Object-Oriented Software Engineering

Instructor : Huang, Chuen-Min

Teamwork1 ver.1

Group 5

ID	Name
B10523018	Jenny
B10523019	Jason
B10523020	Kendy
B10523021	Johnny
B10523022	Howard
B10523028	James
B10523029	Timothy
B10523030	Jerry
B10523032	Xavier
B10523033	Alex

Date 2018/10/31

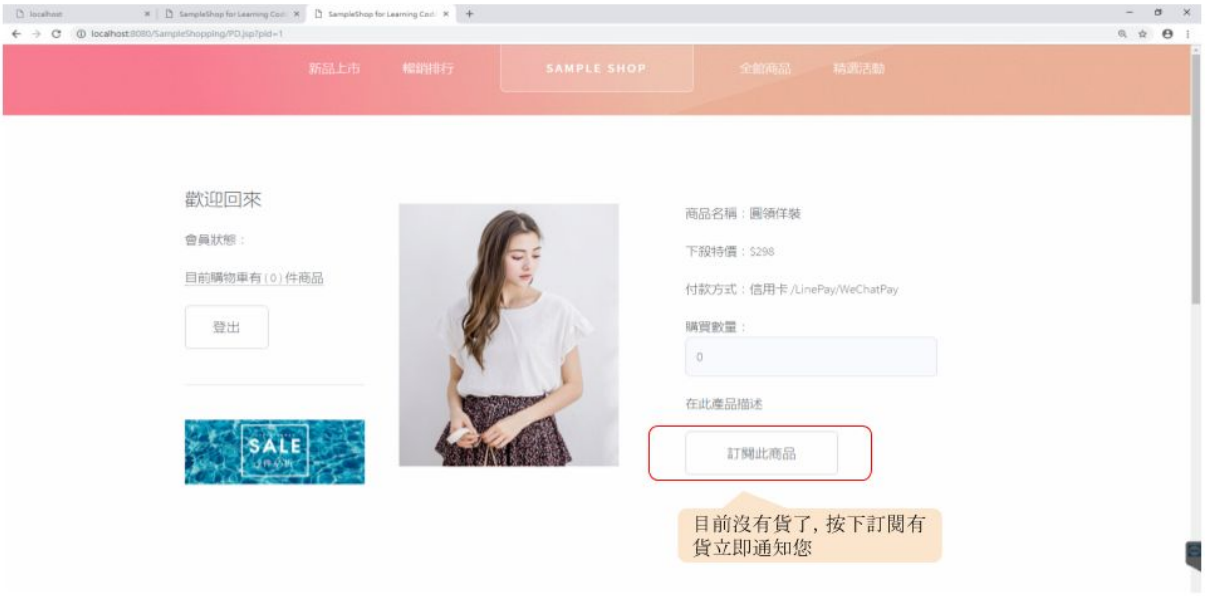
1) **Display some snapshots of the result in the report.**

Building blocks of this online shopping:

- **Products page**

- **Product info. page**

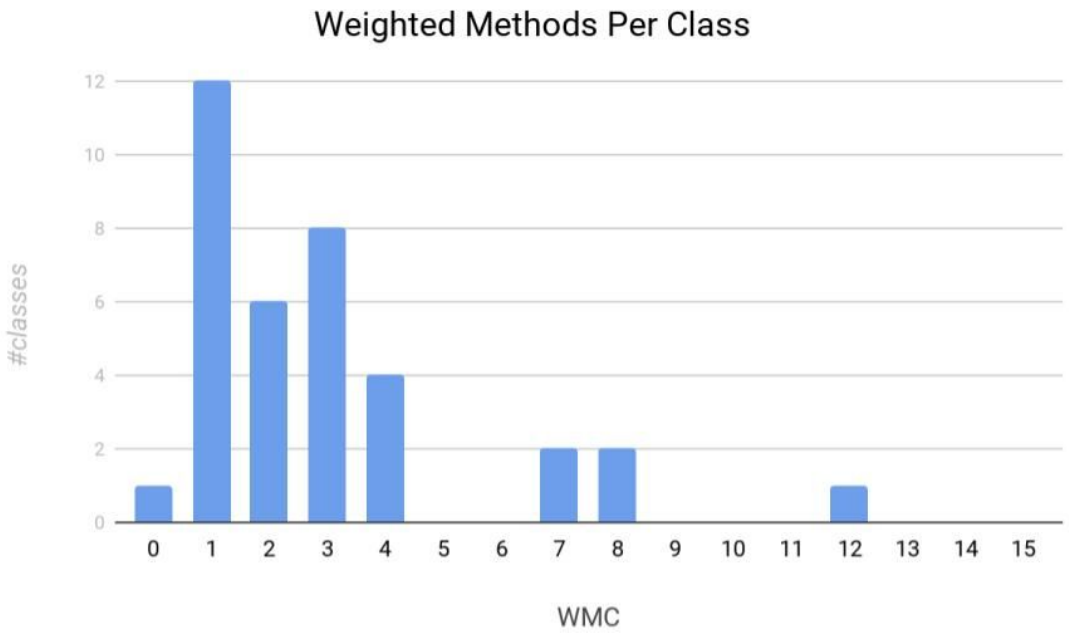
- **Subscribe specific item page**

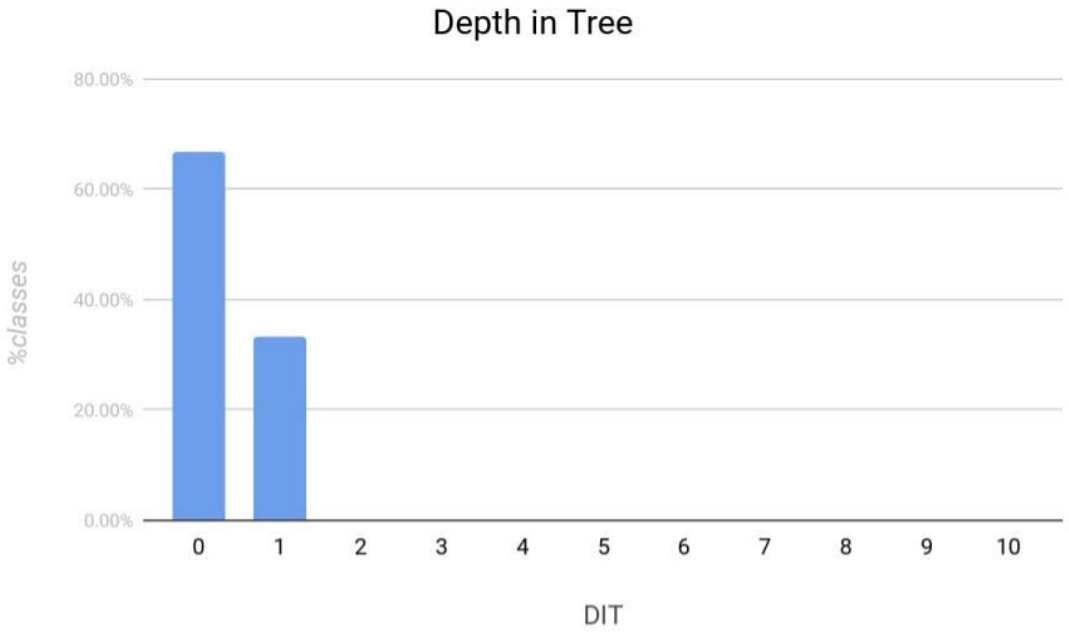


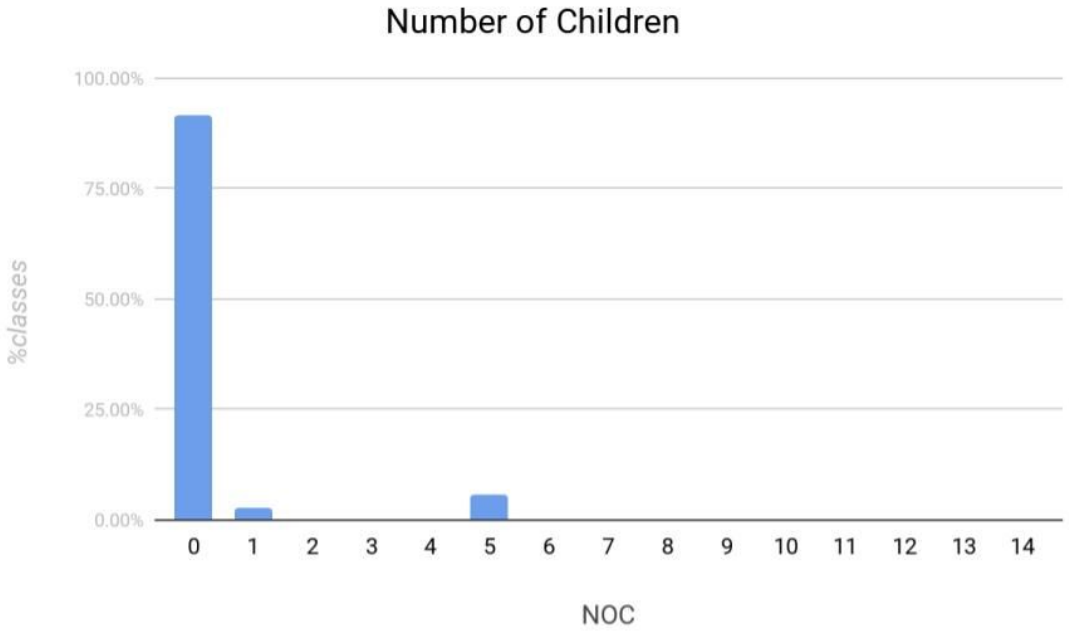
- **Cart page**

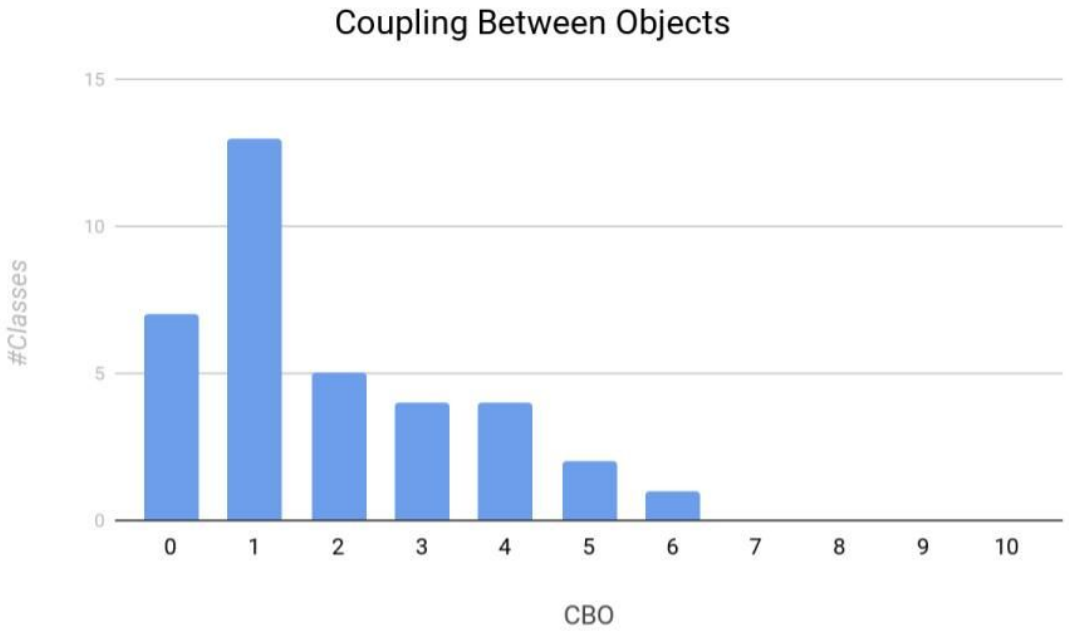
- **Payment page**

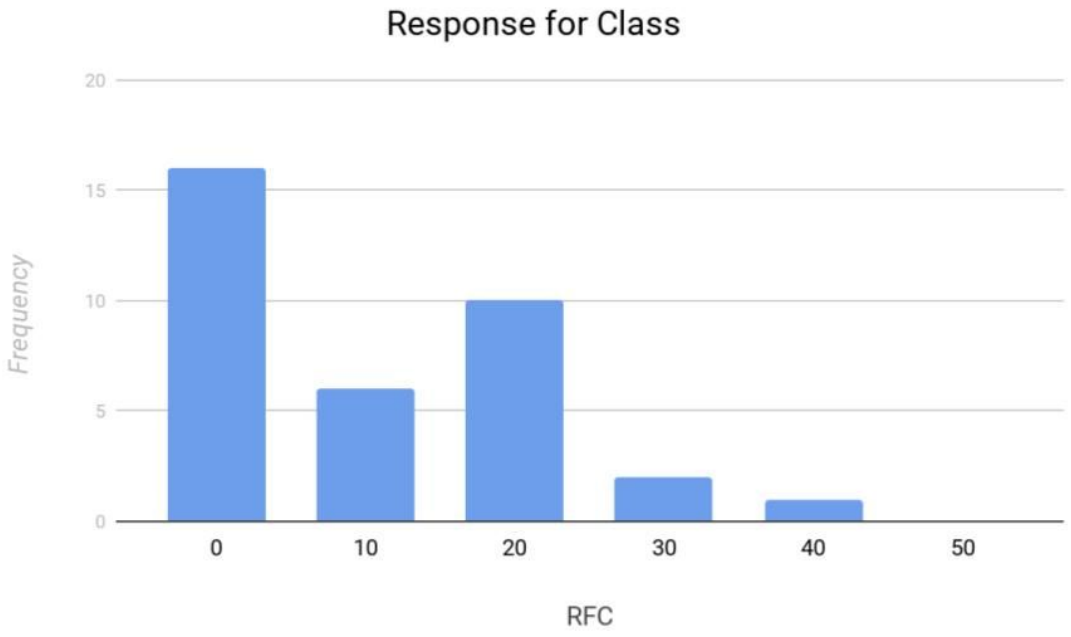
2) You need to evaluate the design quality by using object-oriented quality metrics (WMC, DIT, NOC, CBO, RFC, LCOM). The figure shall be drawn like the provided references below.

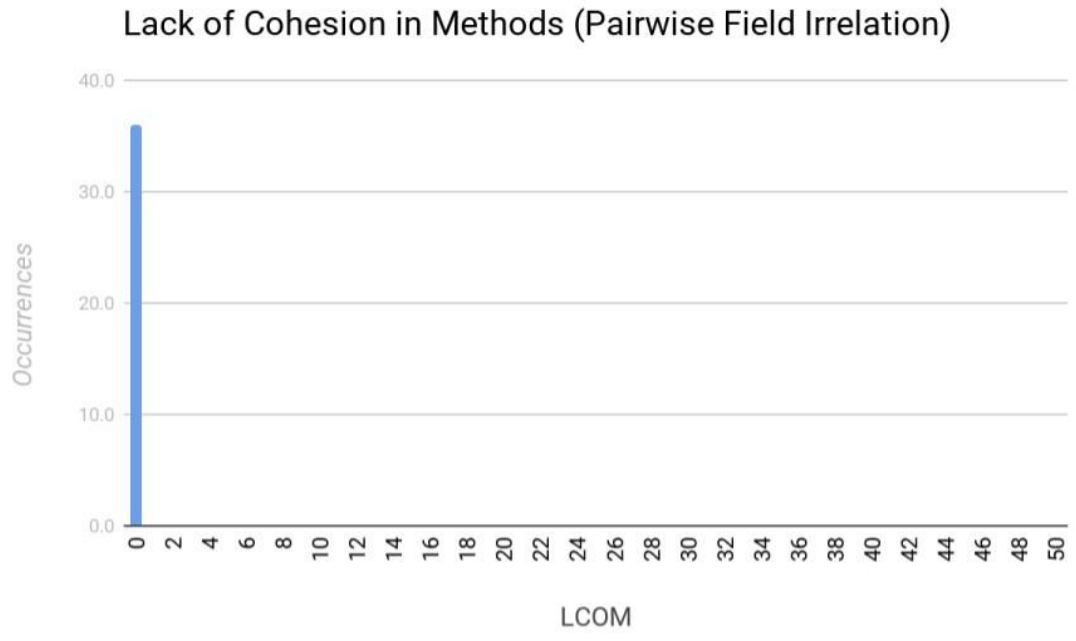

Weighted Methods Per Class
#classes
12
10
8
6
4
2
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
WMC


Depth in Tree
80.00%
60.00%
40.00%
20.00%
0.00%
%classes
0
1
2
3
4
5
6
7
8
9
10
DIT


Number of Children
100.00%
75.00%
50.00%
25.00%
0.00%
%classes
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
NOC


Coupling Between Objects
15
10
5
0
#Classes
0
1
2
3
4
5
6
7
8
9
10
CBO


Response for Class
Frequency
20
15
10
5
0
0
10
20
30
40
50
RFC


Lack of Cohesion in Methods (Pairwise Field Irrelation)
Occurrences
40.0
30.0
20.0
10.0
0.0
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
LCOM

3) Create Junit test cases and Junit test suite to test one selected class.

