System Analysis & Design
Final Exam

01/03/2018

Name: 任懷蕗 ID: B10423011

1. An Employee has an association with an Account object that tracks all the incomes and charges accrued from transactions. The ~~Customer~~ Employee can call the operations of the Account object, but the Account never invokes operations of the ~~Customer~~ Employee. Since the reference to the Account object does not change over time, we need to do something to prevent callers from accidentally modifying the Account. What is relationship between Employee and Account? 2% Please detect the error of the code and correct it. 5% Please describe the aspects of the Law of Demeter in this situation. 5%

描述 code 裡符合的 LoD.

```
public class Employee {
    private Account acc;
    private String empId;

    public Employee() {
        account=new Account();
    }
    public Account getAccount() {
        return account;
    }
    public String getEmpId() {
        return empId;
    }
    public void setEmpId(String
empId) {
        this.empId = empId;
    }

}
```

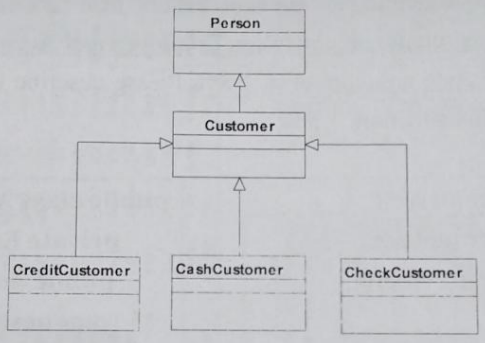
```
public class Account {   追蹤所有交易收入.費用.
    private Employee emp;
    public Account() {
    emp=new Employee();
    }

}
```

2. The direction of an association changes during the development of the system. Assume that we modify the Account class so that the display name of the Account is updated from the name of the Employee. In this situation, the

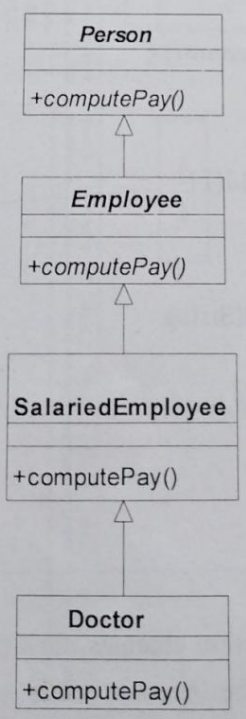
顯示名稱來自 employee
employee 更改 Account也更改 在 Account 新增 Employee

Account needs to access its corresponding Employee object. Therefore, we plan to add an owner attribute to Account. We need to ensure that if a given Account has a reference to a specific Employee, and the Employee has a reference to that same Account. Since neither the Employee class nor the Account class can modify the field anywhere else, this ensures that both reference attributes remain consistent. What is relationship between Employee and Account? 2% Please write the Java code. 8%

3. From a cohesion, coupling, and connascence perspective, is the following class diagram a good model? Why or why not? 5%



4. From inheritance perspective, is the following class a good model? Why or why not? 5%



5. Preconditions and postconditions can be used to specify dependencies among operations in the same class. Please select the right statement for the following situations. **8%**

1) To ensure that we invoke TournamentControl to select sponsor only once. 4

2) To assume that the Player is not yet part of the Tournament of interest. 1

3) To ensure that sponsors cannot be selected before there are interested advertisers. 3

4) To specify how TournamentControl sets the advertisers association when select sponsor. 2

context TournamentControl::IsPlayerOverbooked(p) **pre**: not p.tournaments->includes(self.tournament)	(1)
context TournamentControl:: selectSponsors (advertisers) **post**: tournament.sponsors->sponsors.equals(advertisers)	(2)
context TournamentControl::selectSponsors (advertisers) **pre**: interestedSponsors->notEmpty()	(3)
context TournamentControl::selectSponsors (advertisers) **pre**: tournament.sponsors->isEmpty()	(4)

6. Specify a precondition for renew() of a Membership class. **2%** Draw the activity diagram based on the following renew() algorithm specification. **3%**

1. result ← PaymentProcessor.pay()
2. if result = true, then
3. expirationDate ← today + membershipDuration
4. status ← active

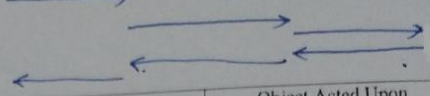
Return the value of result

① 指定 pre-condition
② 畫 activity diagram.

7. The following table is to describe the steps of withdraw money use case. Please draw the sequence diagram. **10%**

Customer ATM GUI withdraw money controller account.

書 sequence diagram.

