	Course Name: Design Patterns/Thinking LAB	**EXPERIMENT NO. 12**	
	Course Code: 20CP210P **Faculty: Dr. Ketan Sabale**	**Branch: CSE**	**Semester: IV**
(To be filled by Student) **Submitted by: Jangle Parth** **Roll no: 22BCP083**			

Objective: To familiarize students with standard Behavioral design patterns.

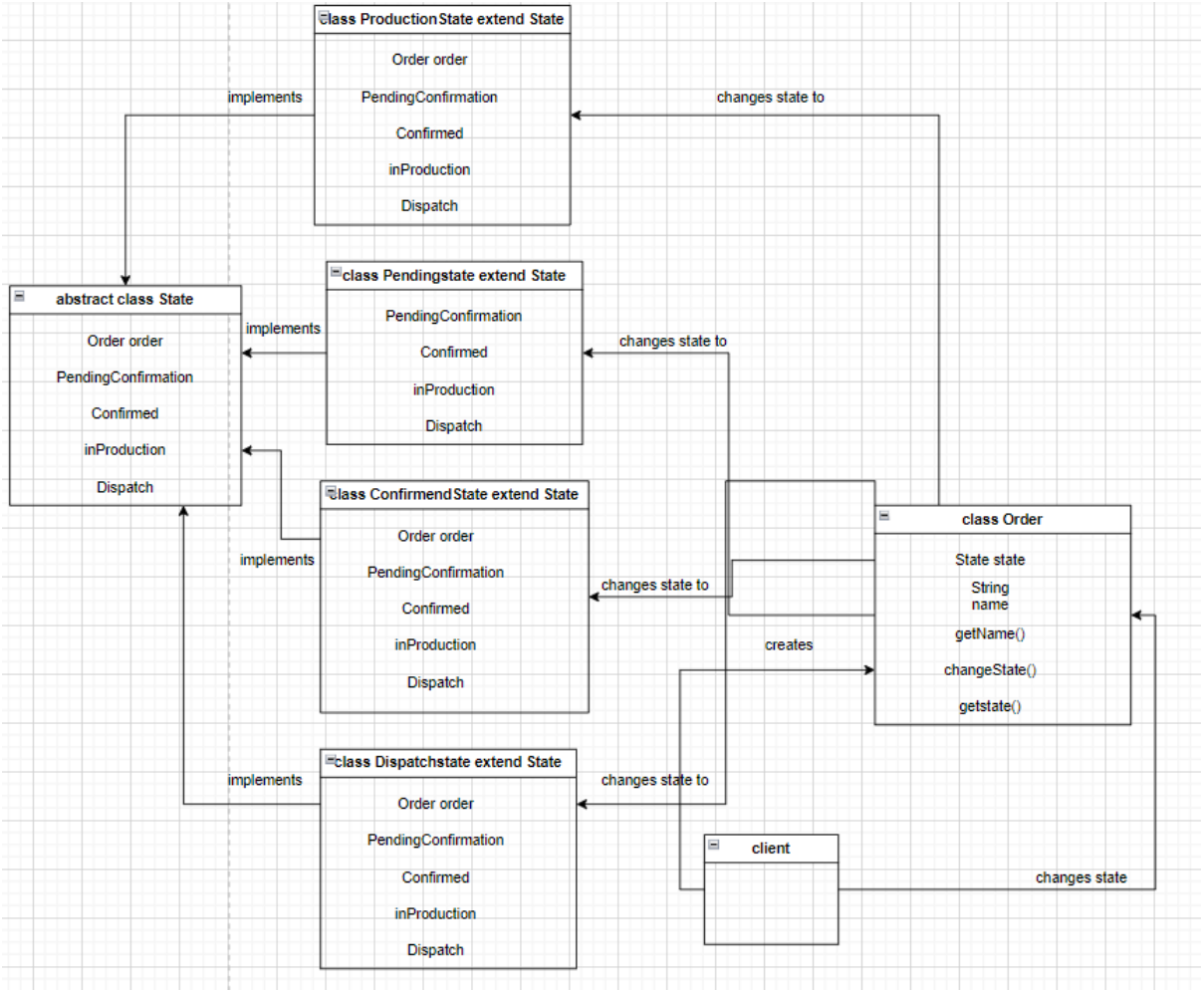
Experiment: Explain the State design pattern and write a program using any object-oriented programming language to demonstrate the working of State design pattern.

