19CSE204 – OBJECT ORIENTED PARADIGM

LAB PRACTICALS

SUBMITTED BY: **MADHUNISHA M V**

CH.EN.U4CCE22017

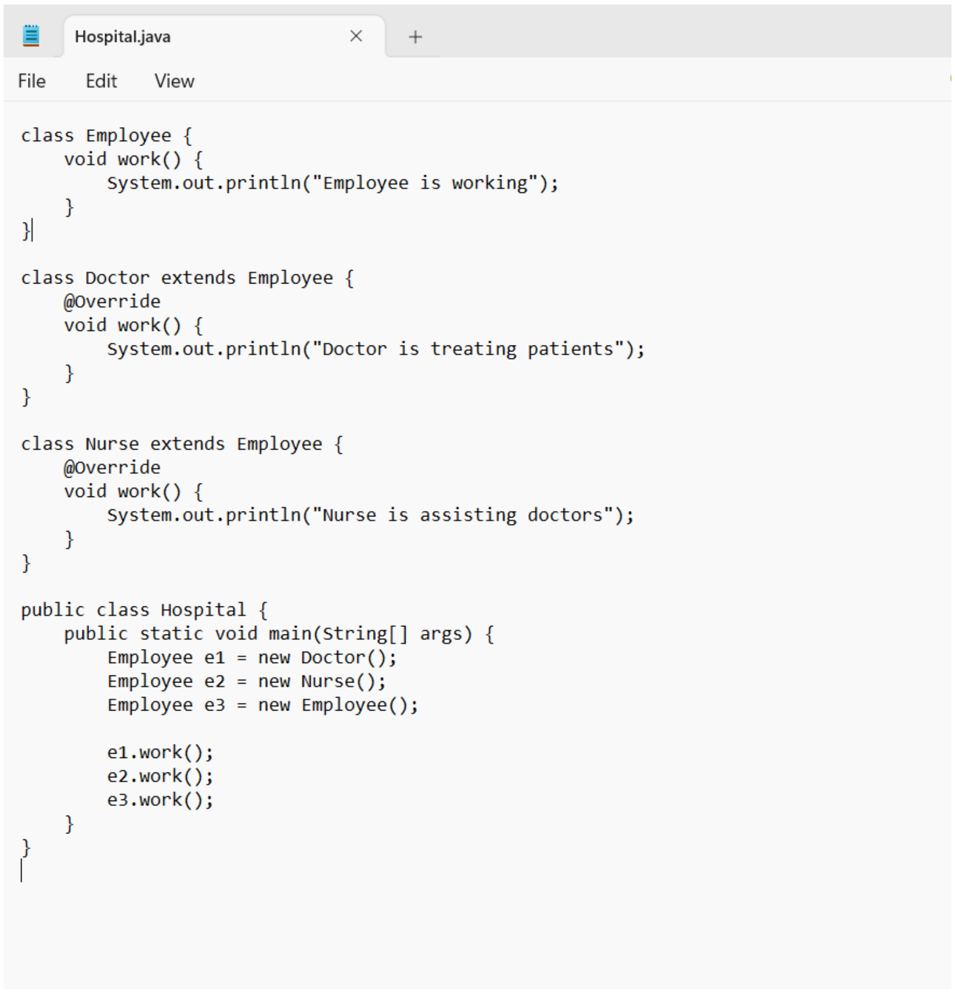
SUBMITTED TO: **DR. S. SUTHIR**

ABSTRACTION

AIM:

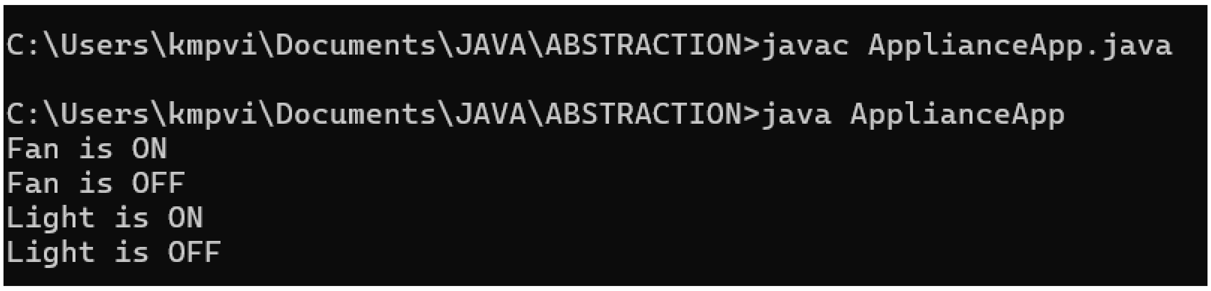
To understand and implement the concept of **abstraction** in Java using **abstract classes** and **abstract methods**, demonstrating how common functionalities can be shared while allowing subclasses to define specific implementations.

