Systems Analysis and Design

Instructor : Huang, Chuen-Min

Teamwork2 ver.1

Group 9

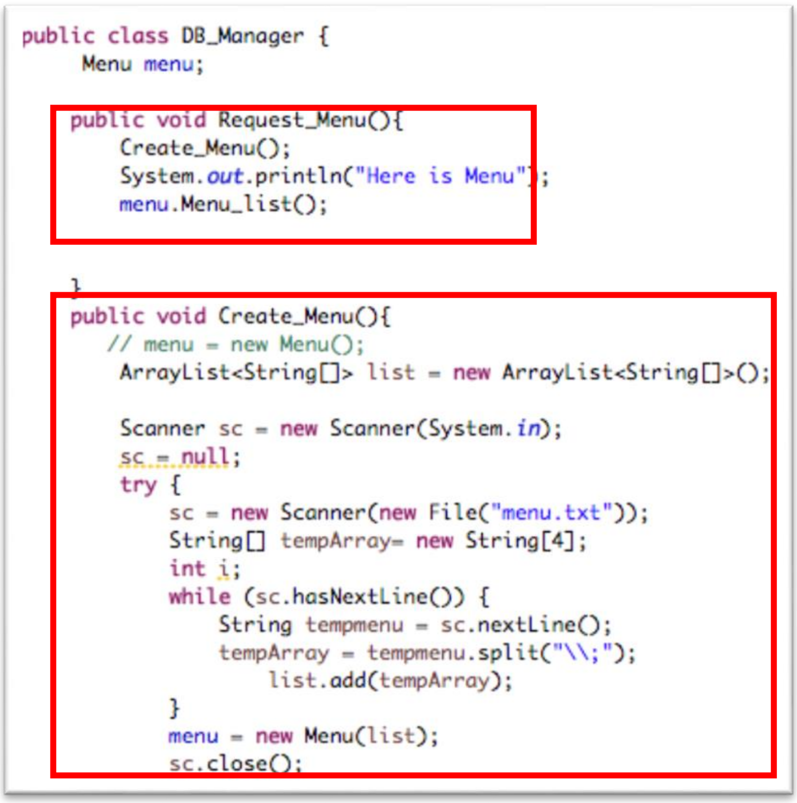
ID	Name
B10523001	Carol
B10523018	Jenny
B10423028	Tony
B10523019	Jason
B10423036	Anne
B10323037	Lulu
B10523023	Ken
B10523039	Jess
B10523051	Grace

Date 2018/ 05 /29

1. **Please explain the Law of Demeter (LoD) by using of your project.**

   (1) To itself.

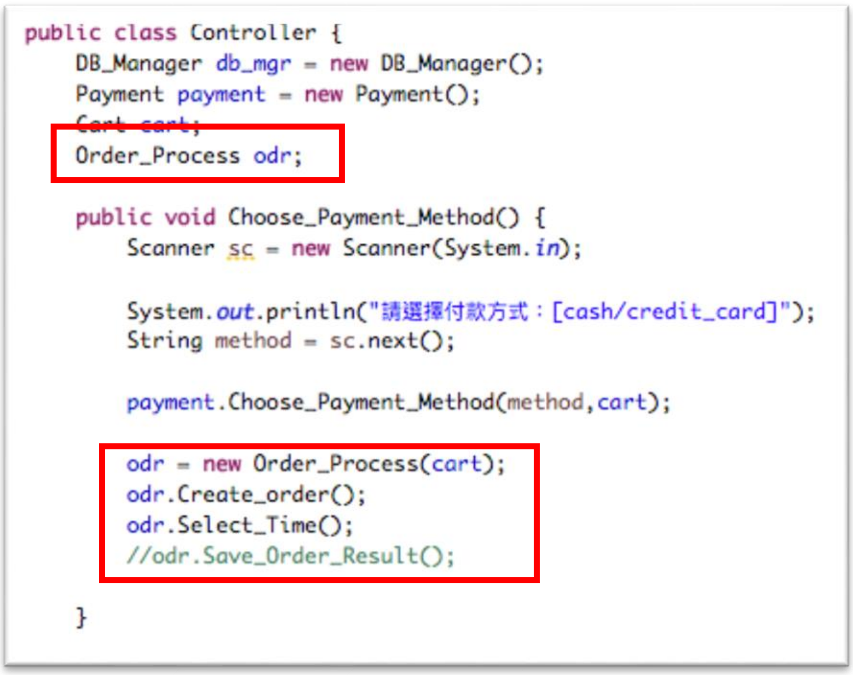
   When DB_Manager use Request_Menu method, then use create_Menu method to create a menu, and display its context.



```
public class DB_Manager {
     Menu menu;

    public void Request_Menu(){
        Create_Menu();
        System.out.println("Here is Menu");
        menu.Menu_list();

    }
    public void Create_Menu(){
       // menu = new Menu();
        ArrayList<String[]> list = new ArrayList<String[]>();

        Scanner sc = new Scanner(System.in);
        sc = null;
        try {
            sc = new Scanner(new File("menu.txt"));
            String[] tempArray= new String[4];
            int i;
            while (sc.hasNextLine()) {
                String tempmenu = sc.nextLine();
                tempArray = tempmenu.split("\\;");
                    list.add(tempArray);
            }
            menu = new Menu(list);
            sc.close();
```

(2) To objects contained in attributes of itself or a superclass.

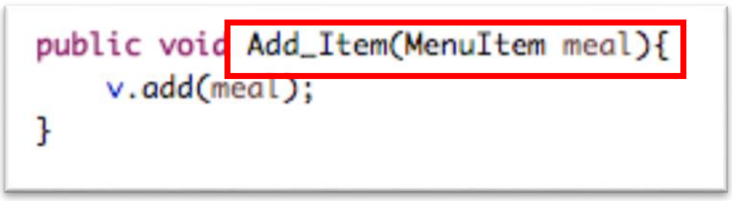
Controller has a object of order_process, it can use order_process method.



```
public class Controller {
    DB_Manager db_mgr = new DB_Manager();
    Payment payment = new Payment();
    Cart cart;
    Order_Process odr;

    public void Choose_Payment_Method() {
        Scanner sc = new Scanner(System.in);

        System.out.println("請選擇付款方式：[cash/credit_card]");
        String method = sc.next();

        payment.Choose_Payment_Method(method,cart);

        odr = new Order_Process(cart);
        odr.Create_order();
        odr.Select_Time();
        //odr.Save_Order_Result();

    }
```

(3) To an object that is passed as a parameter to the method.

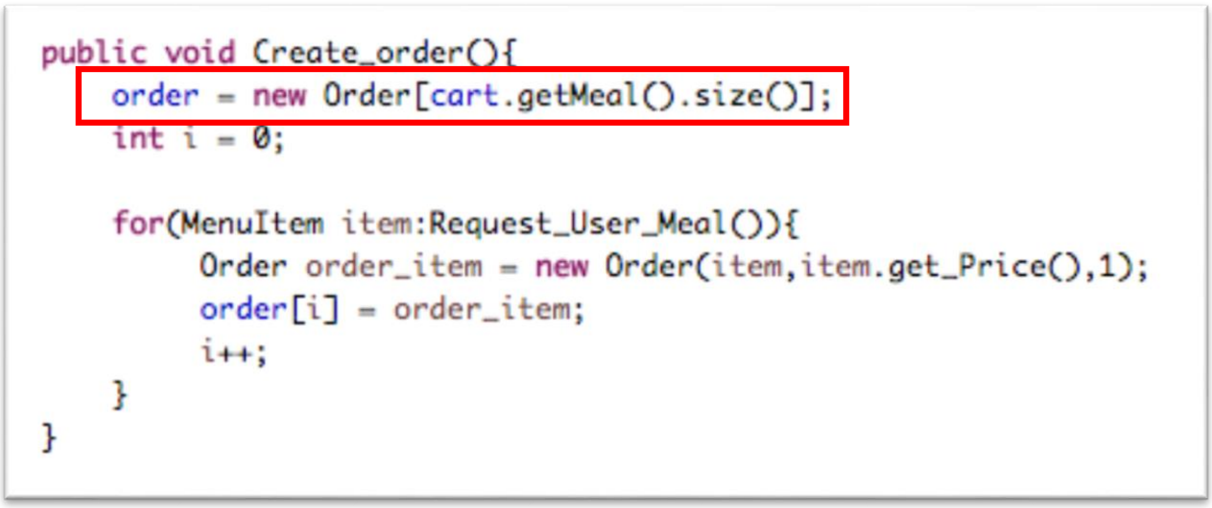
MenuIten as a parameter, pass to Add_item method, so it can add every MenuItem to Menu.



```
public void Add_Item(MenuItem meal){
    v.add(meal);
}
```

(4) To an object that is created by the method.

Create_order method create a Order object in order_process.

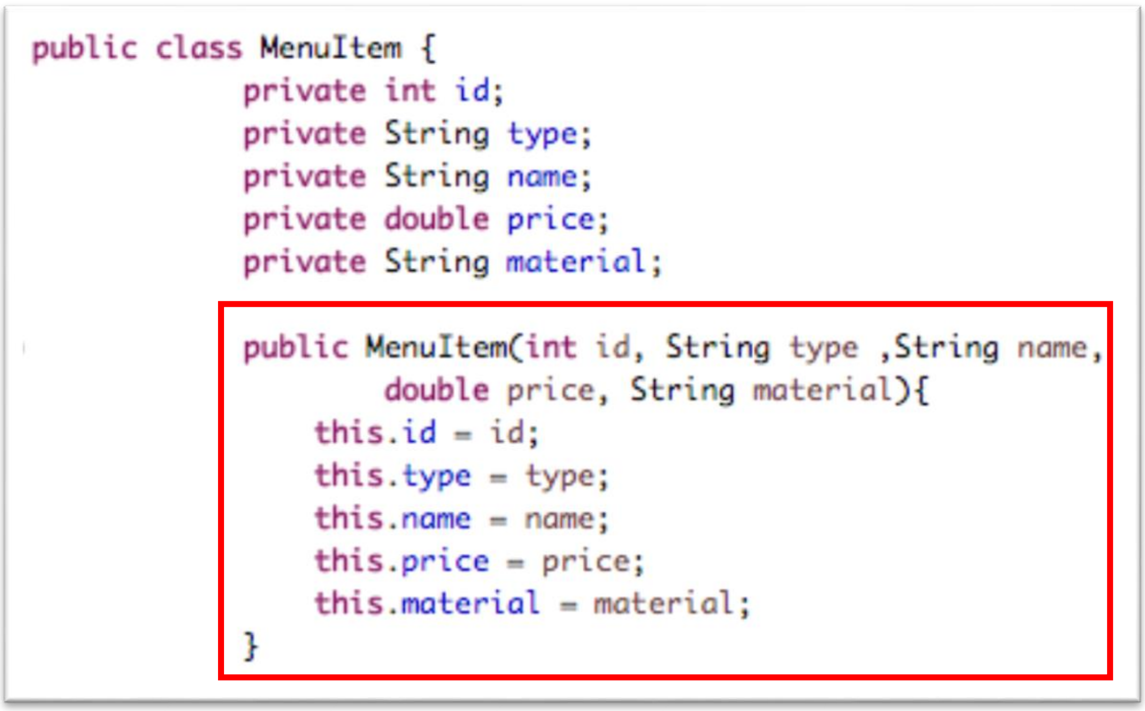


```
public void Create_order(){
    order = new Order[cart.getMeal().size()];
    int i = 0;

    for(MenuItem item:Request_User_Meal()){
        Order order_item = new Order(item,item.get_Price(),1);
        order[i] = order_item;
        i++;
    }
}
```

2. **There are six (or seven) types of interaction coupling, each falling on different parts of a good-to-bad continuum. Choose three pieces of your project to describe what types of the coupling they belong to.**