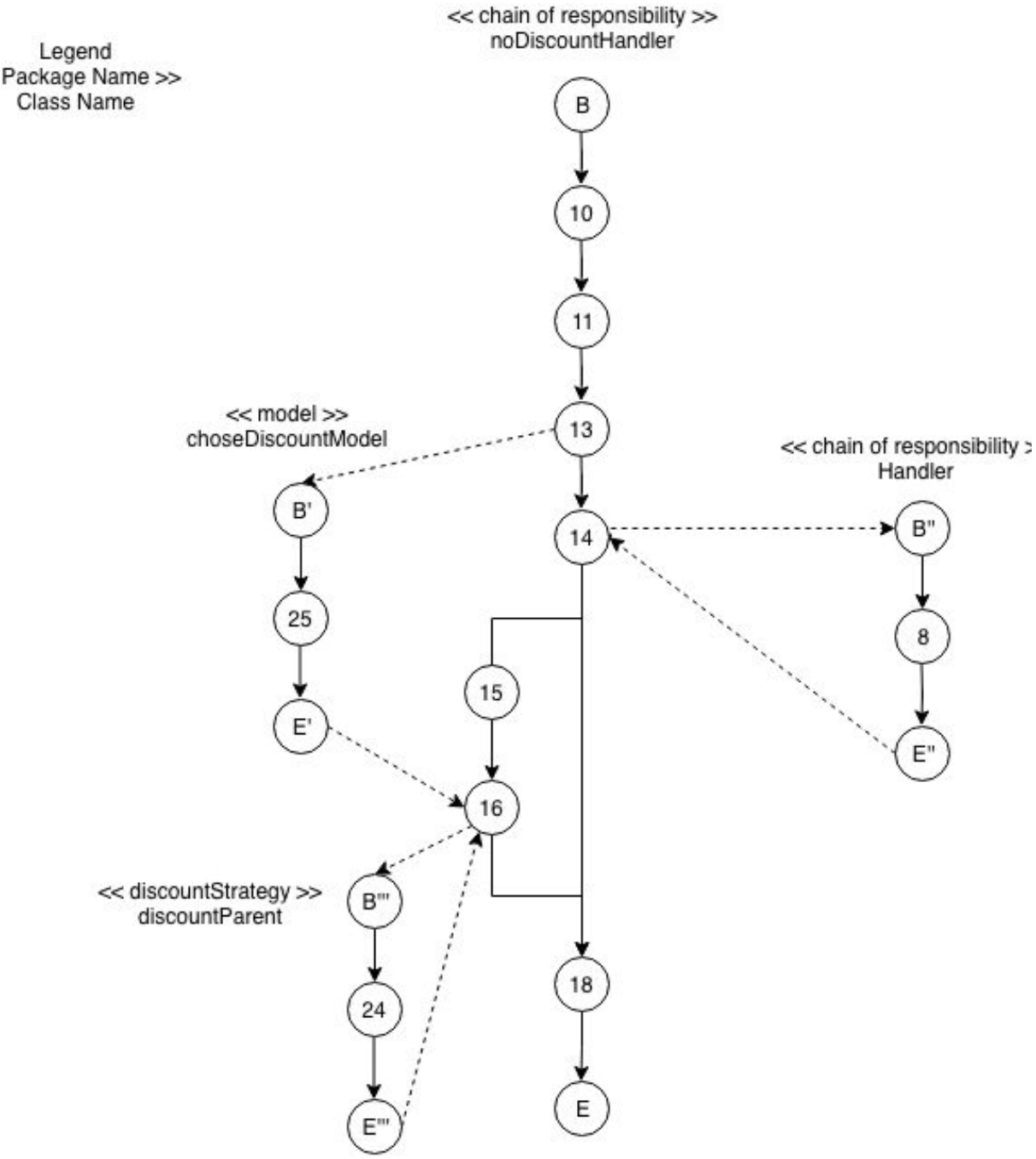
```
public void addtocart_nologin() {

    HttpServletRequest request = mock(HttpServletRequest.class);
    HttpServletResponse response = mock(HttpServletResponse.class);
    String TB_User_Acc = "noadmin";
    String TB_User_Pwd = "12345";
    response.sendRedirect("login");
    String User_ID = usersDAO.getUser_Infor(TB_User_Acc, TB_User_Pwd, "User_ID");
    String quantity = "2";
    String pid = "3";
    response.sendRedirect("addtocart");
}
```

```
public void addtocart_haslogin_noStock() {

    HttpServletRequest request = mock(HttpServletRequest.class);
    HttpServletResponse response = mock(HttpServletResponse.class);
    String TB_User_Acc = "admin";
    String TB_User_Pwd = "12345";
    response.sendRedirect("login");
    String User_ID = usersDAO.getUser_Infor(TB_User_Acc, TB_User_Pwd, "User_ID");
    String quantity = "2";
    String pid = "1"; //pid 1 hasn't stock
    response.sendRedirect("addtocart");

}
```

```
public void addtocart_haslogin_hasStock_noexist() {

    HttpServletRequest request = mock(HttpServletRequest.class);
    HttpServletResponse response = mock(HttpServletResponse.class);
    String TB_User_Acc = "admin";
    String TB_User_Pwd = "12345";
    response.sendRedirect("login");
    String User_ID = usersDAO.getUser_Infor(TB_User_Acc, TB_User_Pwd, "User_ID");
    String quantity = "2";
    String pid = "4"; //pid 4 has stock
    response.sendRedirect("addtocart");

}
```

```java
public void addtocart_haslogin_hasStock_exist() {

    HttpServletRequest request = mock(HttpServletRequest.class);
    HttpServletResponse response = mock(HttpServletResponse.class);
    String TB_User_Acc = "admin";
    String TB_User_Pwd = "12345";
    response.sendRedirect("login");
    String User_ID = usersDAO.getUser_Infor(TB_User_Acc, TB_User_Pwd, "User_ID");
    String quantity = "2";
    String pid = "4"; //pid 4 has stock
    response.sendRedirect("addtocart");
    quantity = "2";
    pid = "4"; //pid 4 has stock
    response.sendRedirect("addtocart");

}
```

