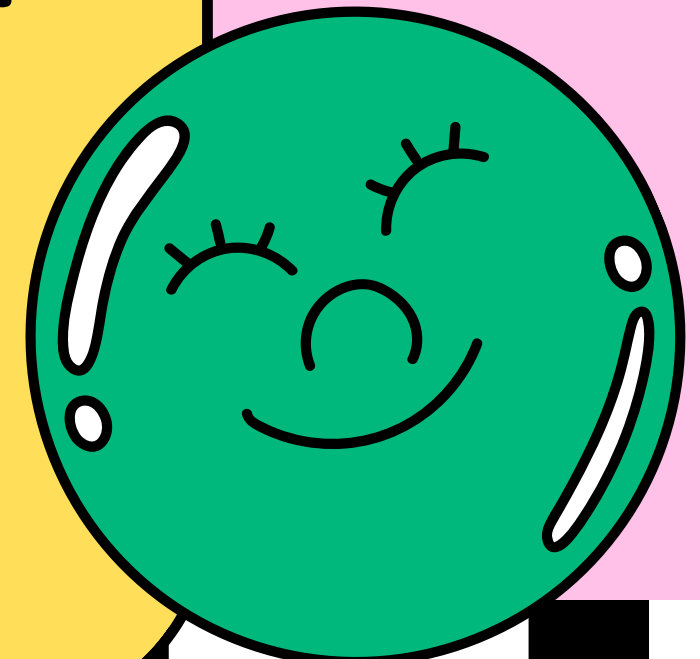


# STATE DESIGN PATTERN

Object-Oriented Software  
Development (Group 19)





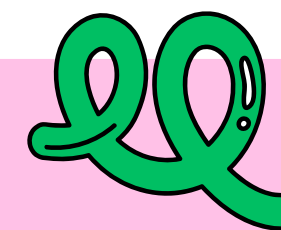
# WHAT IS STATE PATTERN?

State pattern เป็นการเปลี่ยน  
รูปแบบการทำงานของ object  
ตามสถานะของมัน มีการติดต่อกันคือ  
context object หรือ  
ตัว object หลักของแต่ละ state โดย  
จะเปลี่ยนไปตาม method ที่ถูกเรียก

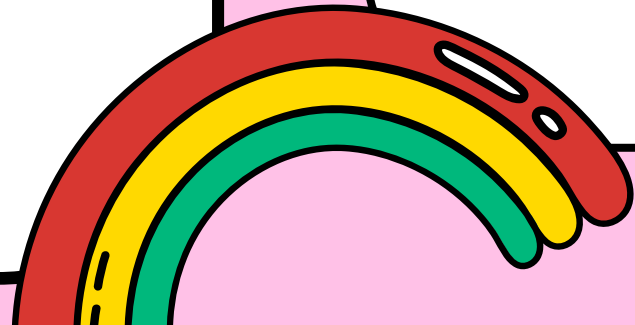


What category?

Behavioral Pattern



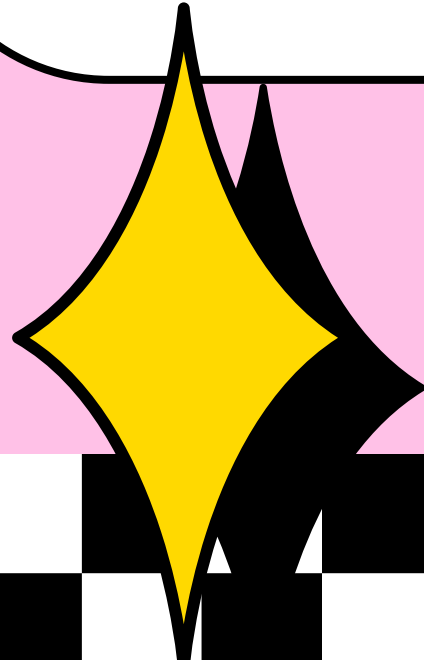
เป็นวิธีการออกแบบการติดต่อกัน  
ระหว่าง OBJECT ให้มีความยืดหยุ่น  
และสามารถติดต่อกันได้อย่างไม่มี  
ปัญหา