(1) Interaction coupling：data

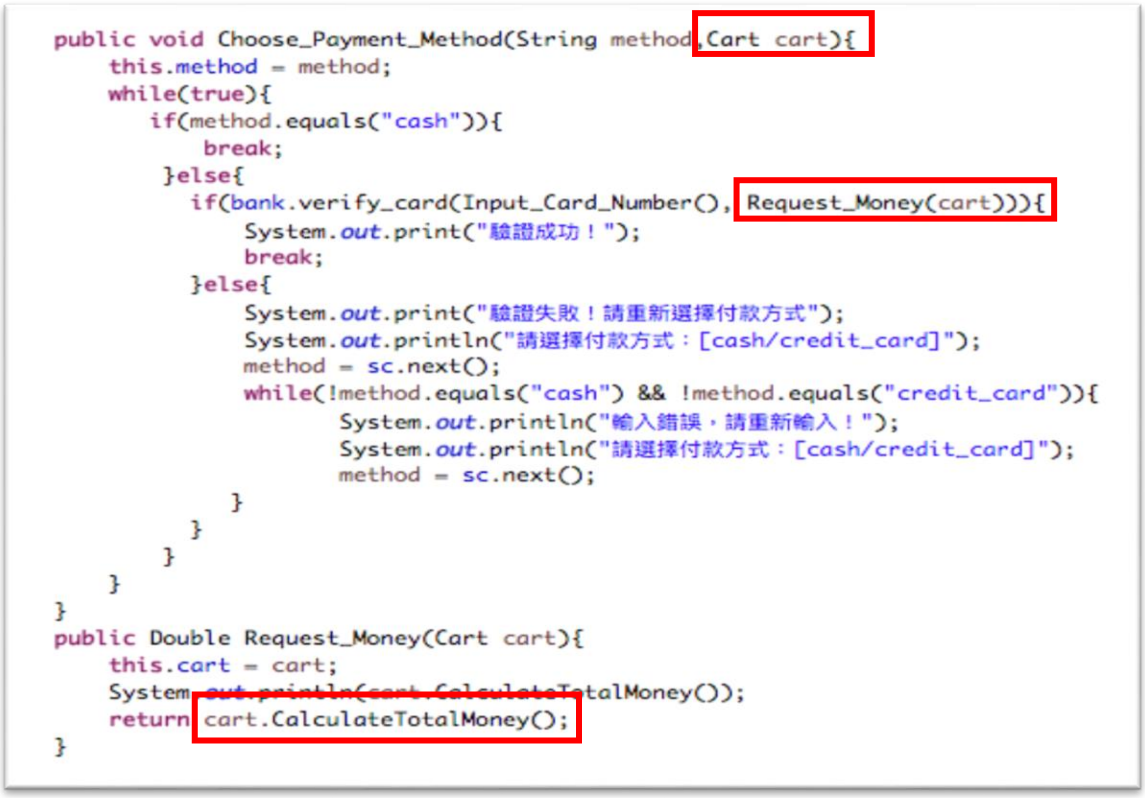
The calling method passed variables(id, type, name, price, material) to the called method. All the passed variables are be used. So it is data type.



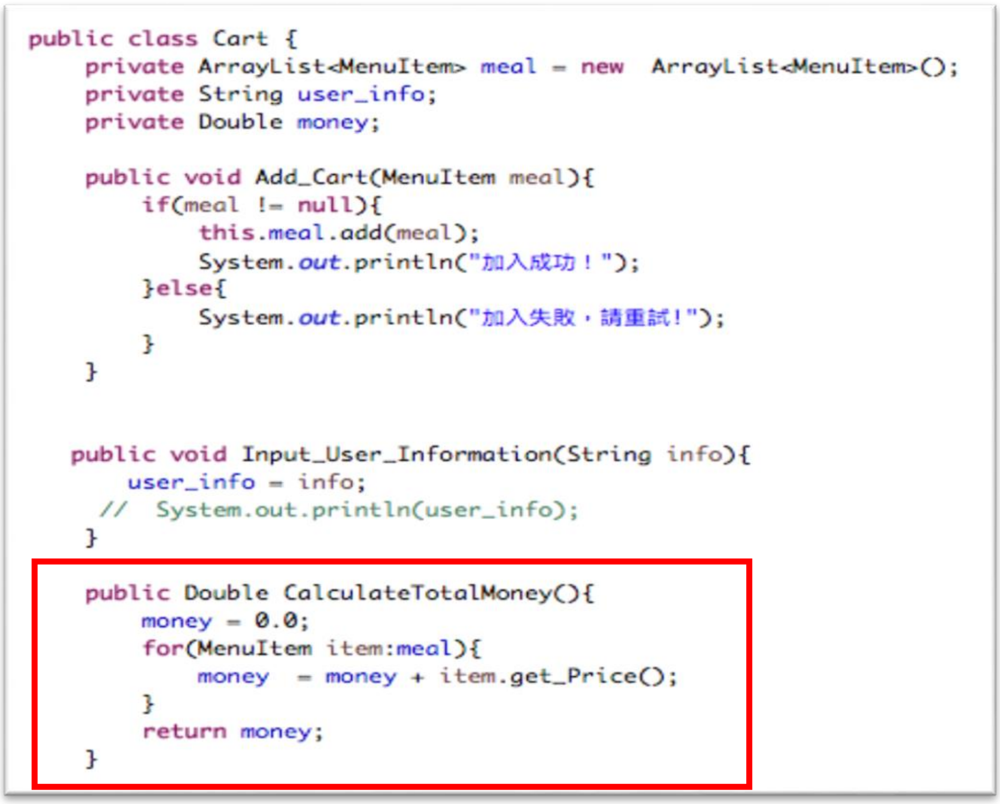
```
public class MenuItem {
            private int id;
            private String type;
            private String name;
            private double price;
            private String material;

            public MenuItem(int id, String type ,String name,
                    double price, String material){
                this.id = id;
                this.type = type;
                this.name = name;
                this.price = price;
                this.material = material;
            }
```

(2) Interaction coupling：stamp

The attribute Cart defined by Choose_Payment_Method will be used by function Request_Money, but we only use CalculateTotalMoney of Cart in Request_Money.



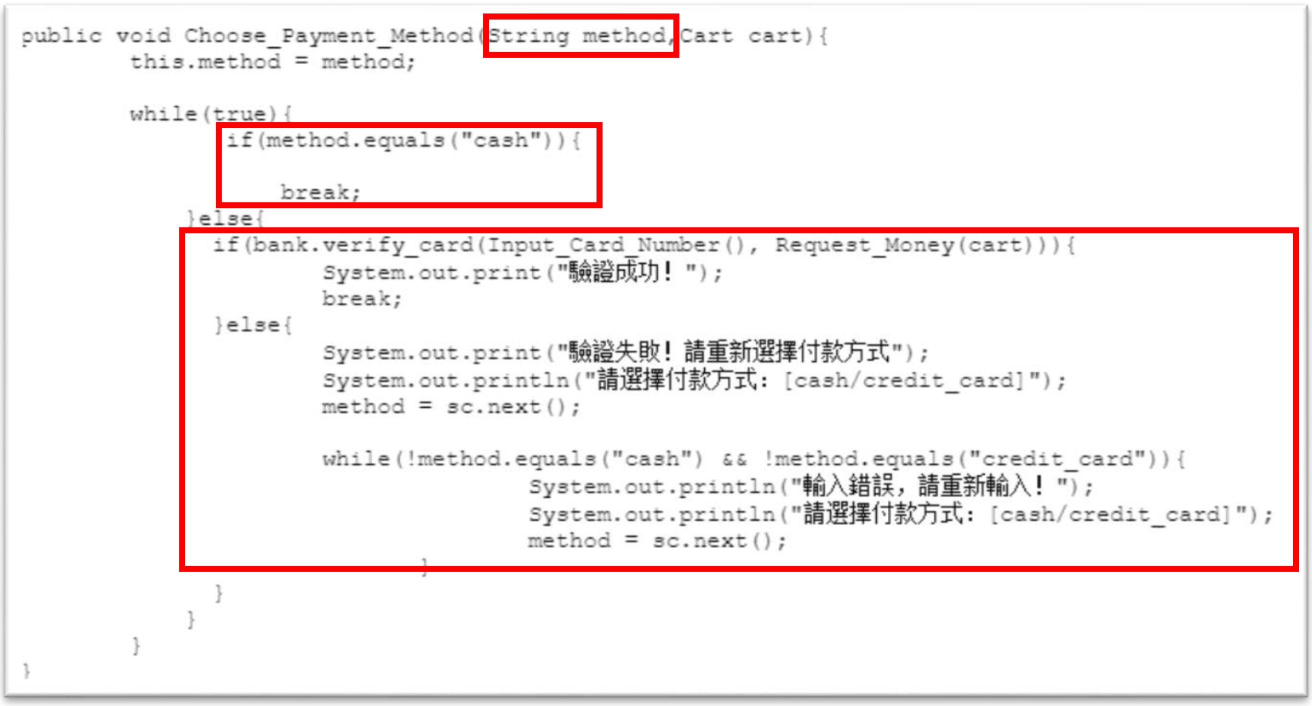
```java
public void Choose_Payment_Method(String method,Cart cart){
    this.method = method;
    while(true){
        if(method.equals("cash")){
            break;
        }else{
            if(bank.verify_card(Input_Card_Number(), Request_Money(cart))){
                System.out.print("驗證成功！");
                break;
            }else{
                System.out.print("驗證失敗！請重新選擇付款方式");
                System.out.println("請選擇付款方式：[cash/credit_card]");
                method = sc.next();
                while(!method.equals("cash") && !method.equals("credit_card")){
                        System.out.println("輸入錯誤，請重新輸入！");
                        System.out.println("請選擇付款方式：[cash/credit_card]");
                        method = sc.next();
                }
            }
        }
    }
}
public Double Request_Money(Cart cart){
    this.cart = cart;
    System.out.println(cart.CalculateTotalMoney());
    return cart.CalculateTotalMoney();
}
```



```java
public class Cart {
    private ArrayList<MenuItem> meal = new  ArrayList<MenuItem>();
    private String user_info;
    private Double money;

    public void Add_Cart(MenuItem meal){
        if(meal != null){
            this.meal.add(meal);
            System.out.println("加入成功！");
        }else{
            System.out.println("加入失敗，請重試!");
        }
    }


   public void Input_User_Information(String info){
       user_info = info;
    //  System.out.println(user_info);
    }

    public Double CalculateTotalMoney(){
        money = 0.0;
        for(MenuItem item:meal){
            money  = money + item.get_Price();
        }
        return money;
    }
```

(3) Interaction coupling：control

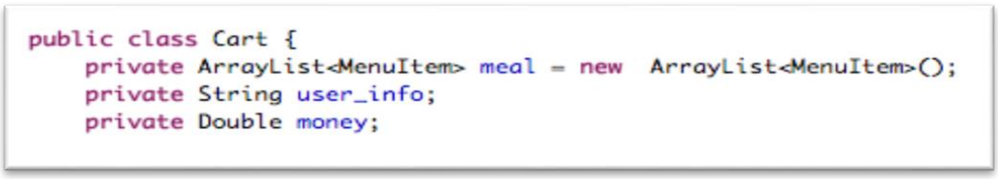
The method comes into Choose_Payment_Method will affect the next step of the program.



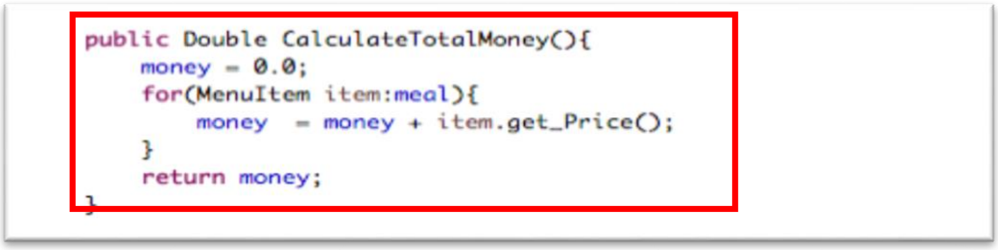
```
public void Choose_Payment_Method(String method,Cart cart){
        this.method = method;

        while(true){
                if(method.equals("cash")){

                    break;
            }else{
              if(bank.verify_card(Input_Card_Number(), Request_Money(cart))){
                      System.out.print("驗證成功！ ");
                      break;
              }else{
                      System.out.print("驗證失敗！ 請重新選擇付款方式");
                      System.out.println("請選擇付款方式: [cash/credit_card]");
                      method = sc.next();

                      while(!method.equals("cash") && !method.equals("credit_card")){
                                      System.out.println("輸入錯誤, 請重新輸入! ");
                                      System.out.println("請選擇付款方式: [cash/credit_card]");
                                      method = sc.next();
                      }
              }
            }
        }
}
```

3. There are seven types of method cohesion, choose three pieces of your project to describe what types of the coupling they belong to.