CODE 1:



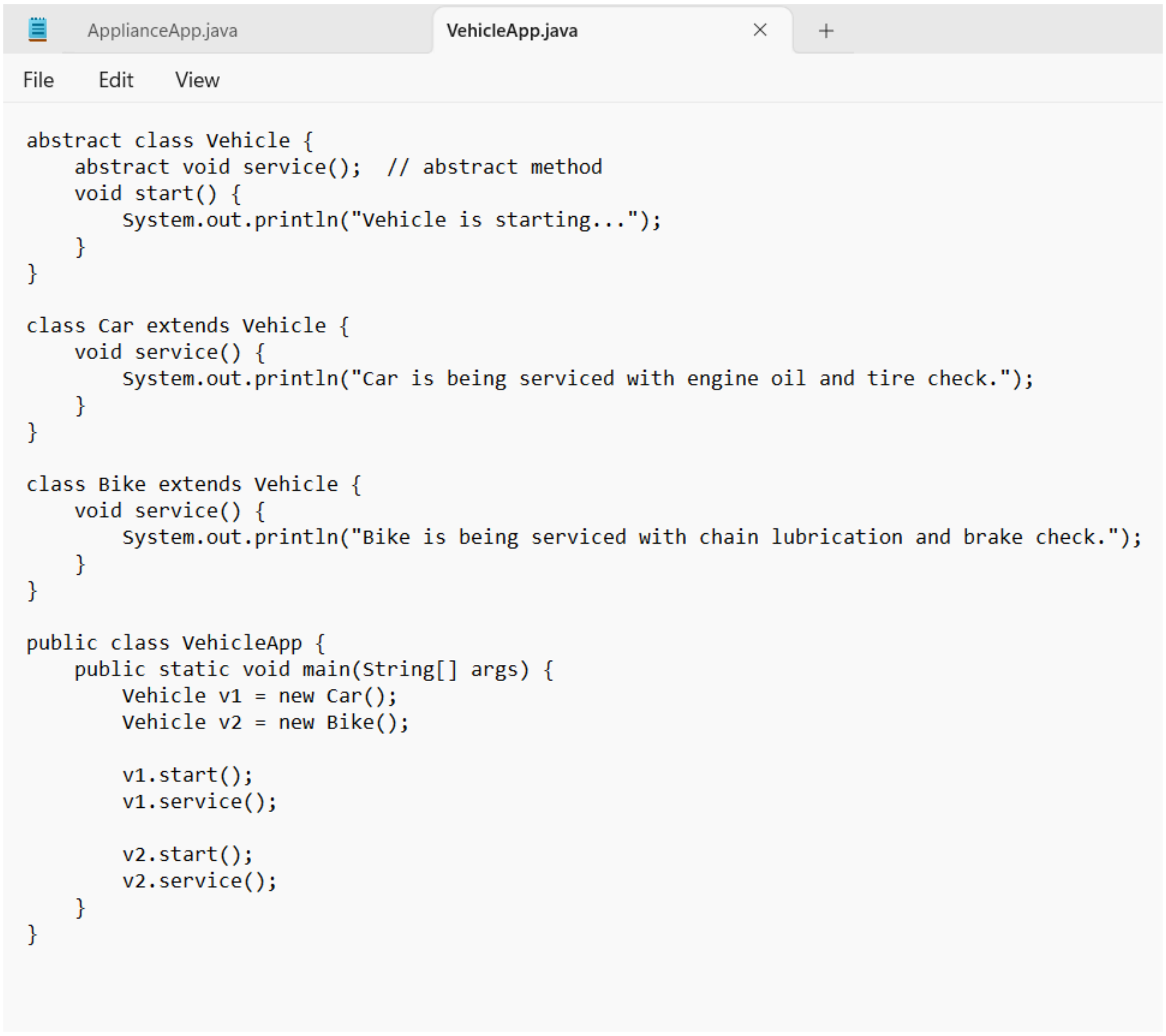
```
class Employee {
    void work() {
        System.out.println("Employee is working");
    }
}

class Doctor extends Employee {
    @Override
    void work() {
        System.out.println("Doctor is treating patients");
    }
}

class Nurse extends Employee {
    @Override
    void work() {
        System.out.println("Nurse is assisting doctors");
    }
}

public class Hospital {
    public static void main(String[] args) {
        Employee e1 = new Doctor();
        Employee e2 = new Nurse();
        Employee e3 = new Employee();

        e1.work();
        e2.work();
        e3.work();
    }
}
```