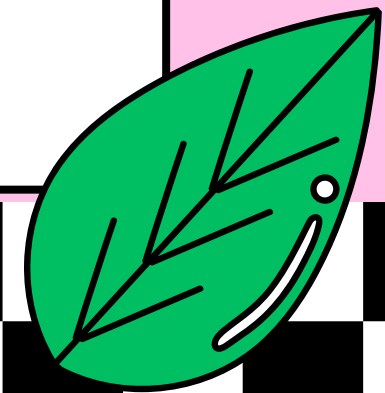
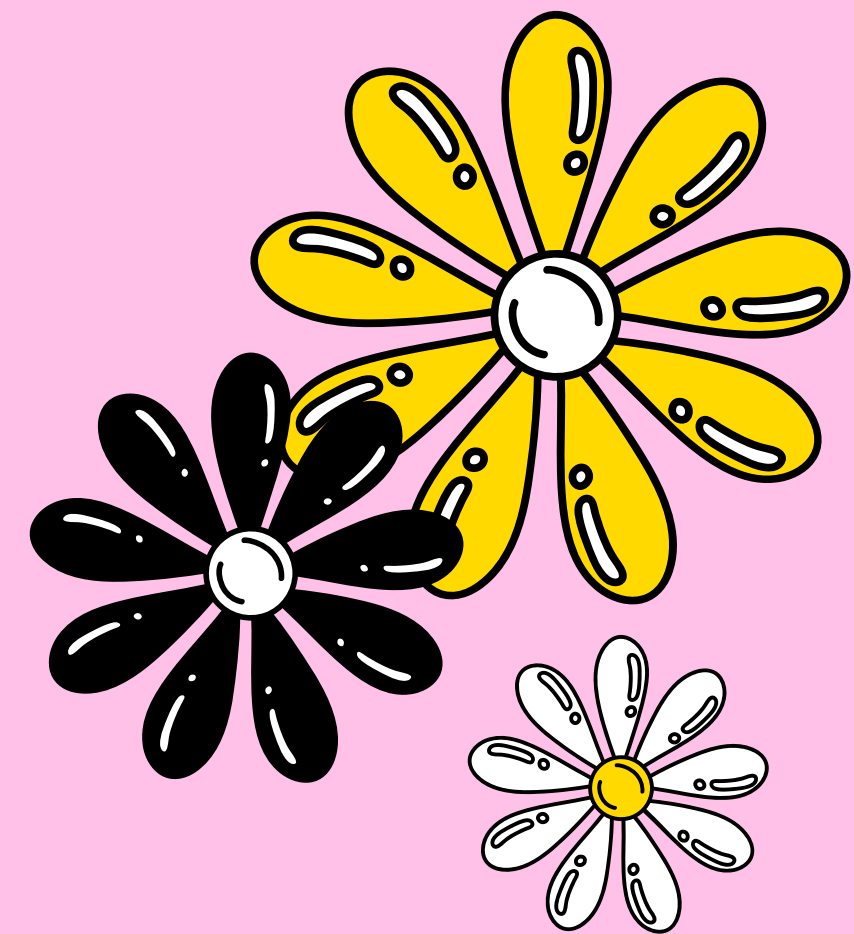


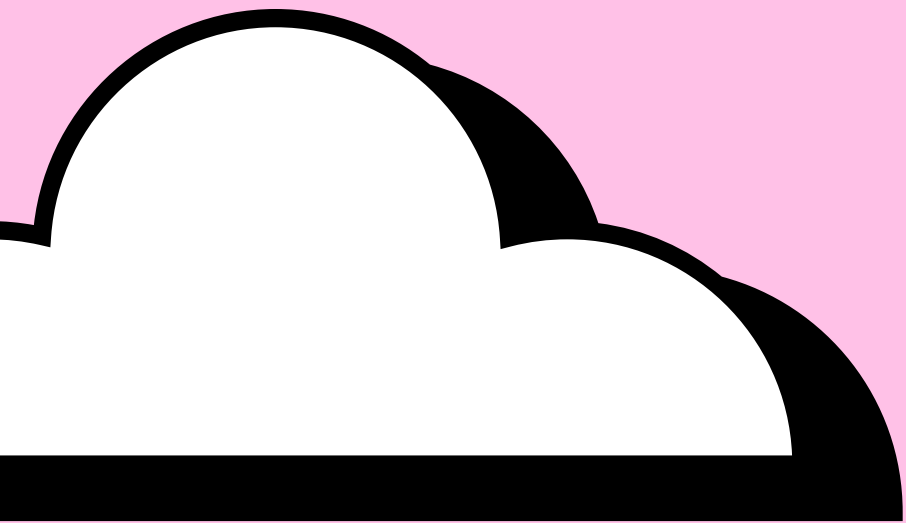


## ลักษณะการทำงานของ **State Pattern**

State อนุญาตให้ object เปลี่ยนแปลง  
พฤติกรรมเมื่อ state ภายใน  
เปลี่ยนแปลง ซึ่ง object จะแสดงให้เห็น  
ถึงการเปลี่ยนแปลงของมันเอง