(1) Functional：The CalculateTotalMeony method of class Cart only calculates total money of order.

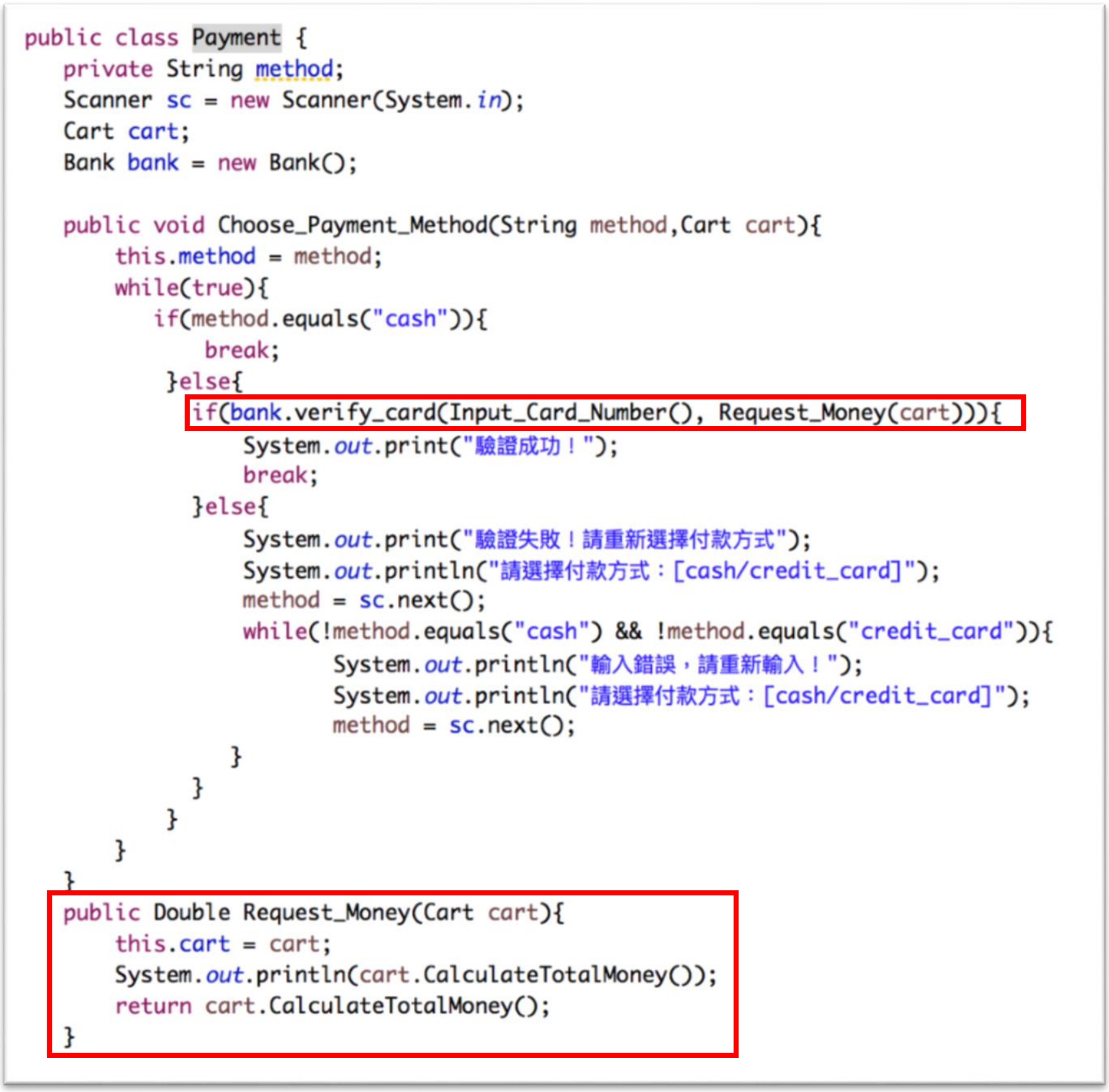


```
public class Cart {
    private ArrayList<MenuItem> meal = new  ArrayList<MenuItem>();
    private String user_info;
    private Double money;
```



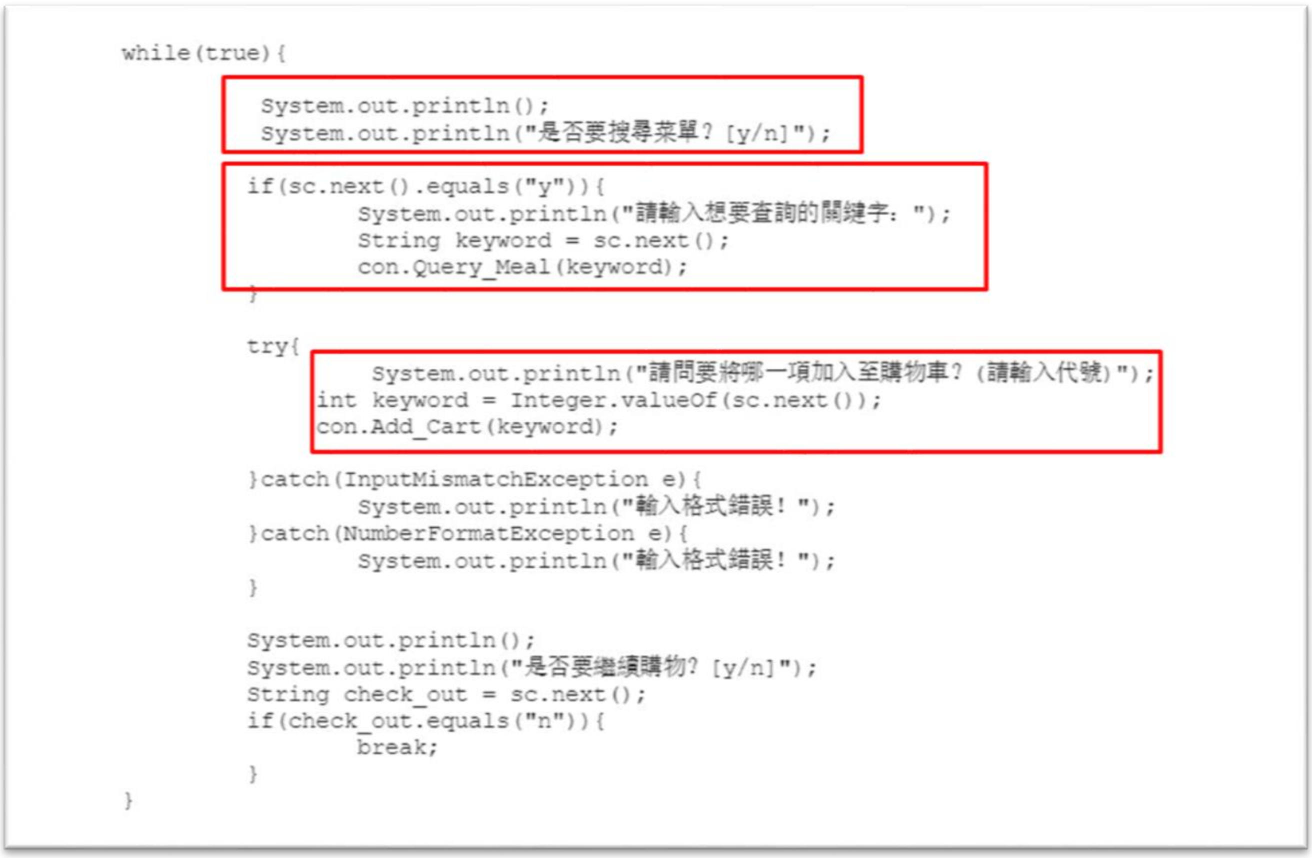
```
    public Double CalculateTotalMoney(){
        money = 0.0;
        for(MenuItem item:meal){
            money  = money + item.get_Price();
        }
        return money;
    }
```

(2) Sequence：The Choose_Payment_Method() in Payment will transfer the Cart into Request_Money(). And it will get the total price of the order from the Cart then output the price to BANK.



```java
public class Payment {
    private String method;
    Scanner sc = new Scanner(System.in);
    Cart cart;
    Bank bank = new Bank();

    public void Choose_Payment_Method(String method,Cart cart){
        this.method = method;
        while(true){
            if(method.equals("cash")){
                break;
            }else{
              if(bank.verify_card(Input_Card_Number(), Request_Money(cart))){
                  System.out.print("驗證成功！");
                  break;
              }else{
                  System.out.print("驗證失敗！請重新選擇付款方式");
                  System.out.println("請選擇付款方式：[cash/credit_card]");
                  method = sc.next();
                  while(!method.equals("cash") && !method.equals("credit_card")){
                          System.out.println("輸入錯誤，請重新輸入！");
                          System.out.println("請選擇付款方式：[cash/credit_card]");
                          method = sc.next();
                  }
              }
            }
        }
    }
    public Double Request_Money(Cart cart){
        this.cart = cart;
        System.out.println(cart.CalculateTotalMoney());
        return cart.CalculateTotalMoney();
    }
```

(3) Logical：The Choose_Payment_Method() in Controller can do a lot of thing, such as Choose payment method, create Order_Process and the order object, select pickup time and store the order information.

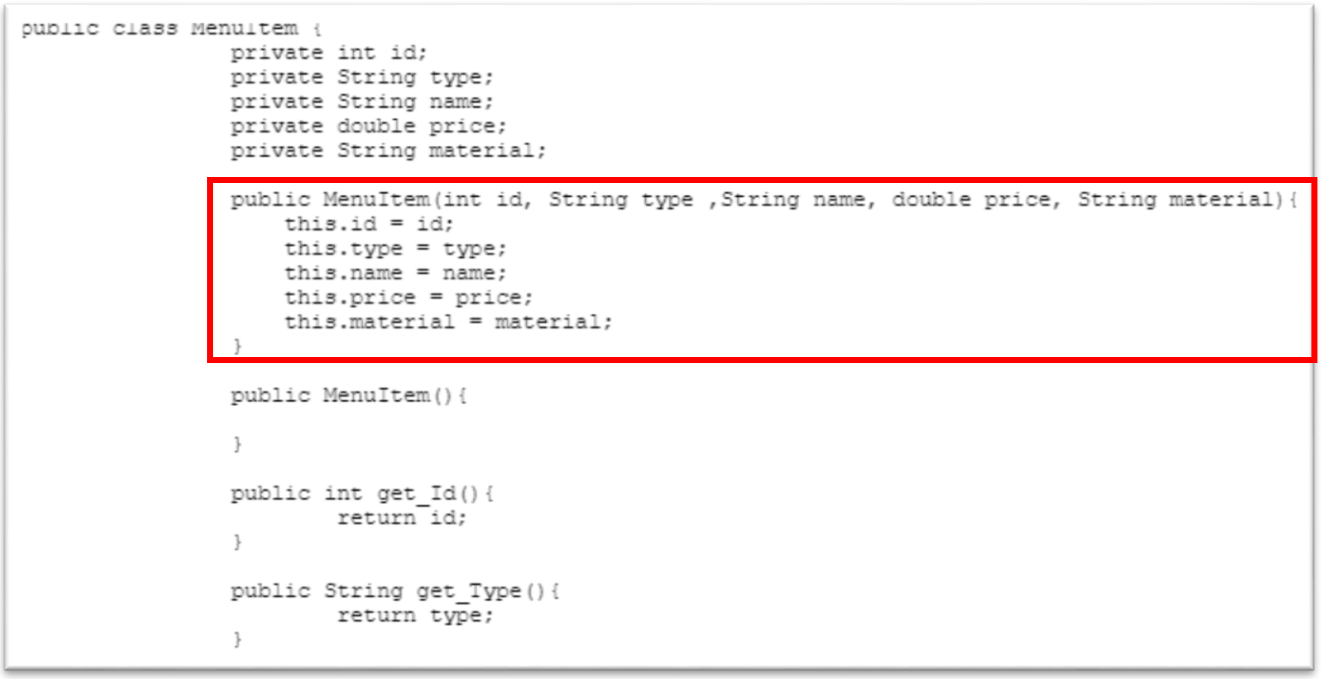


```
while(true){
          System.out.println();
          System.out.println("是否要搜尋菜單？[y/n]");
        if(sc.next().equals("y")){
                System.out.println("請輸入想要查詢的關鍵字：");
                String keyword = sc.next();
                con.Query_Meal(keyword);
        }
        try{
             System.out.println("請問要將哪一項加入至購物車？(請輸入代號)");
        int keyword = Integer.valueOf(sc.next());
        con.Add_Cart(keyword);
        }catch(InputMismatchException e){
                System.out.println("輸入格式錯誤！");
        }catch(NumberFormatException e){
                System.out.println("輸入格式錯誤！");
        }

        System.out.println();
        System.out.println("是否要繼續購物？[y/n]");
        String check_out = sc.next();
        if(check_out.equals("n")){
                break;
        }
}
```