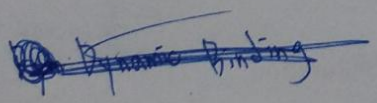


#	Subject	Subject Action	Parameters	Object Acted Upon
3.	Customer	enters	*amount*	ATM GUI
4.1.	ATM GUI	withdraws	*amount*	withdraw money controller.
4.2.	withdraw money controller	verifies	*amount*	account.
4.3.	account	returns	true or false to	withdraw money controller.
4.4.	If true is returned then			
4.4.1.	withdraw money controller	creates	*amount*	message.
4.4.2.	withdraw money controller	dispense	*amount*	dispenser
4.4.3.	withdraw money controller	deducts	*amount*	account.
4.4.5.	withdraw money controller	saves	*account*	database manager.
4.4.3.	else			
4.4.4.	withdraw money controller	creates	*"funds are insufficient to fulfill request."*	message
4.5.	withdraw money controller	returns	*message*	ATM GUI.
4.6.	ATM GUI	displays	*message*	customer.

8. There are six types of interaction cohesion including functional, sequential, communicational, procedural, temporal or classical, logical, and coincidental. Please state the type of cohesion for the following situations.
   1) A system initialization routine: this routine contains all of the code for initializing all of the parts of the system. Lots of different activities occur, all at the initial time. temporal 2%
   2) An object "calculate totals" may keep a running total of the quantity times price subtotal for each item. functional 2%

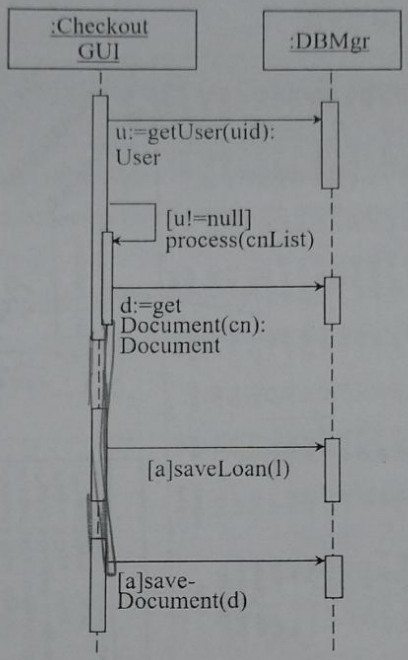
9. Please use an example to specify a method's algorithm for a compute pay method associated with an hourly employee class using an activity diagram. The procedure should include recognition of employee status, the check for hourly employment, calculate the number of hours worked, calculate tax, and the printing of check. 5%

10. Please use an example to illustrate each notation 4%. 畫出先理對應符号

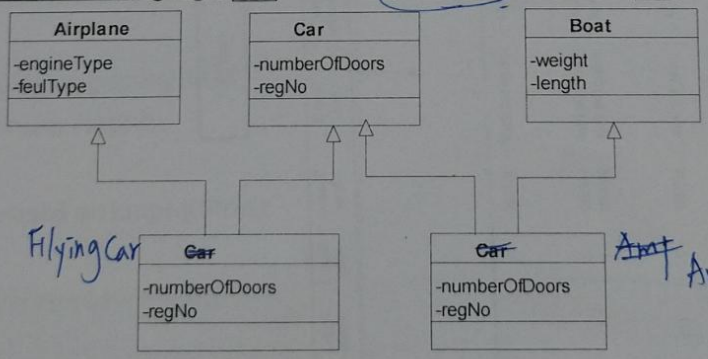
Notation	Meaning
	A named instance without a type, the type is not important, unknown, or to be determined at run time
	An unnamed instance with a type, the name is not important, or not used elsewhere in the sequence diagram

11. Please indicate the commonly seen mistake for the following sequence diagram. 5%

找錯 錯何錯 修正



12. Please use rule 1a and 1b factor out multiple inheritance effect for a single-inheritance language, 6% and an object-based language. 6%



- Product Price = double
- Produce Desc. = String

13. ~~12~~. Given the following Order figure displaying redundant data and null cells in the file. Please use normalization rule to create first normal form, second normal form, and third normal form for it. Please also **add necessary attributes** to the normalized form. You need to identify primary and foreign keys and explain what referential integrity is. 15%

標主鍵. ~~外~~ 外來.
參照完整性.
1 6.
2 4 5

Order
-Order Number : unsigned long
-Date : Date
-Cust ID : unsigned long
-Last Name : String
-First Name : String
-State : String
-Tax Rate : float
-Product 1 Number : unsigned long
-Product 1 Desc. : String
-Product 1 Price : double
-Product 1 Qty. : unsigned long
-Product 2 Number : unsigned long
-Product 2 Desc. : String
-Product 2 Price : double
-Product 2 Qty. : unsigned long
-Product 3 Number : unsigned long
-Product 3 Desc. : String
-Product 3 Price : double
-Product 3 Qty. : unsigned long

Order
Customer
Product
Product Order
State

Redundant Data

Null Cells

Sample Records:

Order Number	Date	Cust ID	Last Name	First Name	State	Tax Rate	Prod. 1 Number	Prod. 1 Desc.	Prod. 1 Price	Prod. 1 Qty.	Prod. 2 Number	Prod. 2 Desc.	Prod. 2 Price	Prod. 2 Qty.	Prod. 3 Number	Prod. 3 Desc.	Prod. 3 Price	Prod. 3 Qty.
239	11/23/00	1035	Black	John	MD	0.05	555	Cheese Tray	$45.00	2								
260	11/24/00	1035	Black	John	MD	0.05	444	Wine Gift Pack	$60.00	1								
273	11/27/00	1035	Black	John	MD	0.05	222	Bottle Opener	$12.00	1								
241	11/23/00	1123	Williams	Mary	CA	0.08	444	Wine Gift Pack	$60.00	2								
262	11/24/00	1123	Williams	Mary	CA	0.08	222	Bottle Opener	$12.00	2								
287	11/27/00	1123	Williams	Mary	CA	0.08	555	Cheese Tray	$45.00	3								
290	11/30/00	1123	Williams	Mary	CA	0.08	555	Cheese Tray	$45.00	2								
234	11/23/00	2242	DeBerry	Ann	DC	0.065	555	Cheese Tray	$45.00	1								
237	11/23/00	2242	DeBerry	Ann	DC	0.065	111	Wine Guide	$15.00	1								
238	11/23/00	2242	DeBerry	Ann	DC	0.065	444	Wine Gift Pack	$60.00	1	444	Wine Gift Pack	$60.00	1				
245	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1								
250	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1								
252	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1	444	Wine Gift Pack	$60.00	2				
253	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	$12.00	1	444	Wine Gift Pack	$60.00	1				
297	11/30/00	2242	DeBerry	Ann	DC	0.065	333	Jams & Jellies	$20.00	2								
243	11/24/00	4254	Bailey	Ryan	MD	0.05	555	Cheese Tray	$45.00	2								
246	11/24/00	4254	Bailey	Ryan	MD	0.05	333	Jams & Jellies	$20.00	3								
248	11/24/00	4254	Bailey	Ryan	MD	0.05	222	Bottle Opener	$12.00	1	333	Jams & Jellies	$20.00	2	111	Wine Guide	$15.00	1
235	11/23/00	9500	Chin	April	KS	0.05	222	Bottle Opener	$12.00	1								
242	11/23/00	9500	Chin	April	KS	0.05	333	Jams & Jellies	$20.00	3								
244	11/24/00	9500	Chin	April	KS	0.05	222	Bottle Opener	$12.00	2								
251	11/24/00	9500	Chin	April	KS	0.05	111	Wine Guide	$15.00	2								

E

國立雲林科技大學考試答案卷

№ 0570722

學年		學期		日期		考別	□平時考 □期中考 □學期考
科目	SAD					評分	64
系所	四資管三A	年級	三	學號	B10423035	姓名	黃均珮 Celia

+3 1. (1) composition 1 to 1 +1

(2) class Account doesn't new Employee because of it never invokes operations of the Employee. In Account , it only has the methods that tracks all the incomes and charges accrued from transactions and attributes that it needs.
Then Employee calling Account's method. +3

Code?

(3) In Employee class.

```
private Account acc;
public Employee(){
    account = new Account();
}
```

+3 to objects that is contained in attributes of itself or a superclass.

LoD 2 to an object that is created by the method.

+5 2. (1) association 1 to 1 +1

(2)

```
public class Employee {
    private Account acc;
    private String empId;
    public Employee() {
        account = new Account(); }
    public Account getAccount(){
        return account;}
    public String getEmpId(){
        return empId; }
    public void setEmpId(String empId){
        this.empId = empId; }
}
```

this

+4

```
public class Account {
    private Employee emp;
    public Account() {
        emp = new Employee(); }
}
```

3. (1) No.

(2) CreditCustomer , CashCustomer and CheckCustomer , they use different payment , so their method are different . It occurs LSP. It makes inheritance conflicts.

So it is not a good model.

4. (1) No

(2) They occurs sematic error easily.

If the top layer (Person) changes something , its subclasses will change.

It is not easy to maintain.