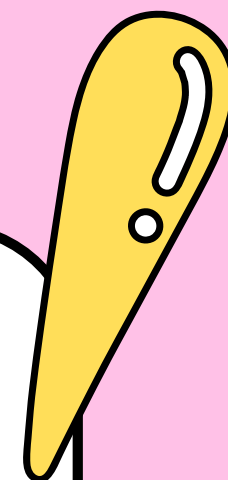
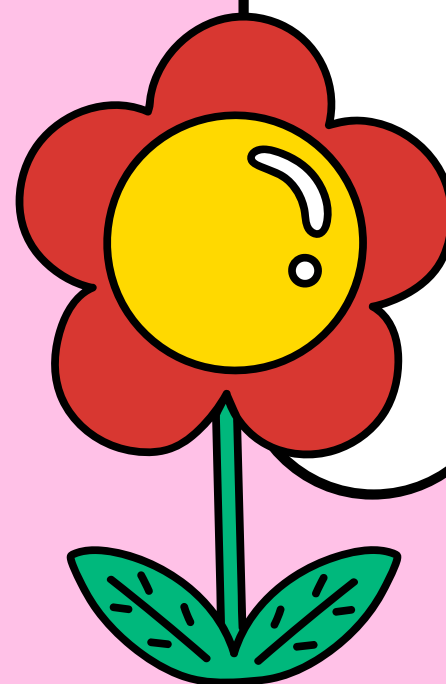
- 
- แยก state ออกมาเป็น class
  - ให้ context ทำงานกับ interface แทนการทำงานกับ concrete state





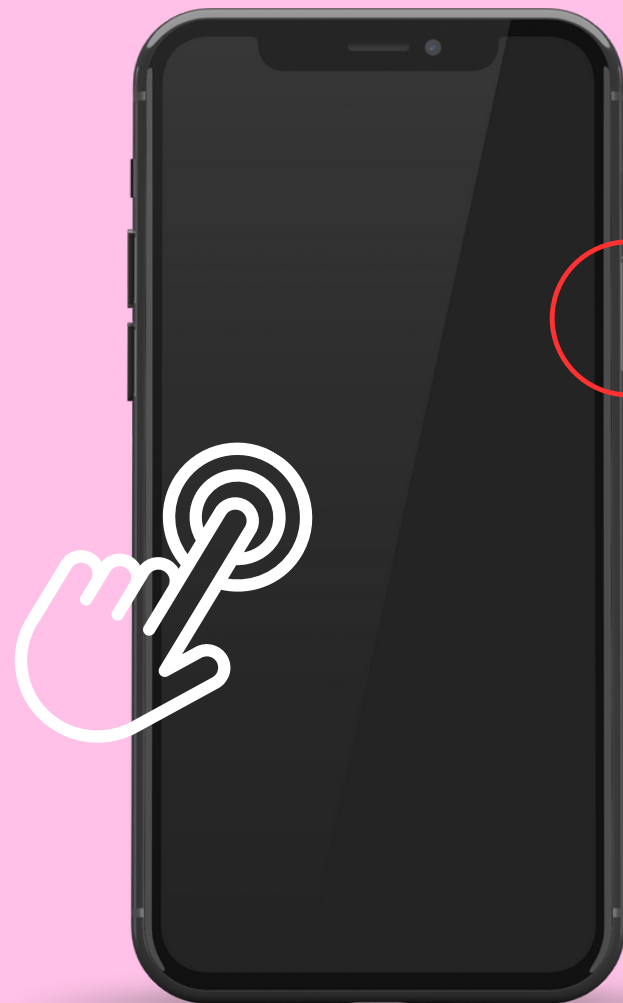
# เหมาะกับงาน ประเภทไหนบ้าง

เหมาะกับงาน software ที่ต้อง  
จัดการสถานะที่หลากหลาย และ  
แต่ละสถานะก็มีลักษณะเฉพาะ  
ของตัว โดยลดการแก้ไข code  
เดิม และสามารถเพิ่มสถานะอื่น  
ได้อีกเรื่อยๆ