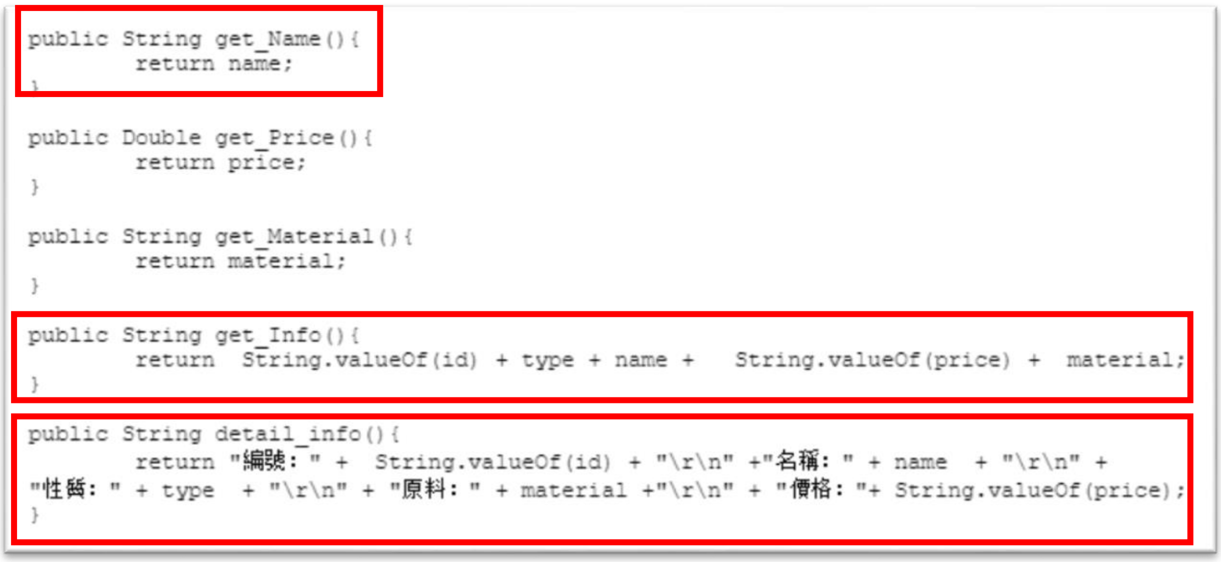
4. **Connascence generalized the ideas of cohesion and coupling, three pieces of your project to describe what types of the connascence they belong to.**

   (1) Name：

   If we change MenuItem attribute “name” to be another name like “ItemName”, then in method MenuItem、get_Name、get_Info and detail_Info we will have to change “name” to be “ItemName”, or it will can’t refer to this attribute.

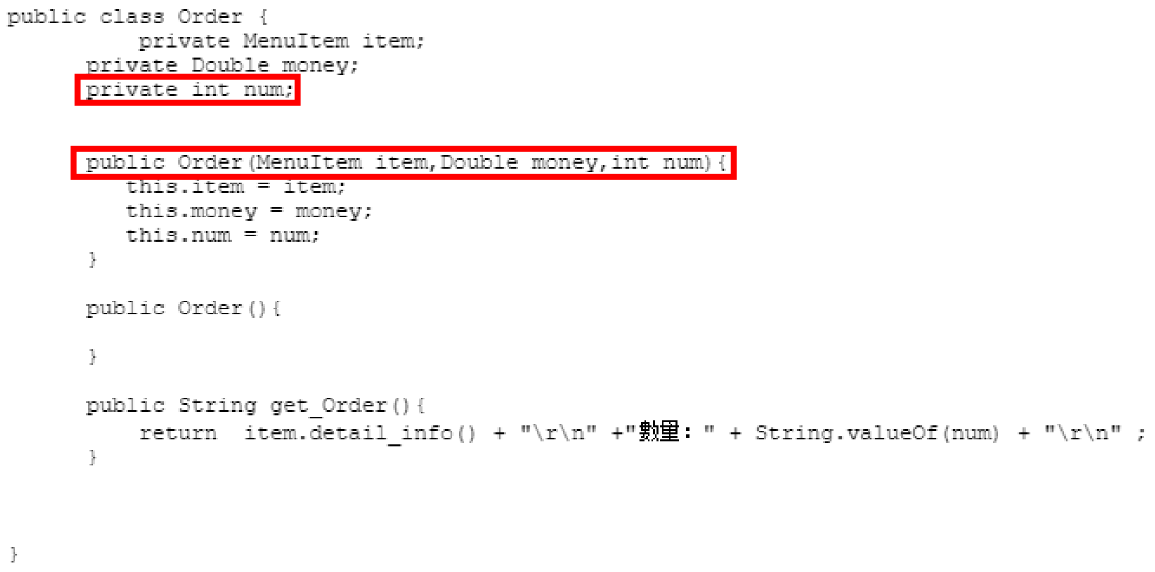


```
public class MenuItem {
                private int id;
                private String type;
                private String name;
                private double price;
                private String material;

                public MenuItem(int id, String type ,String name, double price, String material){
                    this.id = id;
                    this.type = type;
                    this.name = name;
                    this.price = price;
                    this.material = material;
                }

                public MenuItem(){

                }

                public int get_Id(){
                        return id;
                }

                public String get_Type(){
                        return type;
                }
```



```
public String get_Name(){
        return name;
}

public Double get_Price(){
        return price;
}

public String get_Material(){
        return material;
}

public String get_Info(){
        return  String.valueOf(id) + type + name +   String.valueOf(price) +  material;
}

public String detail_info(){
        return "編號: " +  String.valueOf(id) + "\r\n" +"名稱: " + name  + "\r\n" +
"性質: " + type  + "\r\n" + "原料: " + material +"\r\n" + "價格: "+ String.valueOf(price);
}
```

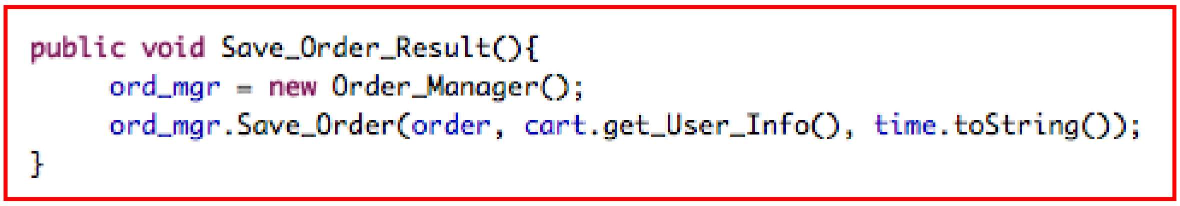
(2) Type：If we change type of num “int” to “Double”, the attribute declaration will have to change to “Double”.



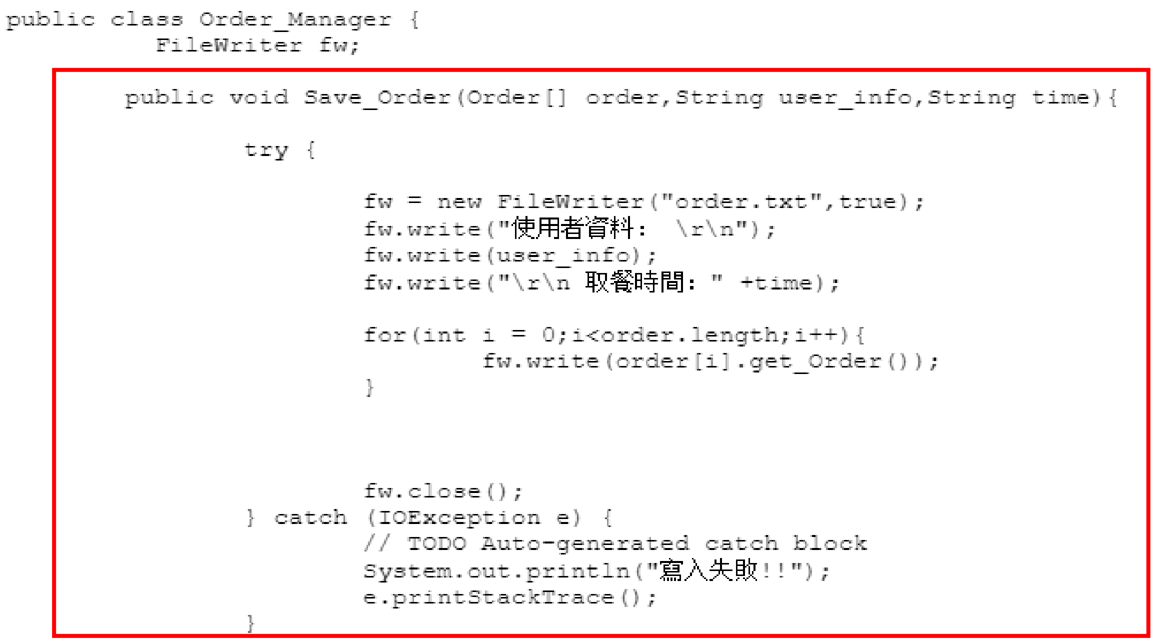
```
public class Order {
        private MenuItem item;
    private Double money;
    private int num;

    public Order(MenuItem item,Double money,int num){
        this.item = item;
        this.money = money;
        this.num = num;
    }

    public Order(){

    }

    public String get_Order(){
        return  item.detail_info() + "\r\n" +"數量: " + String.valueOf(num) + "\r\n" ;
    }


}
```