5. (1) 4

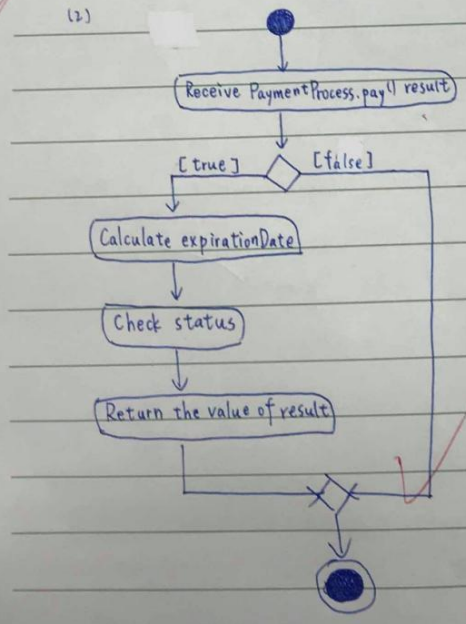
(2) 1

(3) 3

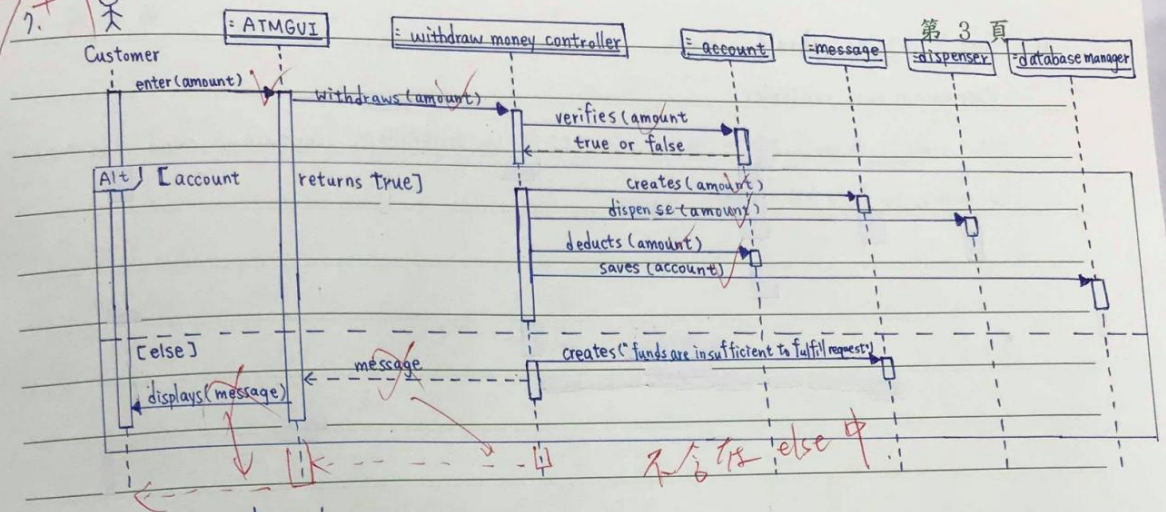
(4) 2

6. (1) result = true

(2)



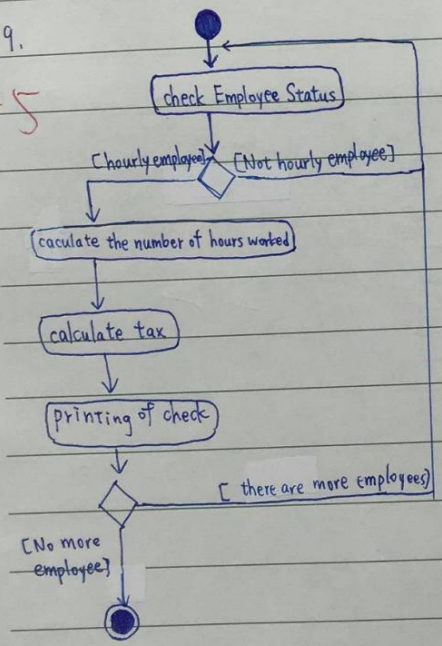
+7+ 7.



+6 8. (1) temporal or classical

(2) functional

9.



+5

10. (1) dynamic linking

+3

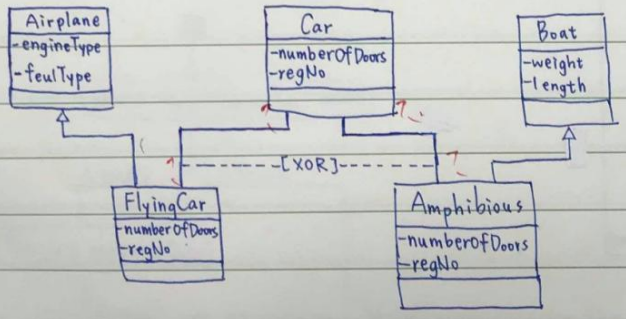
(2) X

11. GUI's lifeline doesn't break.

+5 It doesn't have controller.

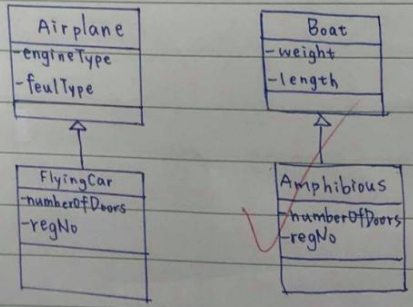
GUI doesn't do method, it just send input to controller. Controller send the messages to models to do methods.

+5 12. 1a.



+2

1b.



+3

1.