# Example

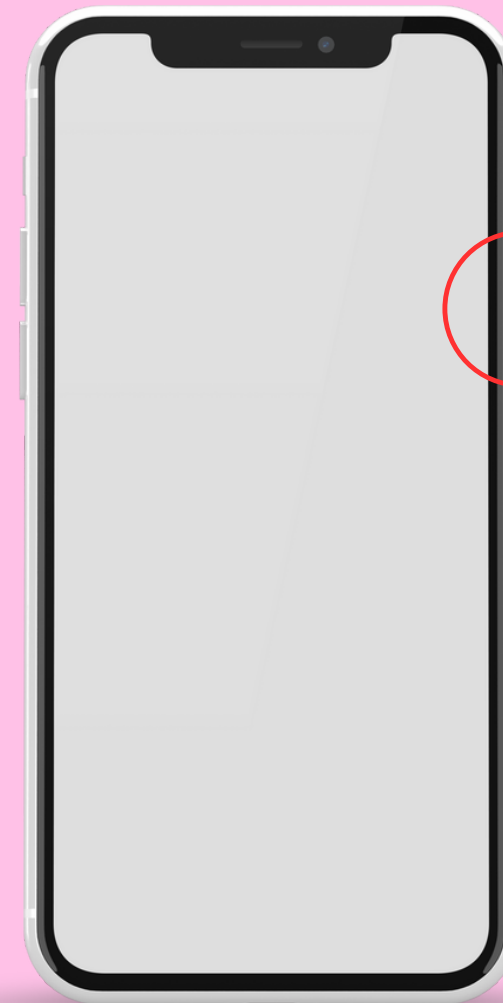


**State off**

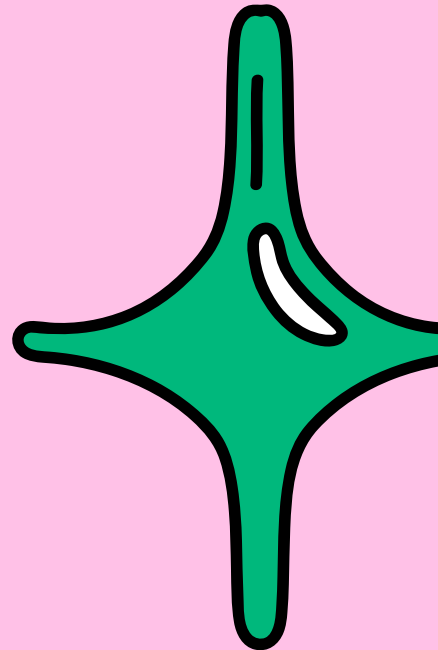