(3) Connascence：Position

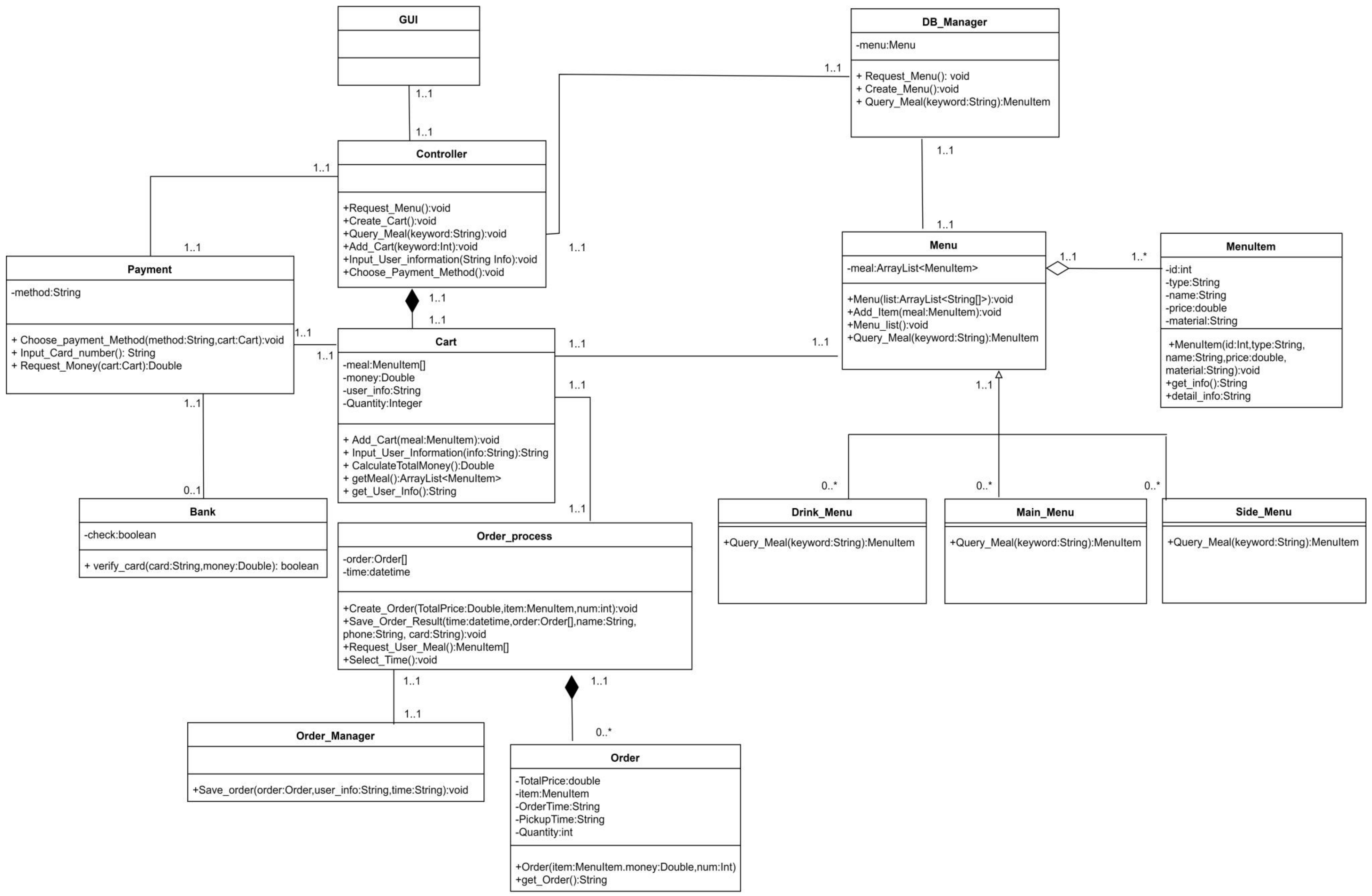
If the variable transfer into Save_Order() has wrong sequence when executing Save_Order_Result(), Save_Order() will not running correctly.



```
public void Save_Order_Result(){
    ord_mgr = new Order_Manager();
    ord_mgr.Save_Order(order, cart.get_User_Info(), time.toString());
}
```



```
public class Order_Manager {
        FileWriter fw;
    public void Save_Order(Order[] order,String user_info,String time){

        try {

            fw = new FileWriter("order.txt",true);
            fw.write("使用者資料:  \r\n");
            fw.write(user_info);
            fw.write("\r\n 取餐時間: " +time);

            for(int i = 0;i<order.length;i++){
                fw.write(order[i].get_Order());
            }



            fw.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("寫入失敗!!");
            e.printStackTrace();
        }
```


GUI
1..1
1..1
Controller
+Request_Menu():void
+Create_Cart():void
+Query_Meal(keyword:String):void
+Add_Cart(keyword:Int):void
+Input_User_information(String Info):void
+Choose_Payment_Method():void
DB_Manager
-menu:Menu
+ Request_Menu(): void
+ Create_Menu():void
+ Query_Meal(keyword:String):MenuItem
1..1
1..1
1..1
1..1
1..1
Payment
-method:String
+ Choose_payment_Method(method:String,cart:Cart):void
+ Input_Card_number(): String
+ Request_Money(cart:Cart):Double
1..1
1..1
1..1
1..1
0..1
Bank
-check:boolean
+ verify_card(card:String,money:Double): boolean
1..1
1..1
Cart
-meal:MenuItem[]
-money:Double
-user_info:String
-Quantity:Integer
+ Add_Cart(meal:MenuItem):void
+ Input_User_Information(info:String):String
+ CalculateTotalMoney():Double
+ getMeal():ArrayList<MenuItem>
+ get_User_Info():String
1..1
1..1
1..1
1..1
Menu
-meal:ArrayList<MenuItem>
+Menu(list:ArrayList<String[]>):void
+Add_Item(meal:MenuItem):void
+Menu_list():void
+Query_Meal(keyword:String):MenuItem
1..1
1..*
MenuItem
-id:int
-type:String
-name:String
-price:double
-material:String
+MenuItem(id:Int,type:String,
name:String,price:double,
material:String):void
+get_info():String
+detail_info:String
1..1
0..*
0..*
0..*
Drink_Menu
+Query_Meal(keyword:String):MenuItem
Main_Menu
+Query_Meal(keyword:String):MenuItem
Side_Menu
+Query_Meal(keyword:String):MenuItem
Order_process
-order:Order[]
-time:datetime
+Create_Order(TotalPrice:Double,item:MenuItem,num:int):void
+Save_Order_Result(time:datetime,order:Order[],name:String,
phone:String, card:String):void
+Request_User_Meal():MenuItem[]
+Select_Time():void
1..1
1..1
1..1
0..*
Order_Manager
+Save_order(order:Order,user_info:String,time:String):void
Order
-TotalPrice:double
-item:MenuItem
-OrderTime:String
-PickupTime:String
-Quantity:int
+Order(item:MenuItem.money:Double,num:Int)
+get_Order():String

5. **Use one class from your project that can create a set of invariants and add them to the CRC card or the class diagram.**

Front:

Class Name: Order_Process	**ID:** 2	**Type:** Concrete, Domain

Description: This class gets all information of order, and input all information to Order, It also access DB from Order_manager.	**Associated Use Cases:** 1

Responsibilities	**Collaborators**
Create_Order	Order
Save_Order_Result	Order_Manager, Order
Request_User_Meal	Cart, Order
Select_Time	Controller

Back:

Attribute:

Order (1..1) (order)

time (1..1) (String)

money (1..1) (double) {money=Cart.calculatetotalmoney()}

item (1..*) (Menuitem) {item=Cart. getMeal()}

Quantity (1..1) (int) {num=Cart.GetQuantity()}

CustomerName (1..1) (String) {name=Cuetomer.GetName()}

CustomerPhone (1..1) (String) {phone= Cuetomer.GetPhone()}

Card_Number (1..1) (String) {Card= Cuetomer.GetCardNumber()}

Relationships:

Generalization(a-kind-of): ______________________________

Aggregation(has-part): Order {0..1}_

Other Association: Order_Manager{1..1} , Cart {1..1} ,
Controller {1..1} , GUI {1..1}

6. **Use a method of a class from your project that can create a contract and describe its algorithm specification. Specify the pre- or post- condition and use both Structured English and an activity diagram to specify the algorithm.**

Method Name: Choose_payment_Method	**Class Name:** payment	**ID:** 1
Client(consumers): controller		
Associated Use Case: Make payment		
Description of Responsibilities: User can select the payment method, then base on the selection will have different action.		
Arguments Received: String method Cart cart		
Type of Value Returned: String		
Pre-Conditions: Method is cash or credit card.		
Post-Conditions: IF method is cash, then goes to next step. IF method is credit card, then Bank will verify card number and balances.		

Method Name: Choose_payment_Method	**Class Name:** Payment	**ID:**
Contract ID:	**Programmer:** Jason	**Date Due:** 5/29
Programming Language: ☐Visual Basic ☐Smalltalk ☐C++ ☑Java		
Trigger/Events: Ready for check out.		
Arguments Received: **Data Type:**	**Note:**	
String	method	
Cart	cart	
Messages Sent & Arguments Passed: **ClassName.methodName:**	**Data Type**	**Notes:**
Payment.Input_Card_Number()	String	
Payment.Request_Money()	Double	
Bank.Verify_Card(String card_number, Double balances)	Boolean	
Arguments Returned: **Data Type:**	**Notes:**	

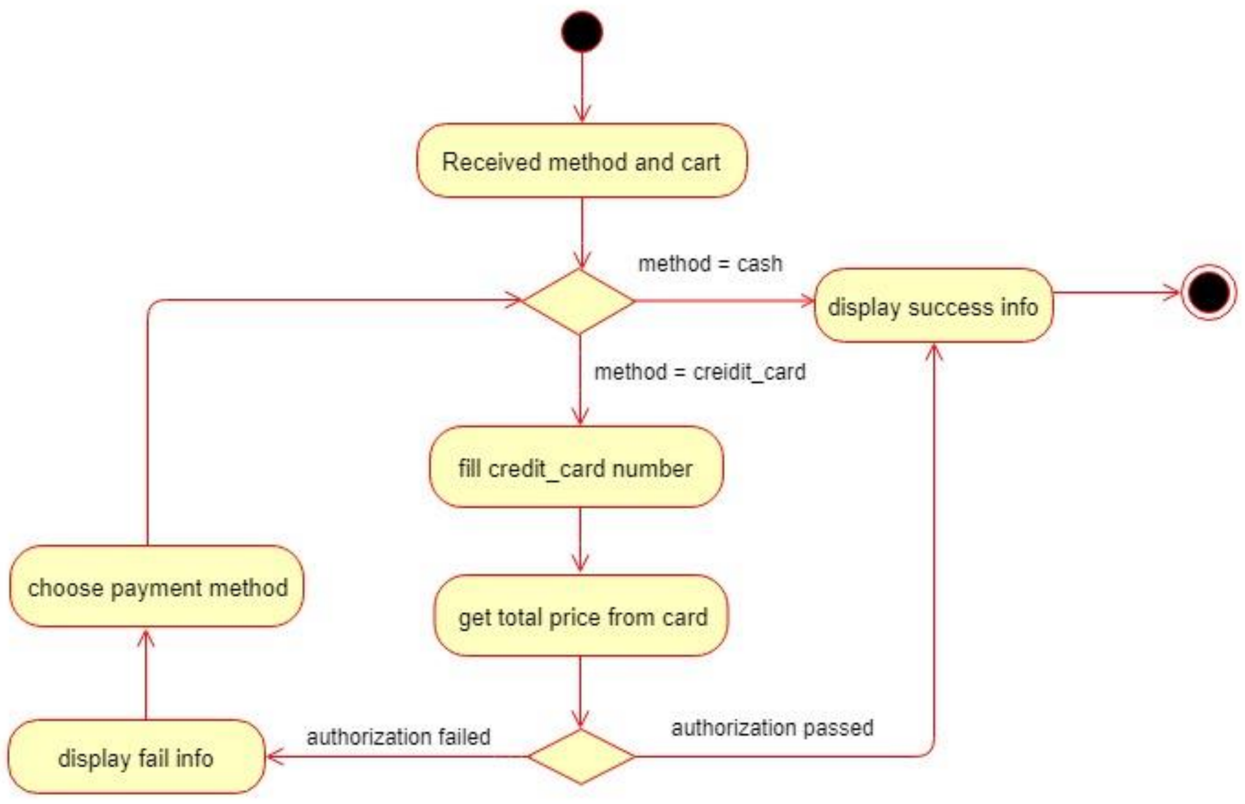
Algorithm Specification:

```
Received method from Controller, and receive cart from Cart.
Do
   IF (method==cash)
      Display success information
   ELSE IF (method=credit card)
      Fill credit_card number
      Get total price from Cart
         IF authorization fail THEN
            Display fail information
            Customer choose payment method
         ELSE
            Display success information
Until payment completed
      .
```

Misc Notes:

None

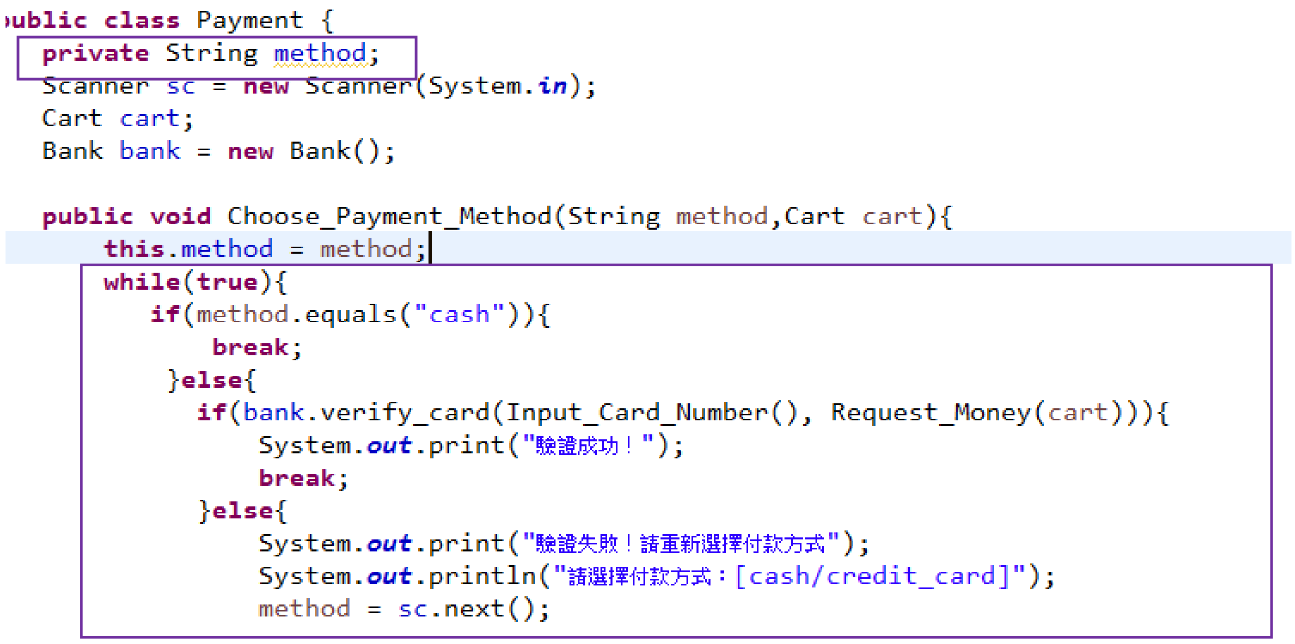
Activity diagram



7. **Please evaluate any piece of your project in terms of cohesion, coupling, and connascence perspective.**