4) **Conduct part of the software testing including white box and black box.**
    a) **While Box**
        i) **Flow graph**



```java
public class noDiscountHandler extends Handler{
    discountParent choseDiscount = null;
    ArrayList<cart> cartlist;
     public discountParent handlerRequest(choseDiscountModel CORM) {
         cartlist = CORM.getCartlist();
         if(super.getSuccessor()==null) {
                choseDiscount=new noDiscount();
                choseDiscount.setcartlist(cartlist);
        }
         return choseDiscount;
    }
}
```

```java
package model;
import java.util.*;
public class choseDiscountModel {
    private  String dateNowStr;
    private  int lastYearCost;
    private  ArrayList<cart> cartlist;
    public choseDiscountModel(String dateNowStr,int lastYearCost,ArrayList<cart> cartlist){
        this.dateNowStr = dateNowStr;
        this.lastYearCost = lastYearCost;
        this.cartlist = cartlist;
    }
    public String getDateNowStr() {
        return dateNowStr;
    }
    public void setDateNowStr(String dateNowStr) {
        this.dateNowStr = dateNowStr;
    }
    public int getLastYearCost() {
        return lastYearCost;
    }
    public void setLastYearCost(int lastYearCost) {
        this.lastYearCost = lastYearCost;
    }
    public ArrayList<cart> getCartlist() {
        return cartlist;
    }
}
```

```java
package chainOfResponsibility;
import discountStrategy.*;
import model.choseDiscountModel;
public abstract class Handler {
    protected Handler successor = null;

    public Handler getSuccessor() {
        return successor;
    }

    public void setSuccessor(Handler successor) {
        this.successor = successor;
    }

    public abstract discountParent handlerRequest(choseDiscountModel CORM);

}
```

```java
package discountStrategy;

import java.util.ArrayList;
import DAO.productsDAO;
import model.cart;

public abstract class discountParent {
    private int sumPrice;
    private  ArrayList<cart> cartlist;
    int afterDiscountPrice=0;
    public int getAfterDiscountPrice() {
        return afterDiscountPrice;
    }

    public void setAfterDiscountPrice(int afterDiscountPrice) {
        this.afterDiscountPrice = afterDiscountPrice;
    }

    public ArrayList<cart> getcartlist() {
        return cartlist;
    }

    public void setcartlist(ArrayList<cart> cartlist) {
        this.cartlist = cartlist;
    }
```

