```
// Vehicle.java
class Vehicle {
    protected String regNo, type, status;
    protected int odometer, lastServiceDate;

    public Vehicle(String regNo, String type, int odometer, int lastServiceDate) {
        this.regNo = regNo;
        this.type = type;
        this.odometer = odometer;
        this.lastServiceDate = lastServiceDate;
        this.status = "Active";
    }

    public void updateOdometer(int km) { odometer += km; }
    public String getRegNo() { return regNo; }
    public int getOdometer() { return odometer; }

    public void addService(ServiceRecord s) {
        lastServiceDate = s.date;
        System.out.println("Service added for " + regNo);
    }

    public int nextServiceDue() { return lastServiceDate + 5000; } // default
    public double operatingCostPerKm(double totalFuelCost) {
        return totalFuelCost / Math.max(1, odometer);
    }
```

```
}

// Truck.java
class Truck extends Vehicle {
   public Truck(String r, int o, int d) { super(r, "Truck", o, d); }
   @Override public int nextServiceDue() { return lastServiceDate + 7000; }
}

// Car.java
class Car extends Vehicle {
   public Car(String r, int o, int d) { super(r, "Car", o, d); }
   @Override public int nextServiceDue() { return lastServiceDate + 10000; }
}

// Bike.java
class Bike extends Vehicle {
   public Bike(String r, int o, int d) { super(r, "Bike", o, d); }
   @Override public int nextServiceDue() { return lastServiceDate + 3000; }
}

// ServiceRecord.java
class ServiceRecord {
   int recordId, date;
   String serviceType, notes;
   Vehicle vehicle;
   double cost;
```

```
    public ServiceRecord(int id, Vehicle v, String t, double c, int d, String n) {
        recordId = id; vehicle = v; serviceType = t; cost = c; date = d; notes = n;
    }
}

// FuelEntry.java
class FuelEntry {
    int entryId, odometer;
    Vehicle vehicle;
    double liters, pricePerLiter;
    String station;

    public FuelEntry(int id, Vehicle v, double l, double p, int o, String s) {
        entryId = id; vehicle = v; liters = l; pricePerLiter = p;
        odometer = o; station = s;
    }

    // Overloaded methods
    public double logFuel(double liters, double price) { return liters * price; }
    public double logFuel(double totalAmount) { return totalAmount; }
}

// FleetService.java
class FleetService {
    public void logFuel(FuelEntry f) {
```

```
        System.out.println("Fuel logged for " + f.vehicle.getRegNo() + " : " +
(f.liters*f.pricePerLiter));
    }
    public void addService(ServiceRecord s) { s.vehicle.addService(s); }
    public void healthReport(Vehicle v) {
        System.out.println(v.getRegNo()+" next service at "+v.nextServiceDue()+" km");
    }
    public void utilization(Vehicle v) {
        System.out.println(v.getRegNo()+" utilization: "+v.getOdometer()+" km");
    }
}

// Main
public class FleetAppMain {
    public static void main(String[] args) {
        Vehicle v1 = new Truck("TN01A1234", 12000, 8000);
        Vehicle v2 = new Car("TN02B5678", 5000, 2000);
        Vehicle v3 = new Bike("TN03C9999", 2000, 1000);

        FleetService fs = new FleetService();

        // Fuel & Service
        FuelEntry f1 = new FuelEntry(1, v1, 50, 100, 12000, "HP");
        fs.logFuel(f1);

        ServiceRecord s1 = new ServiceRecord(1, v2, "Oil Change", 2000, 6000, "OK");
```

```
        fs.addService(s1);


        // Reports (Polymorphism: loop Vehicles)
        Vehicle[] fleet = {v1, v2, v3};
        for (Vehicle v : fleet) {
            fs.healthReport(v);
            fs.utilization(v);
            System.out.println("Cost/km: " + v.operatingCostPerKm(5000));
        }
    }
}
```