- Coupling

Control Coupling

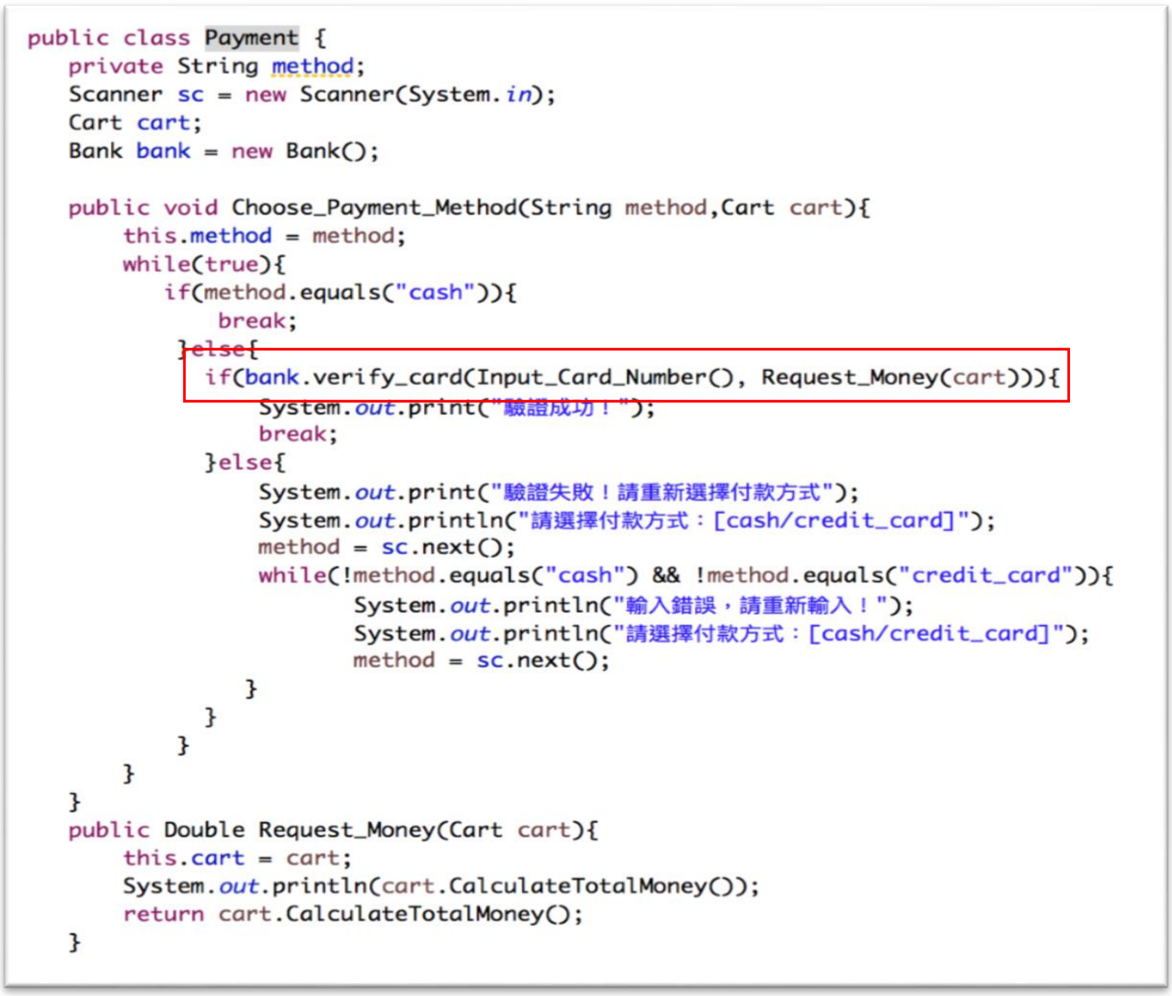


```
ublic class Payment {
   private String method;
   Scanner sc = new Scanner(System.in);
   Cart cart;
   Bank bank = new Bank();

   public void Choose_Payment_Method(String method,Cart cart){
       this.method = method;
       while(true){
          if(method.equals("cash")){
              break;
           }else{
             if(bank.verify_card(Input_Card_Number(), Request_Money(cart))){
                 System.out.print("驗證成功！");
                 break;
             }else{
                 System.out.print("驗證失敗！請重新選擇付款方式");
                 System.out.println("請選擇付款方式：[cash/credit_card]");
                 method = sc.next();
```

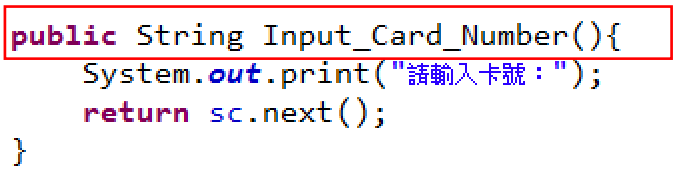
The method of Payment will have different action depends on the payment method customer has chosen. If the customer choose to pay by credit card, then the system will perform the verify_card in the bank.

cohesion：Sequence

The verify_card in Bank will validate the credit card according to the total price from Request_Money() and the card number from Input_Card_Number() in Payment.

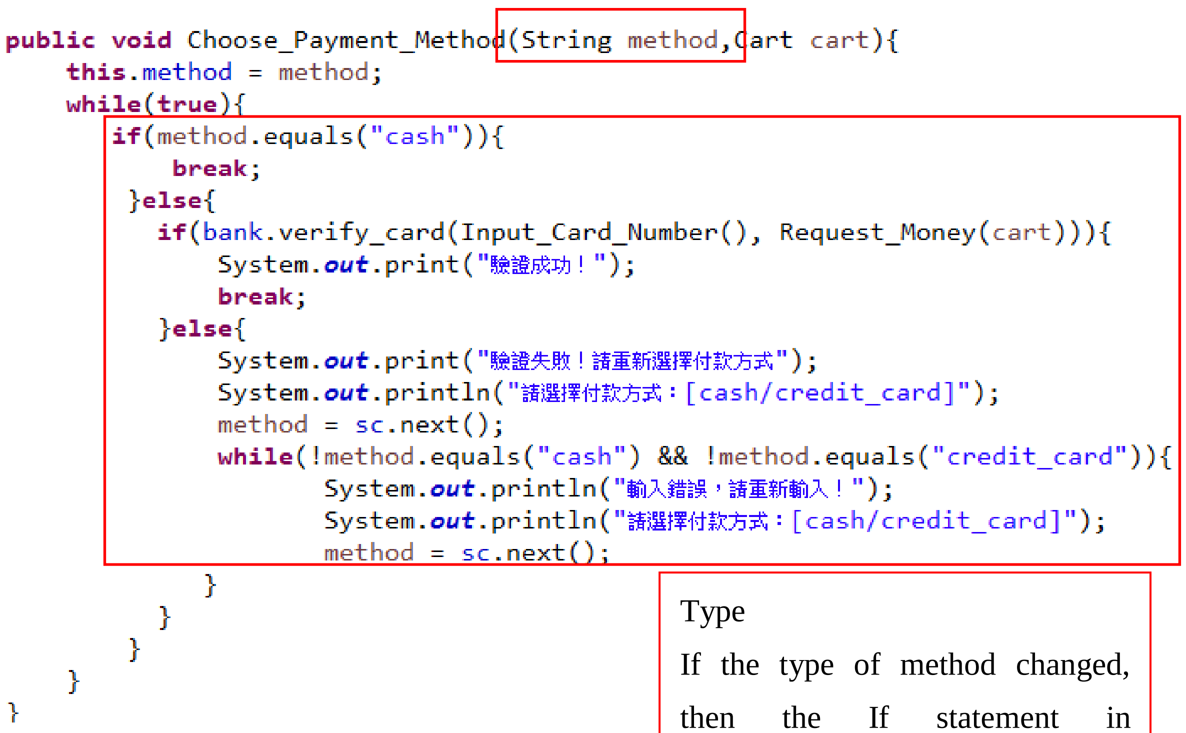


```
public class Payment {
   private String method;
   Scanner sc = new Scanner(System.in);
   Cart cart;
   Bank bank = new Bank();

   public void Choose_Payment_Method(String method,Cart cart){
       this.method = method;
       while(true){
          if(method.equals("cash")){
              break;
          }else{
             if(bank.verify_card(Input_Card_Number(), Request_Money(cart))){
                 System.out.print("驗證成功！");
                 break;
             }else{
                 System.out.print("驗證失敗！請重新選擇付款方式");
                 System.out.println("請選擇付款方式：[cash/credit_card]");
                 method = sc.next();
                 while(!method.equals("cash") && !method.equals("credit_card")){
                        System.out.println("輸入錯誤，請重新輸入！");
                        System.out.println("請選擇付款方式：[cash/credit_card]");
                        method = sc.next();
                 }
             }
          }
       }
   }
   public Double Request_Money(Cart cart){
       this.cart = cart;
       System.out.println(cart.CalculateTotalMoney());
       return cart.CalculateTotalMoney();
   }
```



```
public String Input_Card_Number(){
    System.out.print("請輸入卡號：");
    return sc.next();
}
```

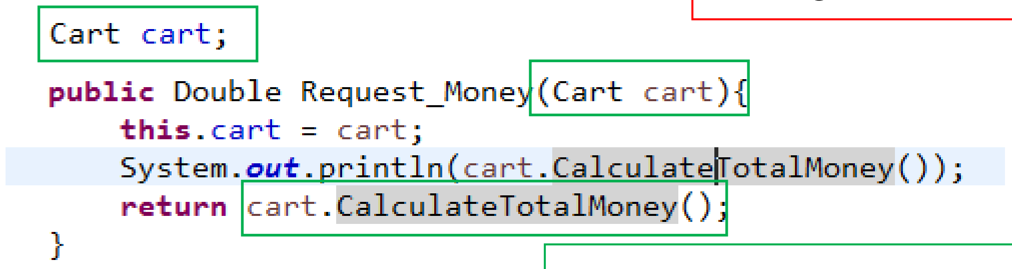
- Connascence



```
public void Choose_Payment_Method(String method,Cart cart){
    this.method = method;
    while(true){
        if(method.equals("cash")){
            break;
        }else{
          if(bank.verify_card(Input_Card_Number(), Request_Money(cart))){
              System.out.print("驗證成功！");
              break;
          }else{
              System.out.print("驗證失敗！請重新選擇付款方式");
              System.out.println("請選擇付款方式：[cash/credit_card]");
              method = sc.next();
              while(!method.equals("cash") && !method.equals("credit_card")){
                      System.out.println("輸入錯誤，請重新輸入！");
                      System.out.println("請選擇付款方式：[cash/credit_card]");
                      method = sc.next();
              }
          }
        }
    }
}
```