Code:沒照以下修正者 扣分

```
public class Employee {
private Account account;
 private String empId;

 public Employee() {
      account=new Account();
 }
 public Account getAccount() {
      return account;
 }
 public String getEmpId() {
      return empId;
 }
 public void setEmpId(String empId) {
      this.empId = empId;
 }

}
```

```
package uni.one2one.association;
ACCOUNT 不應該含有EMPLOYEE
public class Account {

 public Account() {
 }

}
```

Employee 與 Account 關係為 1 對 1 Composition，只寫 Composition 者給 1 分

符合 Law of Demeter 第二條規則，Employee 可使用 Account 的方法。

老師認為 LoD1 在 set method 也會用到

只提到其中任何一條只給 3 分

兩條提到滿分

參照就扣扣到沒分

2.

```
public class Employee {
 /* The account field is initialized
  * in the constructor and never modified
  */
    private Account account;
    private String empId;
沒this等於 無法建立 employee 與account間的關聯 扣一半分數
    public Employee() {
        account=new Account( this );
    }
    public Account getAccount() {
        return account;
    }
    public String getEmpId() {
        return empId;
    }
    public void setEmpId(String empId) {
        this.empId = empId;
    }

}
```

```
public class Account {
 /* The owner field is initialized
  * during the constructor and
  * never modified.        */
 private Employee owner;

 public Account(Employee owner) {
     this.owner=owner;
 }

 public Employee getOwner() {
     return owner;
 }
}
```

Employee 與 Account 關係為 1to1 association 因為 Employee 中有 Account，Acc
Employee
只寫 association 給 1 分

3.

From a cohesion point of view the diagram is excellent because all data pertaining to each elementary class can be collected together while common attributes are contained in higher level classes.

From a coupling point of view, there is more communication needed among classes as lower level classes inherit data as part of their usage.

From a connascence perspective, if specific type of customers need changes in their standard attributes (e.g. if banks require a different ID number than the firm), changes will have to be made at multiple levels. However, this is probably an event that will be infrequently experienced.

4.

兩同時提及 繼承衝突、高耦合、語意重新定義才得滿分，只解釋一半只給 3 分

只提關鍵名詞沒解釋給 1~2 分

this is not good model

because will have many redefinition and inheritance conflict problems.

Doctor is a subclass of employee.

Both have methods named computePay() this causes an inheritance conflict.

Furthermore, when the definition of a superclass is modified, all its subclasses are affected.

5.

1) To ensure that we invoke TournamentControl to select sponsor only once. __4___
2) To assume that the Player is not yet part of the Tournament of interest. __1___
3) To ensure that sponsors cannot be selected before there are interested advertisers. _3____
4) To specify how TournamentControl sets the advertisers association when select sponsor. __2____

6.

架構錯誤 全錯!!!!!!!!!!!!!!!!!!!!!!!!!!!!