Touch or Use button



Use button

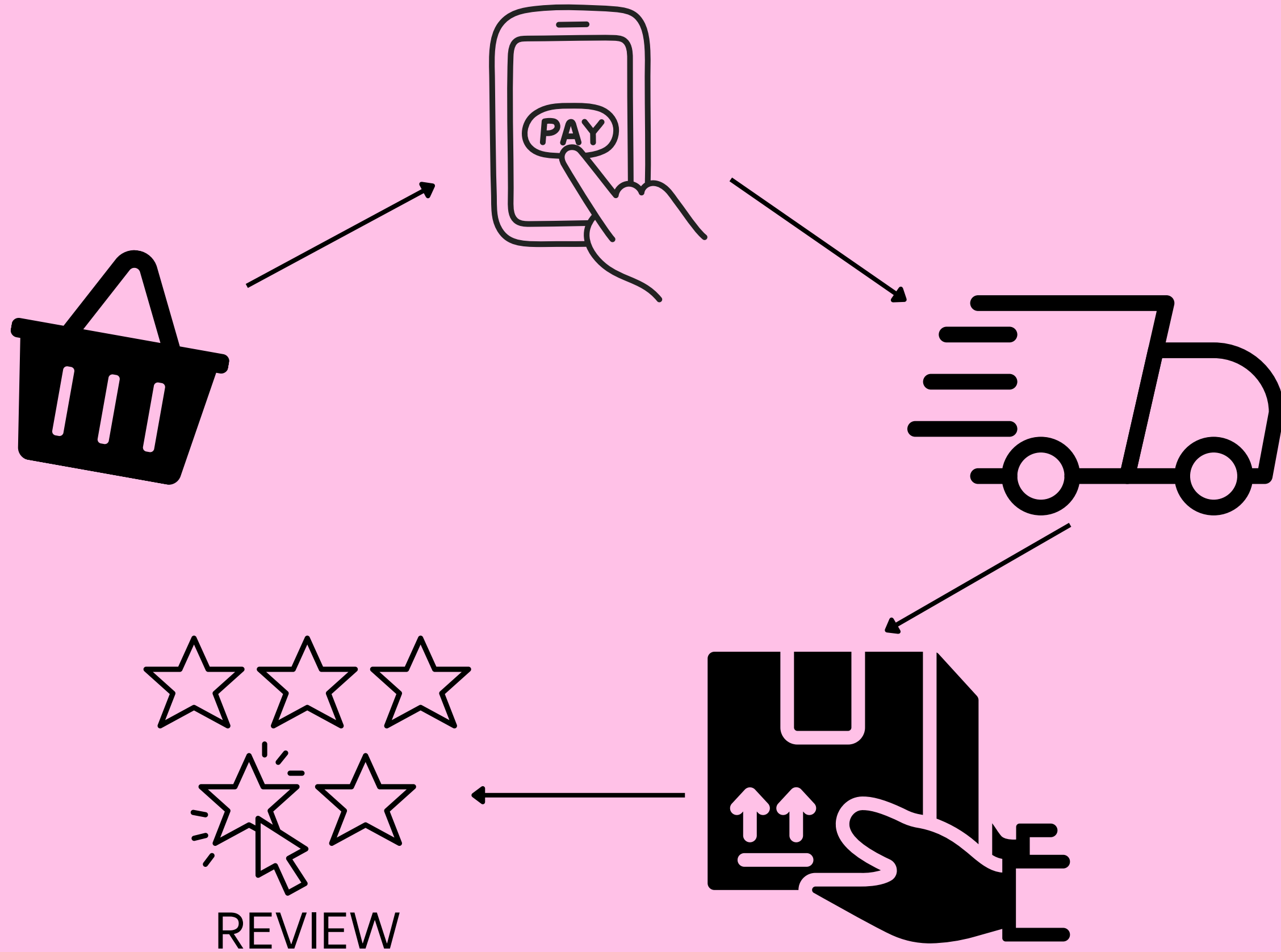
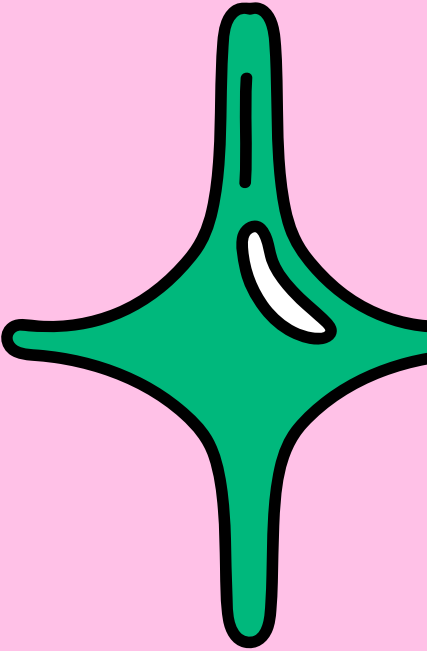