OUTPUT:



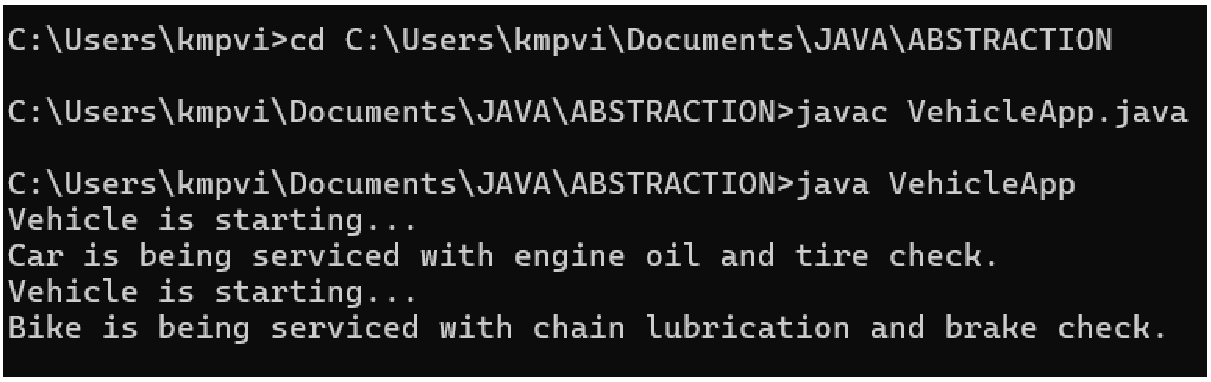
```
C:\Users\kmpvi\Documents\JAVA\ABSTRACTION>javac ApplianceApp.java

C:\Users\kmpvi\Documents\JAVA\ABSTRACTION>java ApplianceApp
Fan is ON
Fan is OFF
Light is ON
Light is OFF
```

CODE 2:



```java
abstract class Vehicle {
    abstract void service();  // abstract method
    void start() {
        System.out.println("Vehicle is starting...");
    }
}

class Car extends Vehicle {
    void service() {
        System.out.println("Car is being serviced with engine oil and tire check.");
    }
}

class Bike extends Vehicle {
    void service() {
        System.out.println("Bike is being serviced with chain lubrication and brake check.");
    }
}

public class VehicleApp {
    public static void main(String[] args) {
        Vehicle v1 = new Car();
        Vehicle v2 = new Bike();

        v1.start();
        v1.service();

        v2.start();
        v2.service();
    }
}
```

OUTPUT:



```
C:\Users\kmpvi>cd C:\Users\kmpvi\Documents\JAVA\ABSTRACTION

C:\Users\kmpvi\Documents\JAVA\ABSTRACTION>javac VehicleApp.java

C:\Users\kmpvi\Documents\JAVA\ABSTRACTION>java VehicleApp
Vehicle is starting...
Car is being serviced with engine oil and tire check.
Vehicle is starting...
Bike is being serviced with chain lubrication and brake check.
```

INFERENCE:

The programs demonstrate **abstraction**, one of the core principles of **Object-Oriented Programming (OOP)**.

In the VehicleApp example:

- The abstract class Vehicle defines a common behavior start() and an abstract method service().
- The subclasses Car and Bike provide their **own implementation** of the service() method, showing how abstraction hides details and exposes only essential functionalities.

In the ApplianceApp example:

- The abstract class Appliance provides a **template** for all electrical appliances with abstract methods turnOn() and turnOff(), while the showStatus() method is implemented to display the current state.
- Subclasses Fan and Light implement these abstract methods to define **specific behaviors** for turning on and off.

Overall, abstraction helps in:

- **Hiding complex implementation details** and showing only relevant features.
- **Encouraging code reusability** and **maintainability**.
- Providing a **blueprint** for subclasses to implement specific functionalities.