ii) Basis Paths

Path1: B 10 11 13 14 15 16 18 E

Path2: B 10 11 13 14 18 E

b) Black Box

i) Cause-Effect Testing

	1	2	3	4	5	6	7	8
login	N	N	N	N	Y	Y	Y	Y
in of stock	N	N	Y	Y	N	N	Y	Y
item(s) was already in your Cart	N	Y	N	Y	N	Y	N	Y

case count	1	1	1	1	1	1	1	1
add successfully							X	
add-fail	X	X	X	X	X	X		X

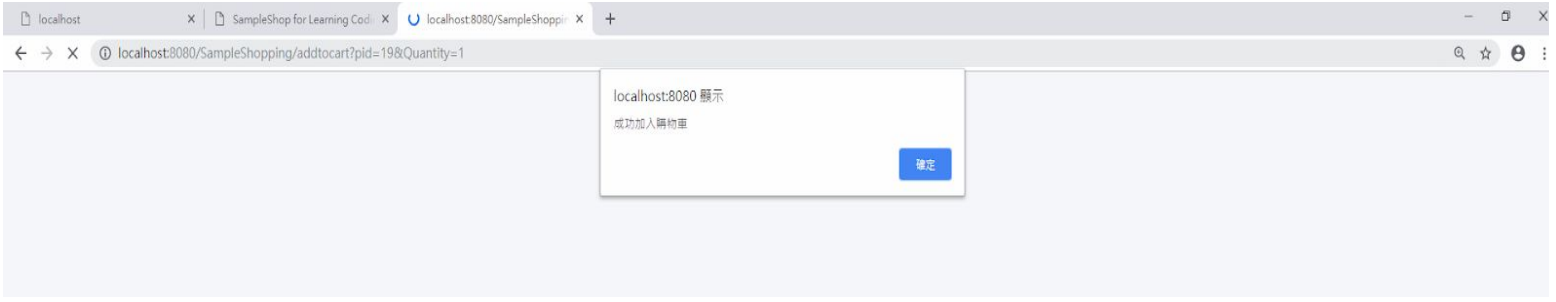
	1	2	3	4
login	N	Y	Y	Y
in of stock	-	N	Y	Y
item(s) was already in your Cart	-	-	N	Y
case count	4	2	1	1
add successfully			X	
add-fail	X	X		X

```
50    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
51        // TODO Auto-generated method stub
52        String quantity = String.valueOf(request.getParameter("Quantity"));
53        String pid =  String.valueOf(request.getParameter("pid"));
54        HttpSession session = request.getSession(true);
55        String User_ID = String.valueOf(session.getAttribute("User_ID"));
56        if ((User_ID!= null) || (User_ID!="")){
57            if(quantity.equals("0")) {
58                Systen.out.println("进入改订阅");
59                usersDAO.updateUser_SubcribePD(User_ID,pid);
60                Systen.out.println("进入改订阅完成");
61                response.setContentType("text/html;charset=UTF-8");
62                response.getWriter().println("<script>alert('商品已订阅'); window.location='PDlist.jsp' </script>");
63            }
64            else {
65                if (cartDAO.ishascart(User_ID, pid)) {
66                    response.setContentType("text/html ;charset=UTF-8");
67                    response.getWriter().println("<script>alert( 提醒：商品已经存在购物车'); window.location='PD.jsp?pid="+ pid +"' </scrip
68                }else{
69                    cartDAO.addcart(User_ID, pid, quantity);
70                    response.setContentType("text/html;charset=UTF-8");
71                    response.getWriter().println("<script>alert( 成功加入购物车'); window.location='PDlist.jsp' </script>");
72                }
73            }
74        }else{
75            response.setContentType("text/html;charset=UTF-8");
76            response.getWriter().println("<script>alert('请先登入会员'); window.location='PDlist.jsp' </script>");
77
78        }
```