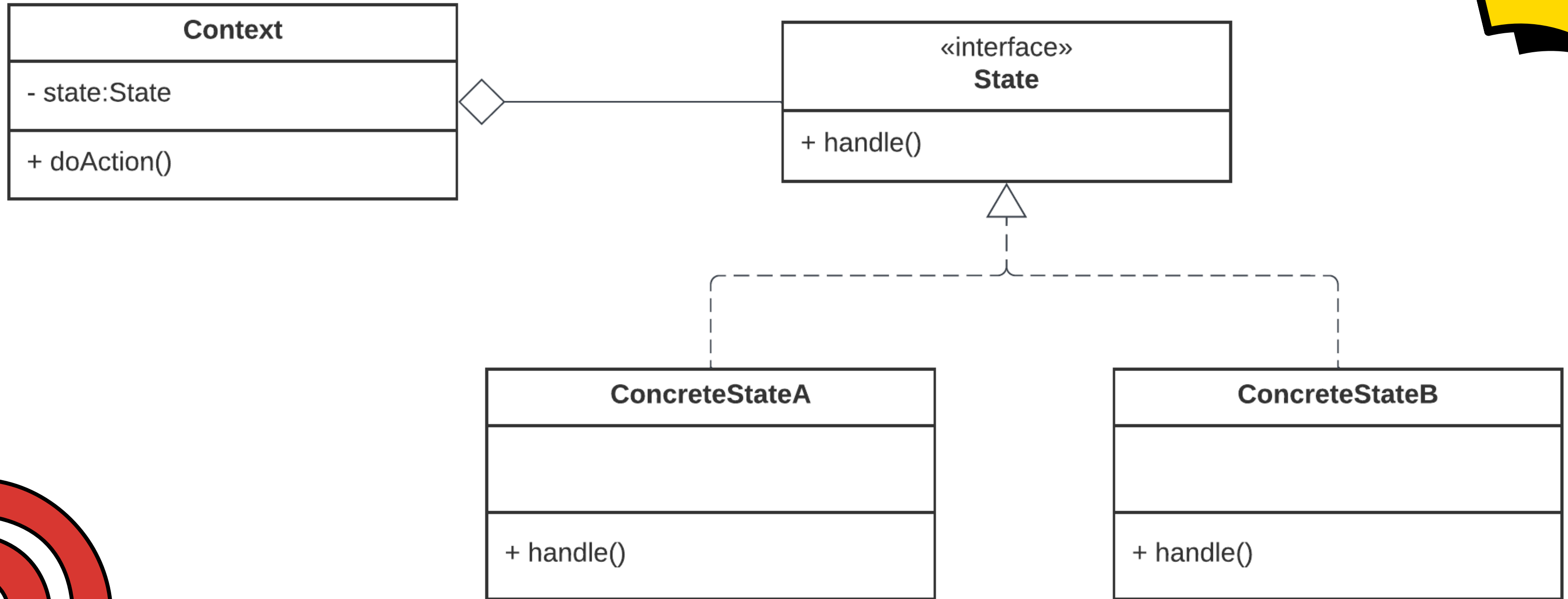
**State on**



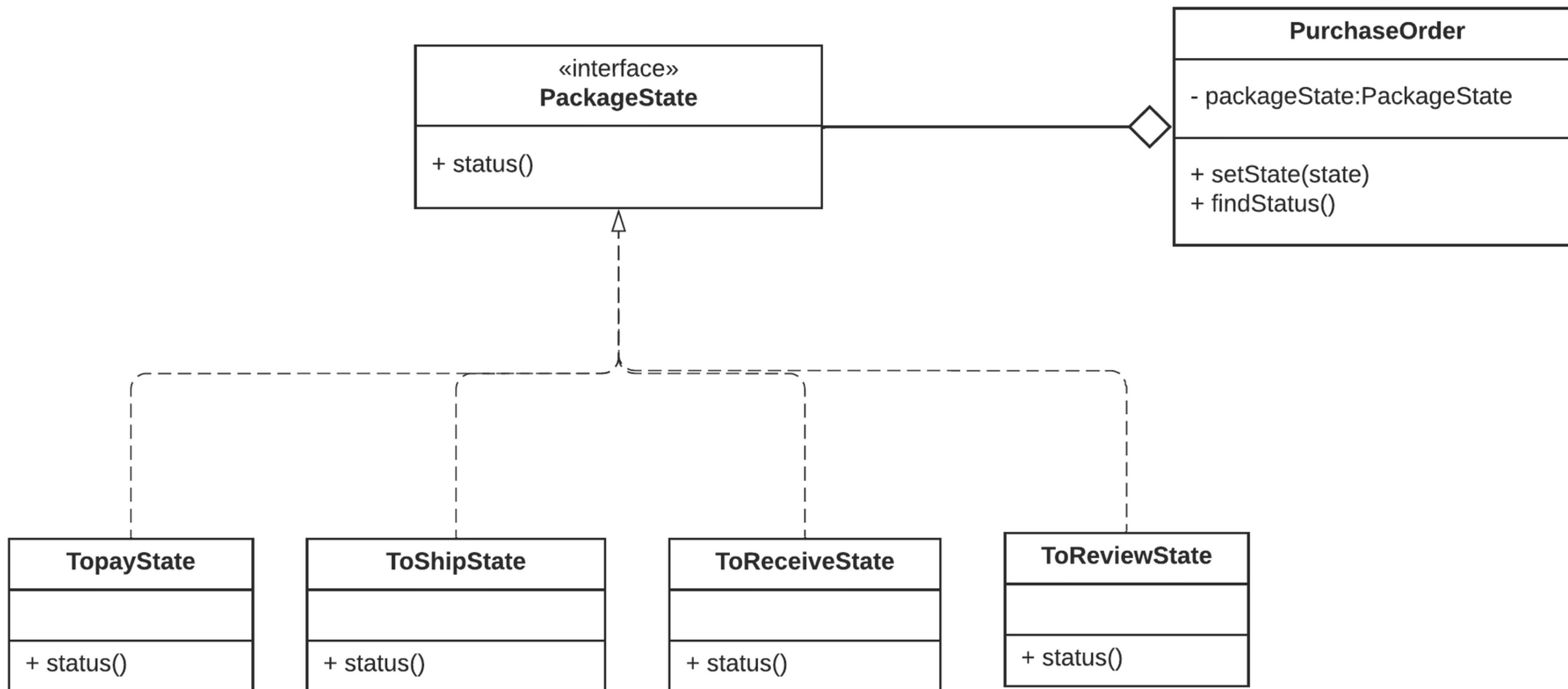


# Example





## STATE PATTERN DIAGRAM







```
1 public interface PackageState{  
2     void status();  
3 }
```



```
1 public class ToPayState implements PackageState{
2     public void status(){
3         System.out.println("is waiting for pay");
4     }
5 }
6 public class ToShipState implements PackageState{
7     public void status(){
8         System.out.println("your package is shipping");
9     }
10 }
11 public class ToReceiveState implements PackageState{
12     public void status(){
13         System.out.println("delivered");
14     }
15 }
16 public class ToReviewState implements PackageState{
17     public void status(){
18         System.out.println("review delivered order");
19     }
20 }
```





```
1 public class PurchaseOrder{
2     private PackageState packageState;
3     public PurchaseOrder(){
4         this.packageState = new ToPayState();
5     }
6     public void setState(String state){
7         if(state.equalsIgnoreCase("pay")){
8             this.packageState = new ToPayState();
9         }