Type

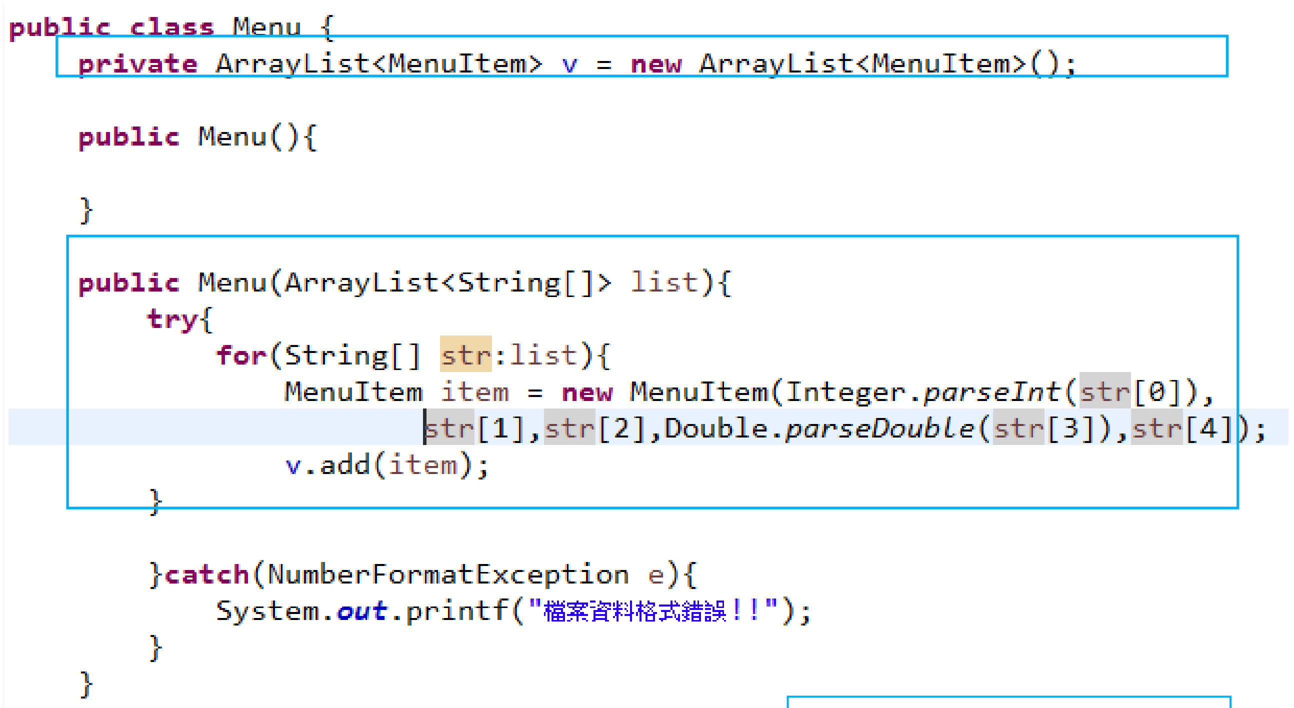
If the type of method changed, then the If statement in Choose_Payment_Method should be changed.



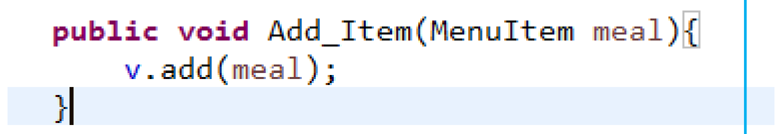
```
Cart cart;

public Double Request_Money(Cart cart){
    this.cart = cart;
    System.out.println(cart.CalculateTotalMoney());
    return cart.CalculateTotalMoney();
}
```

Name

If the name of Cart changed, then the content in Request_Money() should be changed.



```java
public class Menu {
    private ArrayList<MenuItem> v = new ArrayList<MenuItem>();

    public Menu(){

    }

    public Menu(ArrayList<String[]> list){
        try{
            for(String[] str:list){
                MenuItem item = new MenuItem(Integer.parseInt(str[0]),
                                 str[1],str[2],Double.parseDouble(str[3]),str[4]);
                v.add(item);
            }

        }catch(NumberFormatException e){
            System.out.printf("檔案資料格式錯誤！！");
        }
    }
```



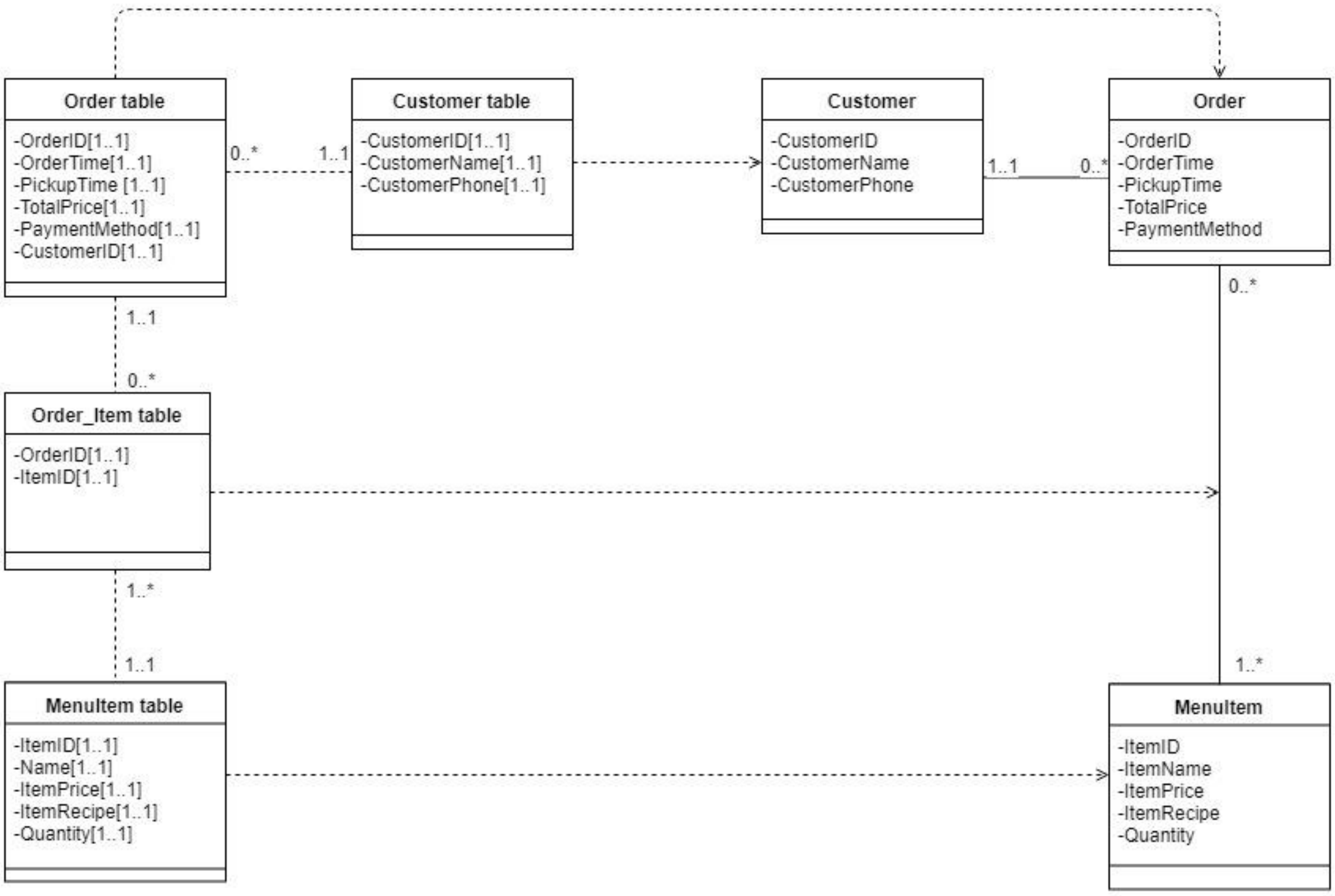
```java
    public void Add_Item(MenuItem meal){
        v.add(meal);
    }
```

Algorithm
If the structure of v in Menu() changed, then the method in add_item should be changed.

Evaluation: Our system has very high cohesion in method. Most of the method in the class focus on doing one thing. However, the coupling of our system was quite high. The Payment class will affect by the payment method choose by customer. For example, Use the method in Bank class or not. On the other hand, Order_process, Bank, Payment, Order class will use the data in Cart class which store the meal customer ordered. Moreover, the data source of Cart is DB_Manager, DB_Manager load the data from database then store in Menu class. So, if the data in database has failure, or the code of Menu and Cart has a mistake, will affect the operation and correctness of the whole system.

Improvement: We need to have some data validation to ensure that data will be correct during transferring, and prevent the logic error. On the other hand, we should add some exceptional process to handle the accidental error and ensure our system will no break down due to user’s misuse.

8. **Assume that you are going to adopt RDBMs to your project, please describe the referential integrity.**



Rule 1: One concrete class correspond a RDBMS table, we have 3 concrete class, so they correspond 3 RDBMS table.

Rule 2: Single-valued attribute should correspond the data column in RDBMS table.

Rule 3: Derived attribute won't appear in RDBMS table normally. We don't have derived attribute, so, doesn't make change.

Rule 4: In the problem domain class of One to One aggregation and association relationship, we should put the object id of counterpart in RDBMS table. In our case, we don't have One to One relationship, so doesn't make change.

Rule 5: Multi-valued attribute will create a new table. However, we don't have Multi-valued attribute.

Rule 6: In Many to Many relation will create a new table. The primary key of two of the original table will be put into the new table. Our Order and MenuItem was Many to Many relation, so, we create a new Order_Ite, table and place the primary key of Order and MenuItem.

Rule 7: In One to Many relation, the Many side will add a new column to store the primary key of the One side. Our Customer and Order was One to Many relation, so we put the primary key of Customer into Order table.
Rule 8a: We don’t have inheritance.
Rule 8b: We don’t have inheritance.

9. Using the steps of normalization, create a model that represents the file of your project in third normal form. Please make necessary assumptions to explain why the tables are related.

0NF

Order												
Order ID	Customer ID	Customer Name	Customer Phone	Order Time	Pickup Time	Payment Method	Total Price	Item ID	Item Name	Item Price	Item Recipe	Quantity
1	A119099070	Tony	0984394801	05/29 12:15	05/29 13:00	Cash	230	3	Orange juice	40	Water	2
								2	Cola	30	Water	1
								1	Soda	30	Water	1
								4	Bacon Burger	90	Meat	1
2	D266276735	Anne	0984562766	05/29 08:00	05/29 09:22	Credit card	30	2	Cola	30	Water	1
3	Z276769530	Tim	0987654321	05/30 09:50	05/30 10:10	Credit card	125	5	Double Cheese	110	Meat	1
								6	Ice cream	15	Dessert	1
4	M203328053	Carol	0987568953	05/30 09:58	05/30 13:20	Cash	180	1	Soda	30	Water	2
								4	Bacon Burger	90	Meat	1
								2	Cola	30	Water	1
5	Y199892270	Jenny	0956639856	06/01 08:55	06/01 10:25	Credit card	120	1	Soda	30	Water	1
								4	Bacon Burger	90	Meat	1

1NF

Order Table

*Order ID	Customer ID	Customer Name	Customer Phone	Order Time	Pickup Time	Payment Method	Total Price
1	A119099070	Tony	0984394801	05/29 12:15	05/29 13:00	Cash	230
2	D266276735	Anne	0984562766	05/29 08:00	05/29 09:22	Credit card	30
3	Z276769530	Tim	0987654321	05/30 09:50	05/30 10:10	Credit card	125
4	M203328053	Carol	0987568953	05/30 09:58	05/30 13:20	Cash	180
5	Y199892270	Jenny	0956639856	06/01 08:55	06/01 10:25	Credit card	120

Item Table

*Order ID	*Item ID	Item Name	Item Price	Item Recipe	Quantity
1	3	Orange juice	40	Water	2
1	2	Cola	30	Water	1
1	1	Soda	30	Water	1
1	4	Bacon Burger	90	Meat	1
2	2	Cola	30	Water	1
3	5	Double Cheese	110	Meat	1
3	6	Ice cream	15	Dessert	1
4	1	Soda	30	Water	2
4	4	Bacon Burger	90	Meat	1
4	2	Cola	30	Water	1
5	1	Soda	30	Water	1
5	4	Bacon Burger	90	Meat	1

2NF

Customer Table

*CustomerID	CustomerName	CustomerPhone
A119099070	Tony	0984394801
D266276735	Anne	0984562766
Z276769530	Tim	0987654321
M203328053	Carol	0987568953
Y199892270	Jenny	0956639856

Item Table

*ItemID	ItemName	ItemPrice	ItemRecipe
1	Soda	30	Water
2	Cola	30	Water
3	Orange juice	40	Water
4	Bacon Burger	90	Meat
5	Double Cheese	110	Meat
6	Ice cream	15	Dessert

Order_Item Table

*OrderID	*ItemID	Quantity
1	3	2
1	2	1
1	1	1
1	4	1
2	2	1
3	5	1
3	6	1
4	1	2
4	4	1
4	2	1
5	1	1
5	4	1

Order Table

*OrderID	CustomerID	OrderTime	PickupTime	TotalPrice	PaymentMethod
1	A119099070	05/29 12:15	05/29 13:00	230	Cash
2	D266276735	05/29 08:00	05/29 09:22	30	Credit card
3	Z276769530	05/30 09:50	05/30 10:10	125	Credit card
4	M203328053	05/30 09:58	05/30 13:20	180	Cash
5	Y199892270	06/01 08:55	06/01 10:25	120	Credit card

3NF

In 2NF, our table doesn’t have transitive functional dependency, so our 3NF is same as 2NF.