Theory: State Design Pattern is Used when your object has to move around specified amount of states for example mobile is either in silent or ringing or vibration etc. In State Design Pattern we have a abstract class State which is then implemented by all kind of state and all state have their own implementation of states.

Problem Statement Explanation:

We have a company which goes order the another company. And order can be in 4 states either there is pending confirmation or the order is confirmed i.e. advance payment is received or the order is in Production or it is dispatched .

Flowchart Explanation:



Code:

```
abstract class State {
    Order order;

    State(Order order) {
        this.order = order;
    }

    public abstract void PendingConfirmation();

    public abstract void Confirmed();

    public abstract void inProduction();

    public abstract void Dispatch();
}

class PendingState extends State {

    PendingState(Order order) {
        super(order);
    }

    public void PendingConfirmation() {
        System.out.println("Order Confirmation is Pending");
        order.changeState(new ConfirmedState(order));
    }

    public void Confirmed() {
        System.out.println("Waiting for Confirmation of " +
order.getName());
    }

    public void inProduction() {
        System.out.println("Waiting for Confirmation of " +
order.getName());
    }

    public void Dispatch() {
        System.out.println("Waiting for Confirmation of " +
order.getName());
    }
}

class ConfirmedState extends State {
```

```
    public ConfirmedState(Order Order) {
        super(Order);
    }

    public void PendingConfirmation() {
        System.out.println("Order Already Confirmed for " +
order.getName());
    }

    public void Confirmed() {
        order.changeState(new ConfirmedState(order));
        System.out.println("Advance Received for " + order.getName());
    }

    public void inProduction() {
        System.out.println("Waiting to Get Material of " +
order.getName());
    }

    public void Dispatch() {
        System.out.println("Waiting for tank to be Produced");
    }
}

class ProductionState extends State {

    ProductionState(Order Order) {
        super(Order);
    }

    public void PendingConfirmation() {
        System.out.println("Order Already Confirmed for " +
order.getName());
    }

    public void Confirmed() {
        System.out.println("Advance Received for " + order.getName());
        System.out.println("Material Sent for " + order.getName());
    }

    public void inProduction() {
        System.out.println("Production Stated for " + order.getName());
        order.changeState(new DispatchState(order));
    }
```

```
    public void Dispatch() {
        System.out.println("Waiting for tank to be Produced");
    }
}

class DispatchState extends State {

    DispatchState(Order Order) {
        super(Order);
    }

    public void PendingConfirmation() {
        System.out.println("Order Already Confirmed for" +
order.getName());
    }

    public void Confirmed() {
        System.out.println("Advance Received for " + order.getName());
        System.out.println("Material Sent for " + order.getName());
    }

    public void inProduction() {
        System.out.println("Production Completed for " +
order.getName());
    }

    public void Dispatch() {
        System.out.println("Dispatched order of " + order.getName());
    }
}

class Order {
    private State state;
    private String name;

    public String getName() {
        return name;
    }

    public Order(String name) {
        this.name = name;
    }

    public void changeState(State state) {
        this.state = state;
    }
```

```
    public State getState() {
        return state;
    }
}

public class tank {
    public static void main(String[] args) {
        Order tetrapack = new Order("Tetrapack");
        tetrapack.changeState(new PendingState(tetrapack));
        tetrapack.getState().Confirmed();
        Order Nestle = new Order("Nestle");
        Nestle.changeState(new ConfirmedState(Nestle));
        Nestle.getState().inProduction();
        Order Amul = new Order("Amul");
        Amul.changeState(new ProductionState(Amul));
        Amul.getState().Dispatch();
        tetrapack.changeState(new ConfirmedState(tetrapack));
        Nestle.changeState(new ProductionState(Nestle));
        Order CocaCola = new Order("Coca-Cola");
        CocaCola.changeState(new DispatchState(CocaCola));
        CocaCola.getState().Dispatch();

    }
}
```

Output:

```
PS C:\Users\onlyf\OneDrive\Desktop\PDEU\Sem4\Design Pattern> cd "c:\Users\onlyf\OneDrive\
op\PDEU\Sem4\Design Pattern\State\" ; if ($?) { javac tank.java } ; if ($?) { java tank }
Waiting for Confirmation of Tetrapack
Waiting to Get Material of Nestle
Waiting for tank to be Produced
Dispatched order of Coca-Cola
PS C:\Users\onlyf\OneDrive\Desktop\PDEU\Sem4\Design Pattern\State>
```