10        else if(state.equalsIgnoreCase("ship")){
11            this.packageState = new ToShipState();
12        }
13        else if(state.equalsIgnoreCase("receive")){
14            this.packageState = new ToReceiveState();
15        }
16        else if(state.equalsIgnoreCase("review")){
17            this.packageState = new ToReviewState();
18        }
19    }
20    public void findStatus(){
21        packageState.status();
22    }
23 }
```



```
1 public class Demo
2 {
3     public static void test(){
4         PurchaseOrder PO = new PurchaseOrder();
5         PO.findStatus();
6         PO.setState("ship");
7         PO.findStatus();
8         PO.setState("receive");
9         PO.findStatus();
10        PO.setState("review");
11        PO.findStatus();
12    }
13 }
```

is waiting for pay

ship

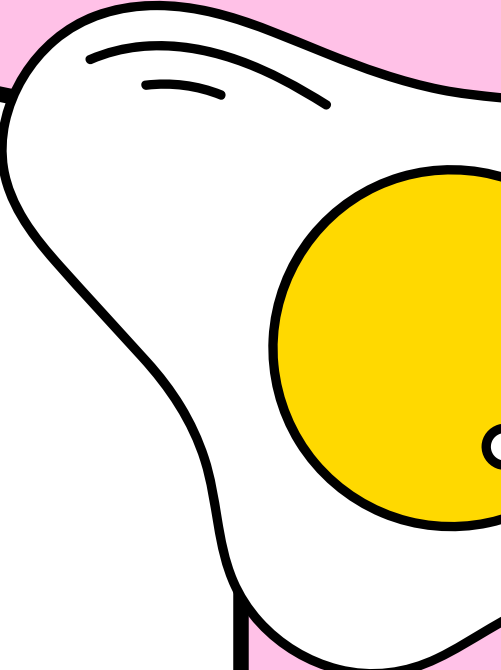
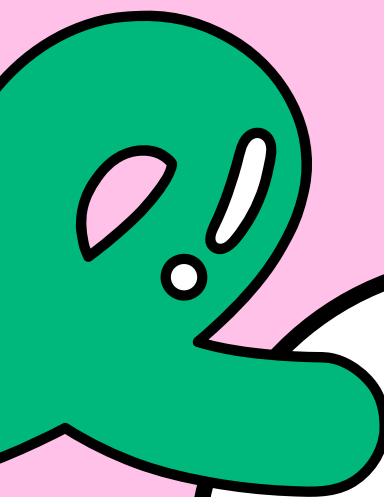
your package is shipping

receive

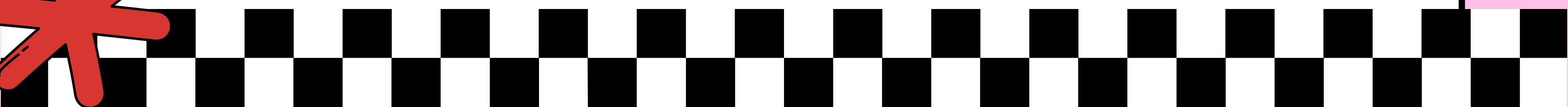

delivered

review

review delivered order



ใช้หลักการ Open-Closed Principle (OCP) สร้าง class ใหม่ขึ้นมาแล้ว implement interface ก็จะสามารถนำ class นั้นมาใช้ได้เลย โดยไม่ต้องแก้ไข code ใน base class