Customer Table

*CustomerID	CustomerName	CustomerPhone
A119099070	Tony	0984394801
D266276735	Anne	0984562766
Z276769530	Tim	0987654321
M203328053	Carol	0987568953
Y199892270	Jenny	0956639856

Item Table

*ItemID	ItemName	ItemPrice	ItemRecipe
1	Soda	30	Water
2	Cola	30	Water
3	Orange juice	40	Water
4	Bacon Burger	90	Meat
5	Double Cheese	110	Meat
6	Ice cream	15	Dessert

Order_Item Table

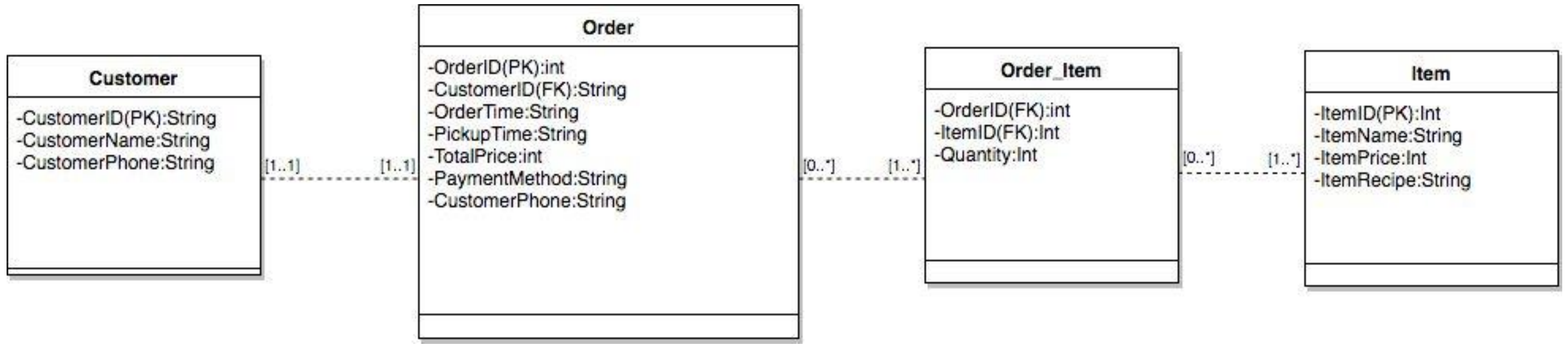
*OrderID	*ItemID	Quantity
1	3	2
1	2	1
1	1	1
1	4	1
2	2	1
3	5	1
3	6	1
4	1	2
4	4	1
4	2	1
5	1	1
5	4	1

Order Table

*OrderID	CustomerID	OrderTime	PickupTime	TotalPrice	PaymentMethod
1	A119099070	05/29 12:15	05/29 13:00	230	Cash
2	D266276735	05/29 08:00	05/29 09:22	30	Credit card
3	Z276769530	05/30 09:50	05/30 10:10	125	Credit card
4	M203328053	05/30 09:58	05/30 13:20	180	Cash
5	Y199892270	06/01 08:55	06/01 10:25	120	Credit card

10. Draw the new class diagram based on your suggested changes. Describe how you would denormalize the model that you created in question.

According to our 3NF result. We need to add the attribute often be used. So the CustomerPhone being add into Order. This might tune the database up.



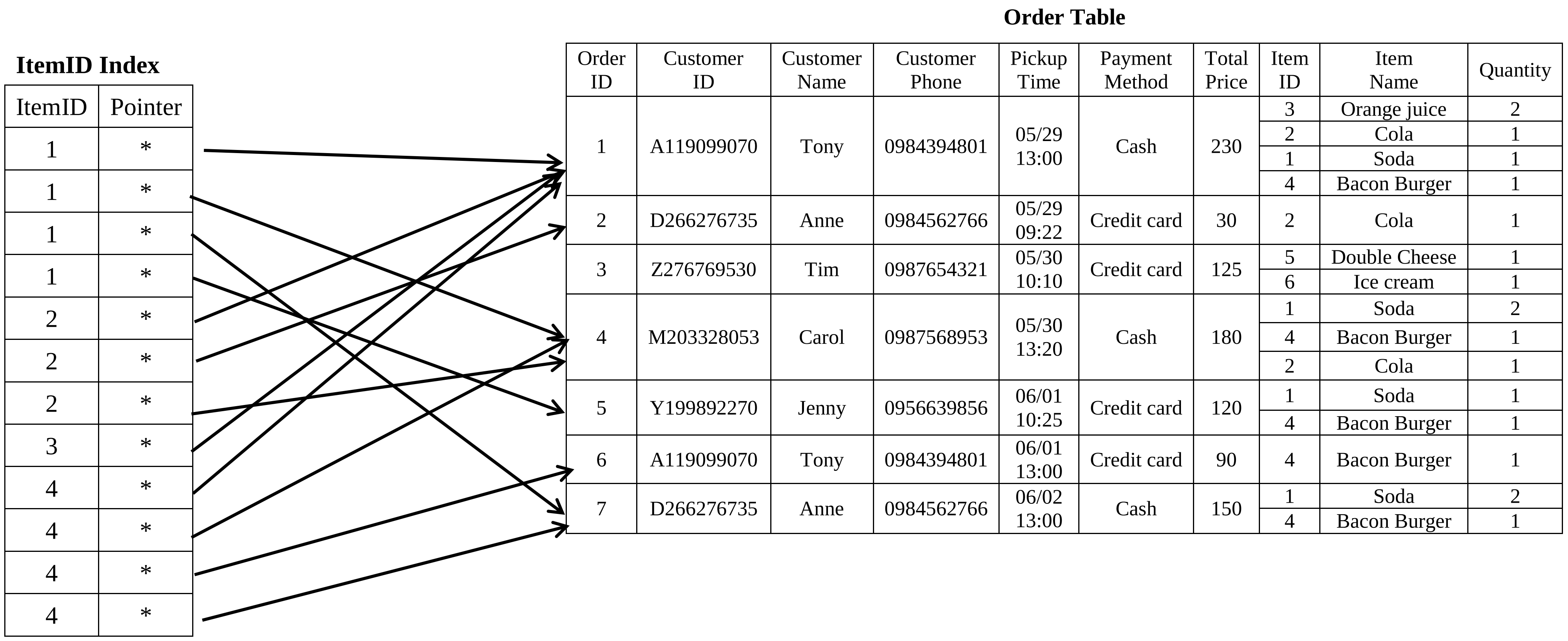
11.Examine the model that you created in question 10. Develop the inter-file clustering and index strategies. Describe how your clustering strategy will improve the performance of the database. List possible indices you would recommend and describe the reasons.

Interfile clustering combine multiple data tables which is often be used. So we add the information of customer into Order table. Furthermore, item information also be used usually, as a result, we integrated the Item table in order to enhance the performance of data access.

Order Table

Order ID	Customer ID	Customer Name	Customer Phone	Pickup Time	Payment Method	Total Price	Item ID	Item Name	Quantity
1	A119099070	Tony	0984394801	05/29 13:00	Cash	230	3	Orange juice	2
							2	Cola	1
							1	Soda	1
							4	Bacon Burger	1
2	D266276735	Anne	0984562766	05/29 09:22	Credit card	30	2	Cola	1
3	Z276769530	Tim	0987654321	05/30 10:10	Credit card	125	5	Double Cheese	1
							6	Ice cream	1
4	M203328053	Carol	0987568953	05/30 13:20	Cash	180	1	Soda	2
							4	Bacon Burger	1
							2	Cola	1
5	Y199892270	Jenny	0956639856	06/01 10:25	Credit card	120	1	Soda	1
							4	Bacon Burger	1

Indices



Order Table

Order ID	Customer ID	Customer Name	Customer Phone	Pickup Time	Payment Method	Total Price	Item ID	Item Name	Quantity
1	A119099070	Tony	0984394801	05/29 13:00	Cash	230	3	Orange juice	2
							2	Cola	1
							1	Soda	1
							4	Bacon Burger	1
2	D266276735	Anne	0984562766	05/29 09:22	Credit card	30	2	Cola	1
3	Z276769530	Tim	0987654321	05/30 10:10	Credit card	125	5	Double Cheese	1
							6	Ice cream	1
4	M203328053	Carol	0987568953	05/30 13:20	Cash	180	1	Soda	2
							4	Bacon Burger	1
							2	Cola	1
5	Y199892270	Jenny	0956639856	06/01 10:25	Credit card	120	1	Soda	1
							4	Bacon Burger	1
6	A119099070	Tony	0984394801	06/01 13:00	Credit card	90	4	Bacon Burger	1
7	D266276735	Anne	0984562766	06/02 13:00	Cash	150	1	Soda	2
							4	Bacon Burger	1

ItemID Index

ItemID	Pointer
1	*
1	*
1	*
1	*
2	*
2	*
2	*
3	*
4	*
4	*
4	*
4	*

For demonstration reason, we added two more example data to make the importance of index more prominent.

The above is the index of ItemID, the column of index is quite complicated as the picture shows. It might lead to error easily. Below is our improvement.

CustomerID Index

CustomerID	Pointer
A119099070	*
A119099070	*
D266276735	*
D266276735	*
Z276769530	*
M203328053	*
Y199892270	*

Order Table

Order ID	Customer ID	Customer Name	Customer Phone	Pickup Time	Payment Method	Total Price	Item ID	Item Name	Quantity
1	A119099070	Tony	0984394801	05/29 13:00	Cash	230	3	Orange juice	2
							2	Cola	1
							1	Soda	1
							4	Bacon Burger	1
2	D266276735	Anne	0984562766	05/29 09:22	Credit card	30	2	Cola	1
3	Z276769530	Tim	0987654321	05/30 10:10	Credit card	125	5	Double Cheese	1
							6	Ice cream	1
4	M203328053	Carol	0987568953	05/30 13:20	Cash	180	1	Soda	2
							4	Bacon Burger	1
							2	Cola	1
5	Y199892270	Jenny	0956639856	06/01 10:25	Credit card	120	1	Soda	1
							4	Bacon Burger	1
6	A119099070	Tony	0984394801	06/01 13:00	Credit card	90	4	Bacon Burger	1
7	D266276735	Anne	0984562766	06/02 13:00	Cash	150	1	Soda	2
							4	Bacon Burger	1

Above is the index of CustomerID, compare to the first picture, it is more simple and less relation. It will have better performance when having greater data.

12.Participate

Score chart

ID	**NAME**	**SCORE**	**Description**
B10523001	Carol	100%	Cohesion, Coupling, Connascence, CRC Card, Class Method
B10523018	Jenny	100%	Cohesion, Coupling, Connascence, Class Method
B10423028	Tony	100%	CRC Card, Mapping, Interfile cluster, Denormalization
B10523019	Jason	100%	Java Code, LOD, Class Diagram, Normalization
B10423036	Anne	100%	Cohesion, Coupling, Connascence, CRC Card
B10323037	Lulu	100%	Java Code, Interfile cluster, Normalization, Keynote
B10523023	Ken	100%	Cohesion, Coupling, Connascence, CRC Card
B10523039	Jess	100%	Normalization, Denormalization, Keynote
B10523051	Grace	100%	Cohesion, Coupling, Connascence, CRC Card