:

```
Car car = new Car("Toyota", "Camry", 2020, "Petrol");
```

This line creates a new Car object by calling the constructor and passing the required arguments.

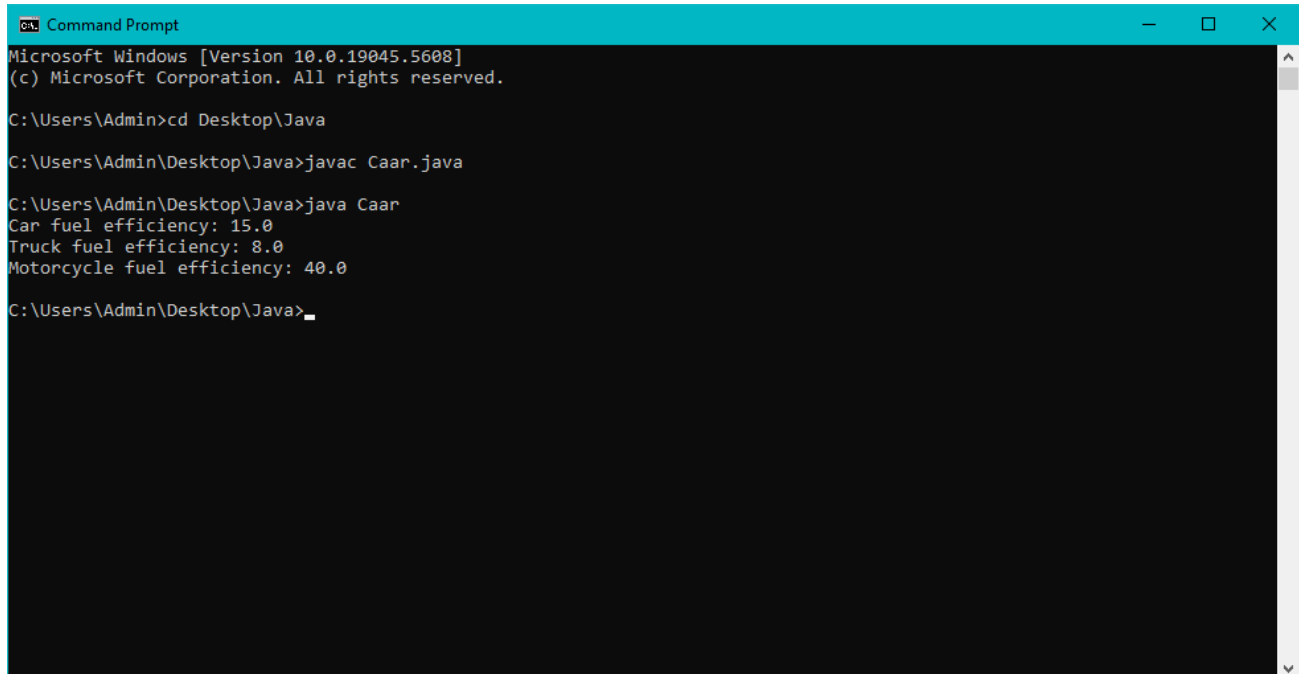
### **CODING:**

```
class Vehicle {  
    String make, model, fuelType;  
    int year;  
  
    public Vehicle(String make, String model, int year, String fuelType) {  
        this.make = make;  
        this.model = model;  
        this.year = year;  
        this.fuelType = fuelType;  
    }  
  
    public double calculateFuelEfficiency() { return 0.0; }  
    public double getMaxSpeed() { return 0.0; }  
}  
  
class Car extends Vehicle {  
    public Car(String make, String model, int year, String fuelType) {  
        super(make, model, year, fuelType);  
    }  
    public double calculateFuelEfficiency() { return 15.0; }  
    public double getMaxSpeed() { return 200.0; }  
}  
  
class Truck extends Vehicle {  
    public Truck(String make, String model, int year, String fuelType) {  
        super(make, model, year, fuelType);  
    }  
    public double calculateFuelEfficiency() { return 8.0; }  
    public double getMaxSpeed() { return 120.0; }
```

## Object Oriented Programming (N-PCCCS401P)

```
}  
  
class Motorcycle extends Vehicle {  
    public Motorcycle(String make, String model, int year, String fuelType) {  
        super(make, model, year, fuelType);  
    }  
    public double calculateFuelEfficiency() { return 40.0; }  
    public double getMaxSpeed() { return 180.0; }  
}  
  
public class Caar {  
    public static void main(String[] args) {  
        Vehicle car = new Car("Toyota", "Camry", 2020, "Petrol");  
        System.out.println("Car fuel efficiency: " + car.calculateFuelEfficiency());  
  
        Vehicle truck = new Truck("Ford", "F-150", 2019, "Diesel");  
        System.out.println("Truck fuel efficiency: " +  
truck.calculateFuelEfficiency());  
  
        Vehicle motorcycle = new Motorcycle("Yamaha", "YZF-R1", 2021, "Petrol");  
        System.out.println("Motorcycle fuel efficiency: " +  
motorcycle.calculateFuelEfficiency());  
    }  
}
```

### OUTPUT:



```
Microsoft Windows [Version 10.0.19045.5608]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\Admin>cd Desktop\Java  
  
C:\Users\Admin\Desktop\Java>javac Caar.java  
  
C:\Users\Admin\Desktop\Java>java Caar  
Car fuel efficiency: 15.0  
Truck fuel efficiency: 8.0  
Motorcycle fuel efficiency: 40.0  
  
C:\Users\Admin\Desktop\Java>
```

## ***Object Oriented Programming (N-PCCCS401P)***

### **CONCLUSION:**

The program demonstrates key OOP concepts like class creation, inheritance, method overriding, polymorphism, encapsulation, and constructors in Java. These principles help in structuring programs in a modular and reusable way, making the code easier to maintain and extend.

### **REFERENCE:**

<https://docs.oracle.com/javase/tutorial/java/concepts/class.html>  
<https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html>  
<https://docs.oracle.com/javase/tutorial/java/IandI/override.html>