不完整 視為 active 全錯

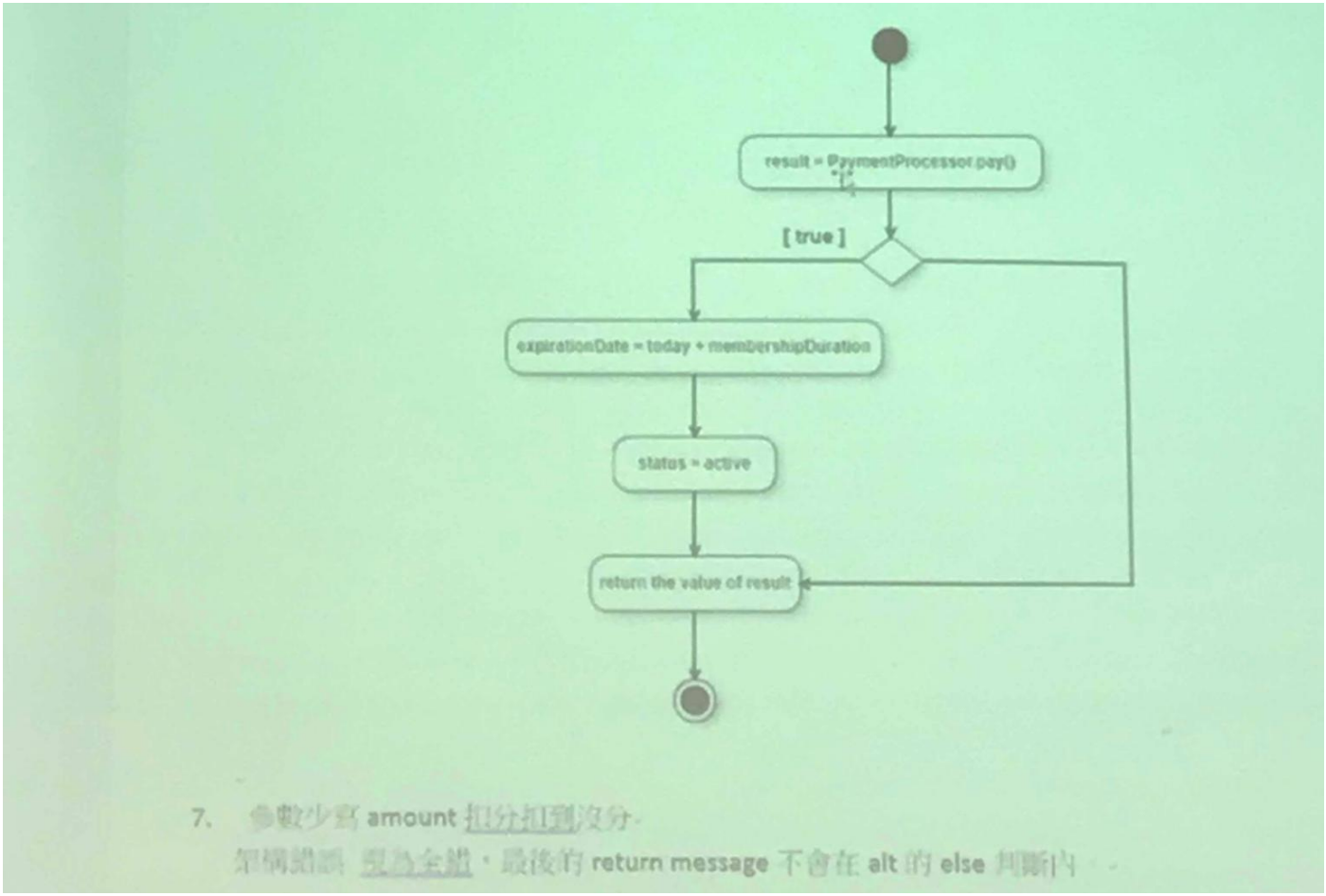
錯字 扣一分扣到沒分

前置條件:

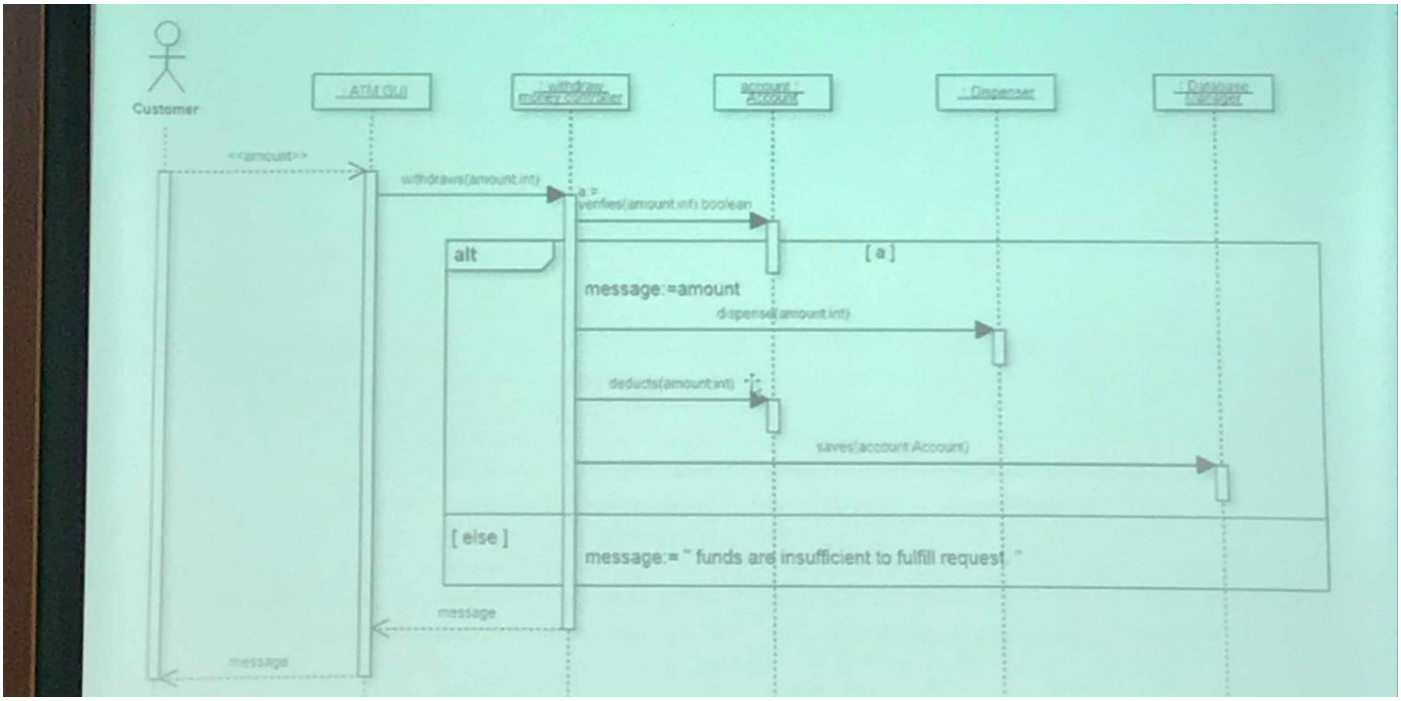
PaymentProcessor.pay() = true

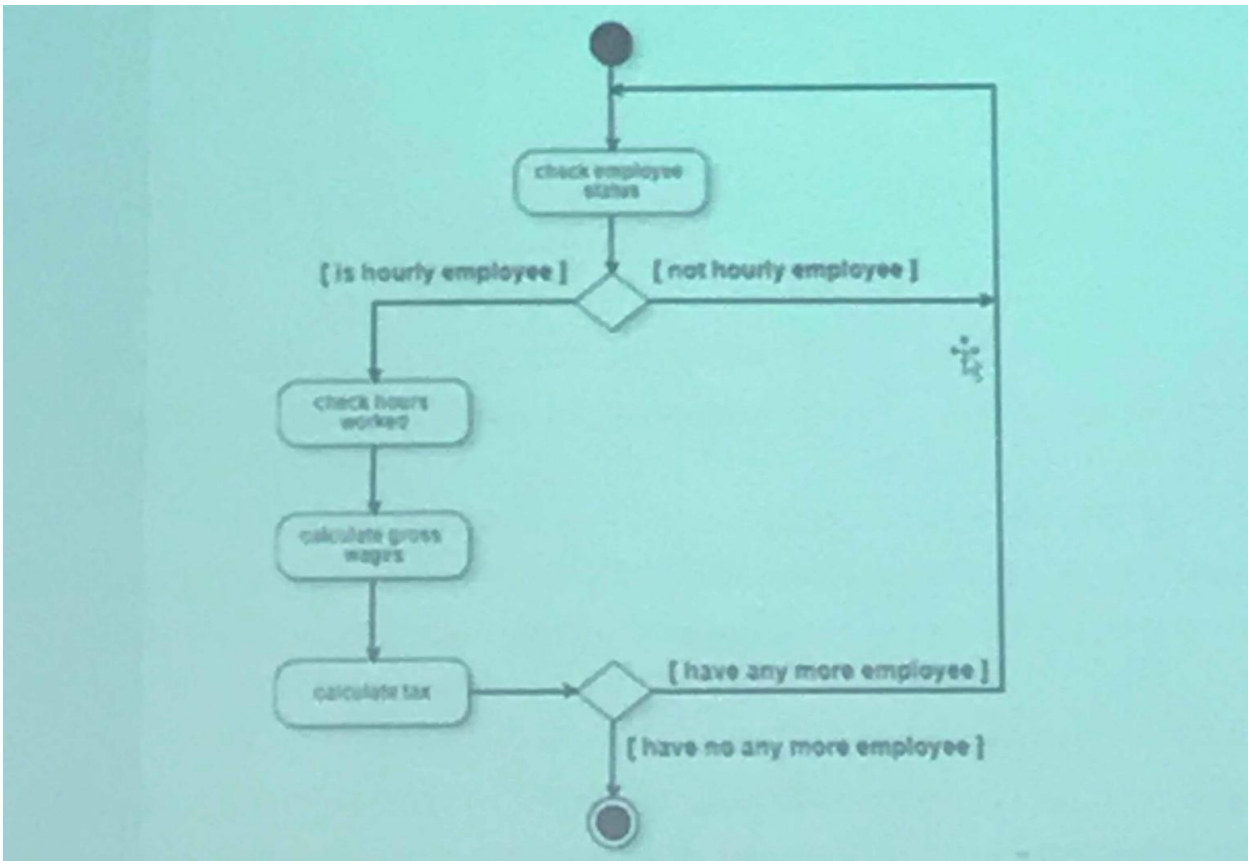
不符以上者錯

只寫 result = true 錯，沒解釋 result 是何物



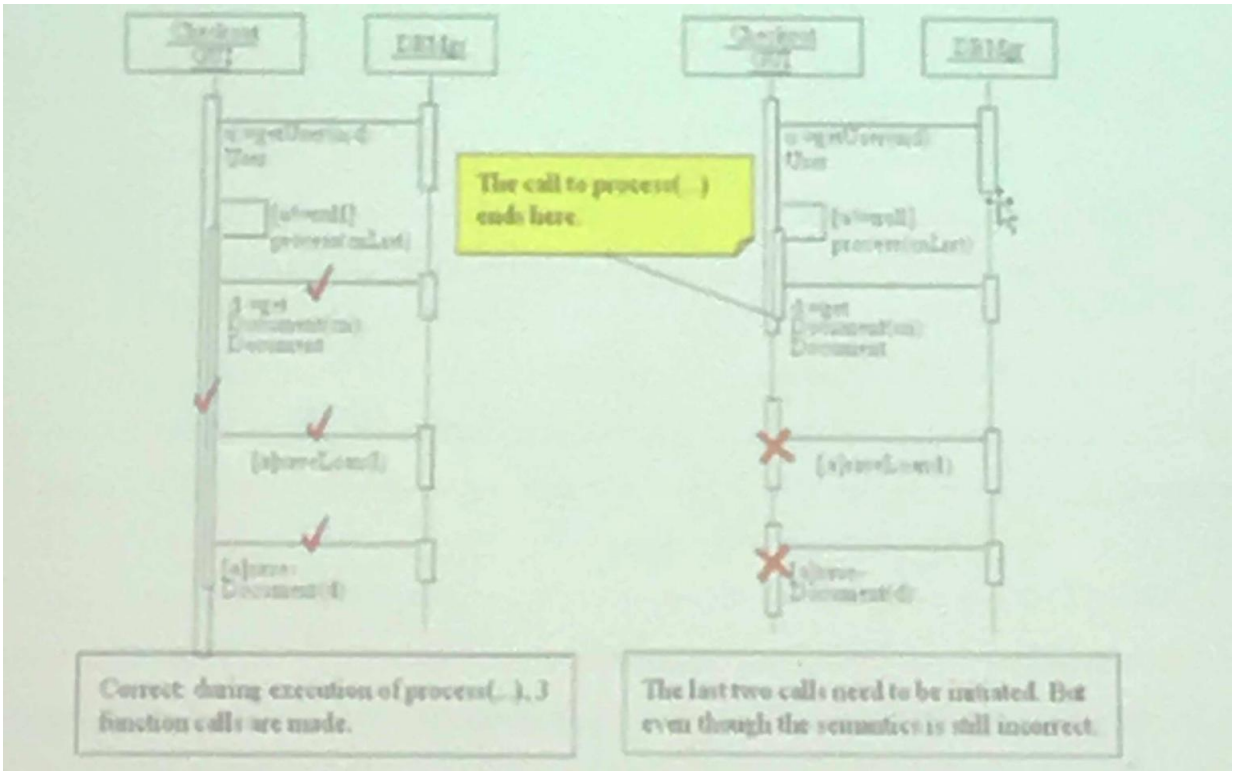
7. 參數少套 amount 扣分扣到沒分.