ii) Boundary Value Analysis

m = # in stock before Customer's choice

No.	# of item(s) in stock	output
1	-1	Error
2	0	Message := 'sold out'
3	1	add successfully
4	m-1	add successfully
5	m	add successfully
6	m+1	Error
7	[0,m]	add successfully



iii) Deriving Test Cases

m = # in stock before Customer's choice
n = # in stock after checkout completed
Stock = 1, 2, … n , n<=m

case 1: n<=0
Input : Stock = (1,2,3,...m)
expected output : Message := 'sold out'

case 2: n=1 (After checkout completed, only one item in stock)
Input : Stock=(1,2,3,...m)
expected output: Stock= (1).

case 3.1: m>n (After checkout completed, the remainder is greater than one)
Input : Stock=(1, 2, 3,...m)
expected output: (1, 2,...n).

case 3.2: m=n (An order isn't completed for some unknown reason)
Input : Stock=(1,2,3,...m)
expected output: Stock=(1,2,3,...n).

5) Please analyze the invocation chains of your design.

a) public class addtocart extends HttpServlet

doGet(HttpServletRequest request, HttpServletResponse response)→ request.getParameter("Quantity")→ String.valueOf(request.getParameter("Quantity"))	2
doGet(HttpServletRequest request, HttpServletResponse response)→request.getParameter("pid")→ String.valueOf(request.getParameter("pid"))	2
doGet(HttpServletRequest request, HttpServletResponse response)→request.getSession(true)	1
doGet(HttpServletRequest request, HttpServletResponse response)→session.getAttribute("User_ID"))→ String.valueOf(session.getAttribute("User_ID"))	2
doGet(HttpServletRequest request, HttpServletResponse response)→usersDAO.updateUser_SubcribePD(User_ID,pid)	1
doGet(HttpServletRequest request, HttpServletResponse response)→response.setContentType("text/html;charset=UTF-8")	1
doGet(HttpServletRequest request, HttpServletResponse response)→response.getWriter().println("<script>alert('商品已訂閱'); window.location='PDlist.jsp' </script>")	1
doGet(HttpServletRequest request, HttpServletResponse response)→cartDAO.ishascart(User_ID, pid)→ response.setContentType("text/html ;charset=UTF-8")→response.getWriter().println("<script>alert('提醒：商品已經存在於購物車');window.location='PD.jsp?pid="+ pid +"' </script>")	3
doGet(HttpServletRequest request, HttpServletResponse response)→cartDAO.ishascart(User_ID, pid)→cartDAO.addcart(User_ID, pid, quantity)→ response.setContentType("text/html;charset=UTF-8")→response.getWriter().println("<script>alert('成功加入購物車'); window.location='PDlist.jsp' </script>")	4

doGet(HttpServletRequest request, HttpServletResponse response)→response.setContentType(**"text/html;charset=UTF-8")→response.getWriter().println("<script>alert('請先登入會員'); window.location='PDlist.jsp' </script>"**)	2
doPost(HttpServletRequest request, HttpServletResponse response)→doGet(request, response)	1

b) public class deletecartCommand implements command

execute()→cartDAO.getCartByShop_ID(Shop_ID)→DC.add(cartDAO.getCartByShop_ID(Shop_ID))→cartDAO.deleteCart(Shop_ID)	3

c) Count table (public class addtocart extends HttpServlet, public class deletecartCommand implements command)

Invocation Chain Length	1	2	3	4
Number of Chain	5	4	2	1

6) Score Sheet

Name	Score	accountabilities
Jenny	100%	slide,presenter, invocation chain
Jason	100%	code implementation, presenter
Kendy	100%	code implementation, presenter
Johnny	100%	presenter, material collection
Howard	100%	snapshots of the result, presenter
James	85%	audit the project
Timothy	100%	document, presenter, all tasks
Jerry	100%	Black box, presenter
Xavier	100%	White box, presenter
Alex	100%	White box, presenter