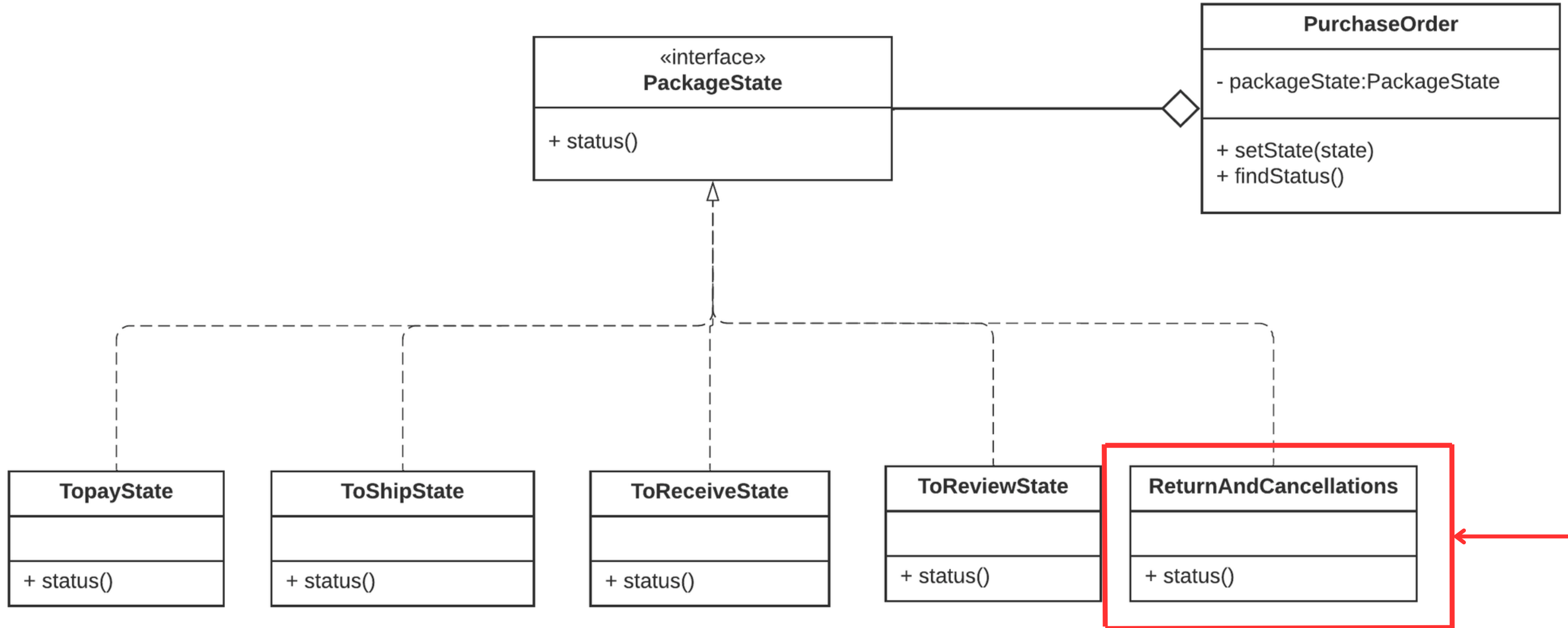


```
1 public class ToPayState implements PackageState{
2     public void status(){
3         System.out.println("is waiting for pay");
4     }
5 }
6 public class ToShipState implements PackageState{
7     public void status(){
8         System.out.println("your package is shipping");
9     }
10 }
11 public class ToReceiveState implements PackageState{
12     public void status(){
13         System.out.println("delivered");
14     }
15 }
16 public class ToReviewState implements PackageState{
17     public void status(){
18         System.out.println("review delivered order");
19     }
20 }
21 public class ReturnAndCancellations implements PackageState{
22     public void status(){
23         System.out.println("refund issued");
24     }
25 }
```





```
1 public class PurchaseOrder{
2     private PackageState packageState;
3     public PurchaseOrder(){
4         this.packageState = new ToPayState();
5     }
6     public void setState(String state){
7         if(state.equalsIgnoreCase("pay")){
8             this.packageState = new ToPayState();
9         }
10        else if(state.equalsIgnoreCase("ship")){
11            this.packageState = new ToShipState();
12        }
13        else if(state.equalsIgnoreCase("receive")){
14            this.packageState = new ToReceiveState();
15        }
16        else if(state.equalsIgnoreCase("review")){
17            this.packageState = new ToReviewState();
18        }
19        else if(state.equalsIgnoreCase("refund")){
20            this.packageState = new ReturnAndCancellations();
21        }
22    }
23    public void findStatus(){
24        packageState.status();
25    }
26 }
```







```
1 public class Demo
2 {
3     public static void test(){
4         PurchaseOrder PO = new PurchaseOrder();
5         PO.findStatus();
6         PO.setState("ship");
7         PO.findStatus();
8         PO.setState("receive");
9         PO.findStatus();
10        PO.setState("review");
11        PO.findStatus();
12        PO.setState("refund");
13        PO.findStatus();
14    }
15 }
```

is waiting for pay

ship

your package is shipping

receive

delivered

review

review delivered order

refund

refund issued



**THANK YOU  
FOR LISTENING!**



**GROUP 19**