	Course Name: Design Patterns/Thinking LAB	**EXPERIMENT NO. 12**	
	Course Code: 20CP210P **Faculty: Dr. Ketan Sabale**	**Branch: CSE**	**Semester: IV**
Submitted by: Rhythm Shah **Roll no:** 22BCP071			

Objective: To familiarize students with standard Behavioral design patterns.
Experiment: Explain the State design pattern and write a program using any object-oriented programming language to demonstrate the working of State design pattern.

<u>Theory: -</u>

A behavioral software design pattern called the State pattern enables an entity to change its behavior in response to changes in its internal state. To do this, the object's behavior is encapsulated behind many state objects, which it dynamically transitions between according on its current state.

Components of State Design Pattern
1. Context
The Context is the class that contains the object whose behavior changes based on its internal state. It maintains a reference to the current state object that represents the current state of the Context. The Context provides an interface for clients to interact with and typically delegates state-specific behavior to the current state object.

2. State Interface or Base Class
The State interface or base class defines a common interface for all concrete state classes. This interface typically declares methods that represent the state-specific behavior that the Context can exhibit. It allows the Context to interact with state objects without knowing their concrete types.

3. Concrete States

Concrete state classes implement the State interface or extend the base class. Each concrete state class encapsulates the behavior associated with a specific state of the Context. These classes define how the Context behaves when it is in their respective states.

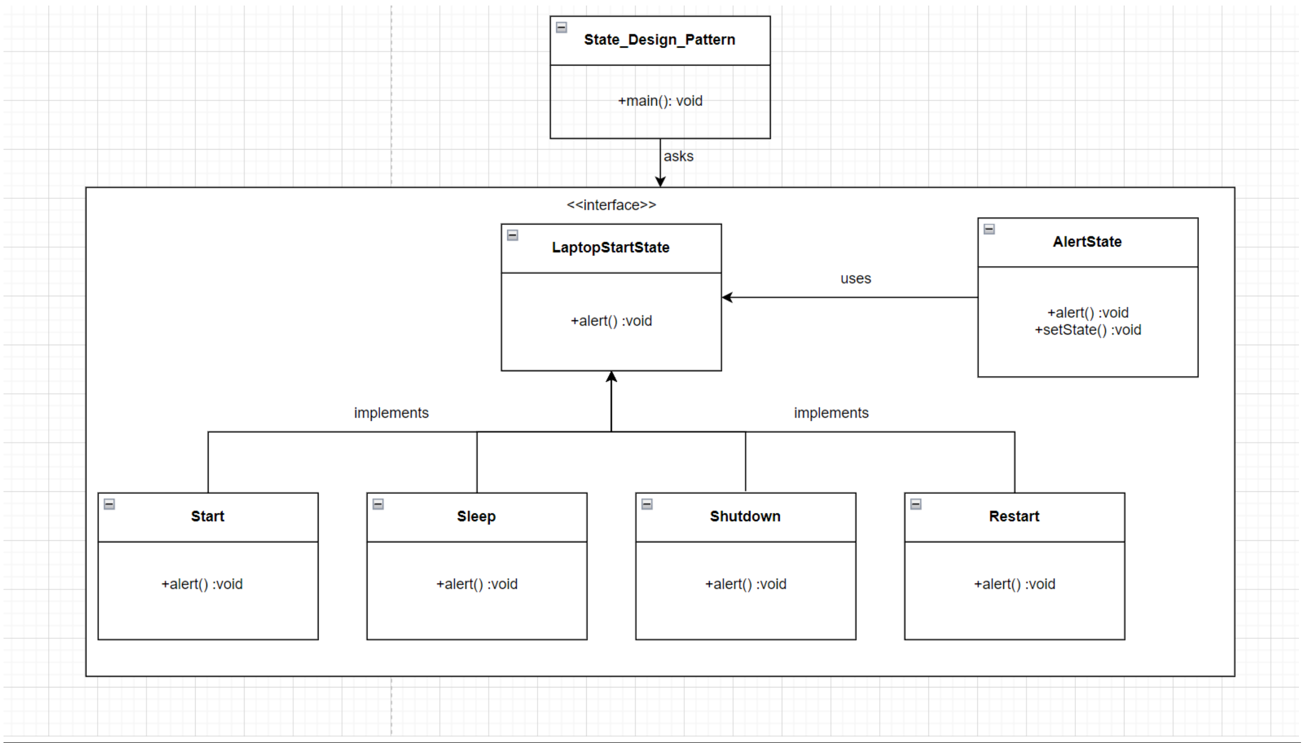
Implementation: -

In this stated design pattern, I have taken example of operating system of Laptop.
I have taken 4 states which are: -