架構錯誤 視為全錯，最後的 return message 不會在 alt 的 else 判斷內



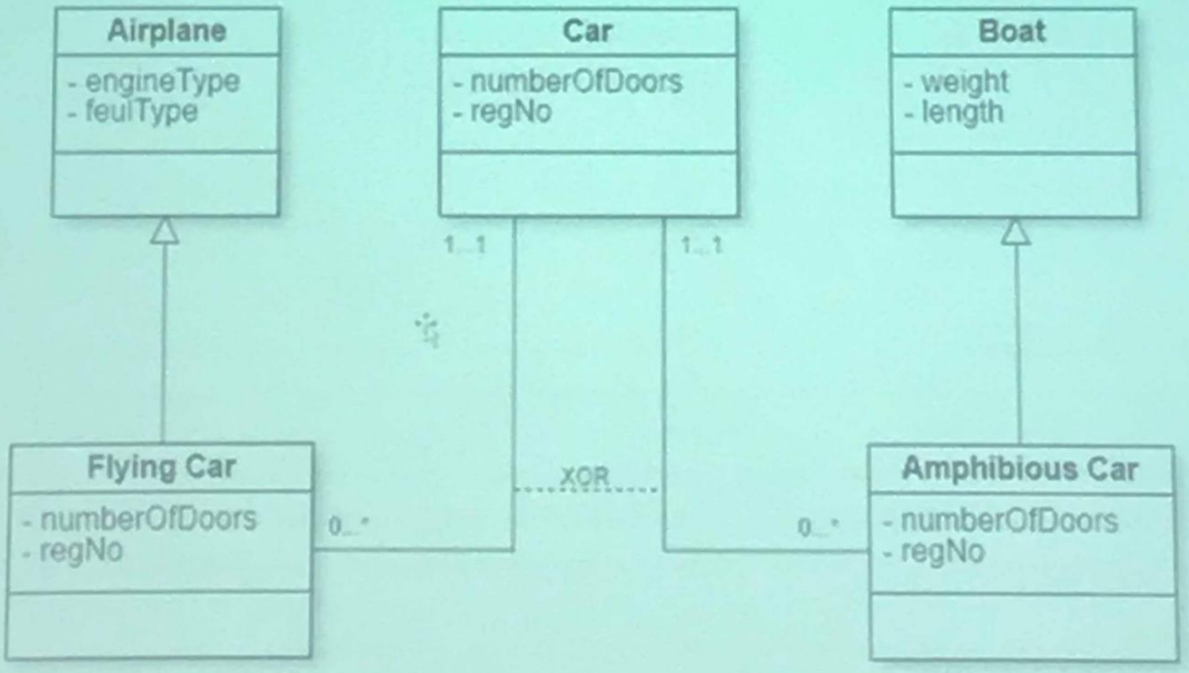


10.

Notation	Meaning
myCar:	A named instance without a t type is not important, unknown.



Single inheritance language
1a



Single inheritance language
1b