1) Start
2) Sleep
3) Shutdown
4) Restart

One interface is created of LaptopStartState and a class of AlertState the four states as mentioned above implements the LaptopStartState and a statecontext method is created which can change the state in main method.

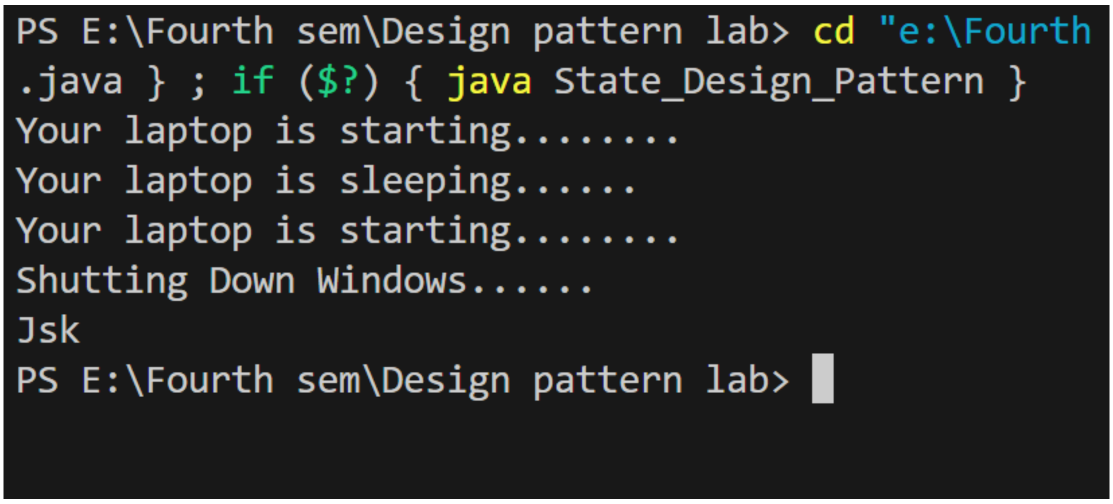
UML Diagram: -



Code: -

```java
interface LaptopStartState {
    public void alert(AlertState x);
}

class AlertState {
    private LaptopStartState currentState;

    public AlertState() {
        currentState = new Start();
    }

    public void setState(LaptopStartState state) {
        currentState = state;
    }

    public void alert() {
        currentState.alert(this);
    }
}

class Start implements LaptopStartState {


    public void alert(AlertState x) {
        System.out.println("Your laptop is starting........");
    }
}
class Sleep implements LaptopStartState {


    public void alert(AlertState x) {
        System.out.println("Your laptop is sleeping......");
    }

}

class ShutDown implements LaptopStartState {
    public void alert(AlertState x){
        System.out.println("Shutting Down Windows......");
        System.out.println("Jsk");
    }
}

class Restart implements LaptopStartState{
    public void alert(AlertState x) {
        System.out.println("Your laptop is restarting.....");
    }
}


public class State_Design_Pattern {
    public static void main(String[] args) {
        AlertState stateContext = new AlertState();
        stateContext.alert();
        stateContext.setState(new Sleep());
        stateContext.alert();
        stateContext.setState(new Start());
        stateContext.alert();
        stateContext.setState(new ShutDown());
        stateContext.alert();
    }
}
```

<u>Output: -</u>



```
PS E:\Fourth sem\Design pattern lab> cd "e:\Fourth
.java } ; if ($?) { java State_Design_Pattern }
Your laptop is starting........
Your laptop is sleeping......
Your laptop is starting........
Shutting Down Windows......
Jsk
PS E:\Fourth sem\Design pattern lab>
```