EXAM 12 August 2016, answers

Software Design (City University of Hong Kong)

CITY UNIVERSITY OF HONG KONG

Text

Course code & title : CS3342 Software Design

Session : Semester B 2015/16

Time allowed : Two hours

SOLUTION

This paper has **14** pages (including this cover page).

1. This paper consists of **5** questions.
2. Answer **ALL** questions **within** the examination booklet.

*This is a **closed-book** examination.*

No materials or aids are allowed during the whole examination. If any unauthorized materials or aids are found on a candidate during the examination, the candidate will be subject to disciplinary action.

Student Number: _____

Seat Number: _____

Answer all questions						
	CILO 1	*CILO 2*	*CILO 3,4*	*CILO 3,4*	*CILO 5*	
Question	**1**	**2**	**3**	**4**	**5**	**Total**
Max	**10%**	**20%**	**30%**	**35%**	**5%**	**100%**

Question 1 – Software Development Process (10 Marks):

1a). What are the advantages and disadvantages of **Waterfall Model**? (4 Marks)

Advantages of waterfall model: (x2 points)
This model is simple and easy to understand and use.
It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
In this model phases are processed and completed one at a time. Phases do not overlap.
Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model: (x2 points)
Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
No working software is produced until late during the life cycle.
High amounts of risk and uncertainty.
Not a good model for complex and object-oriented projects.
Poor model for long and ongoing projects.
Not suitable for the projects where requirements are at a moderate to high risk of changing.

1b). What are the advantages and disadvantages of **Software Prototyping**? (4 Marks)

Advantages (x2 points)
- reduced time and costs
- improved and increased user involvements and feedbacks

Disadvantages (x2 points)
- insufficient analysis
- user confusion of prototype and finished product
- developer misunderstanding of user objectives
- excessive development time of the prototype

1c). Explain why is **Software Reuse** important in Software Engineering? (2 Marks)

In most engineering disciplines, systems are designed by composing existing components that have been used in other systems.
Software engineering has been more focused on original development but it is now recognised that to achieve better software, more quickly and at lower cost, we need to adopt a design process that is based on systematic software reuse.

Increased dependability
Reduced process risk
Effective use of specialists
Standards compliance and accelerated evelopment

Question 2 – Software Requirements – CILO 2 (20 Marks):

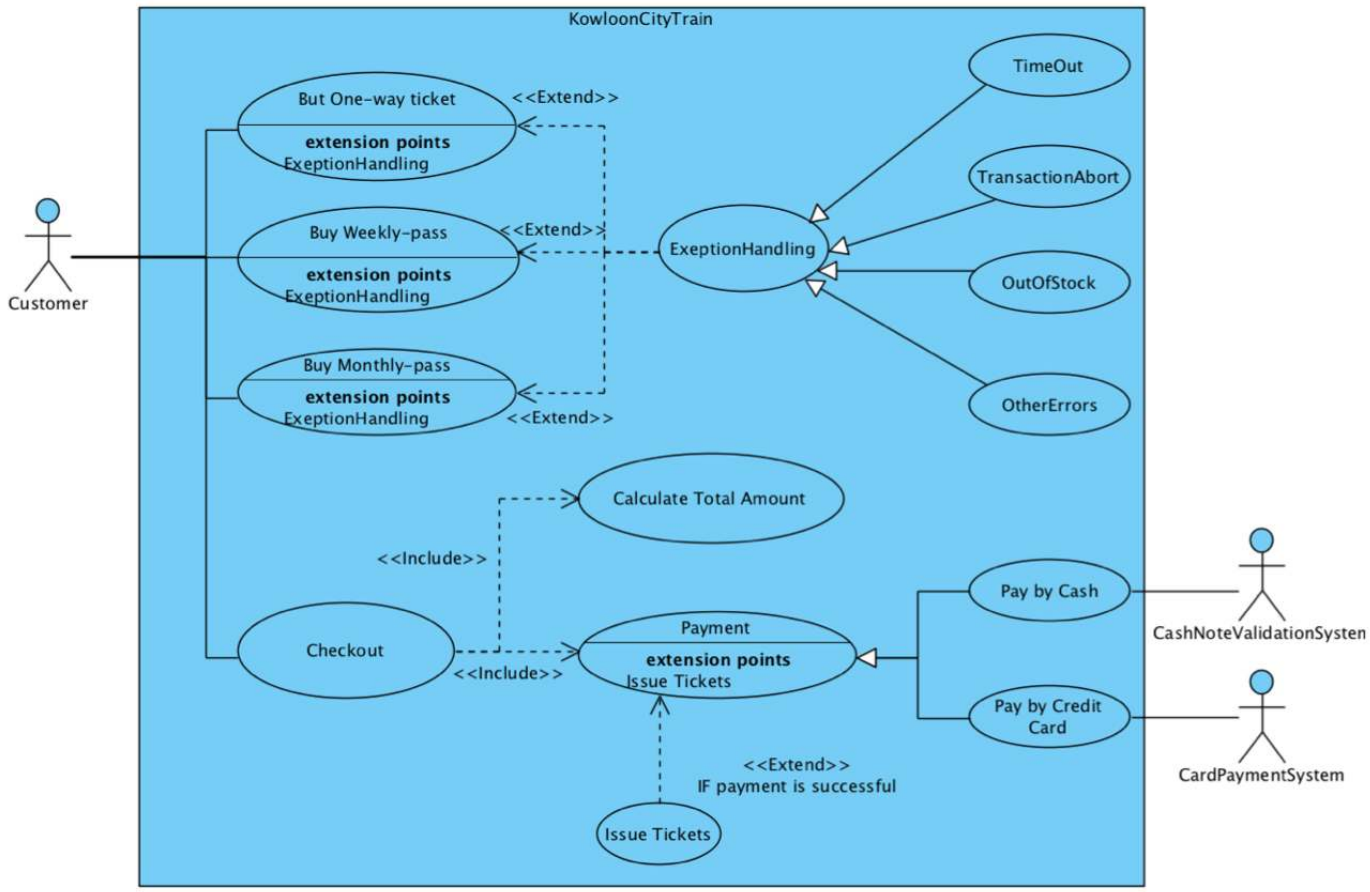
2a). Study the following scenario. Draw a complete **use case diagram** of a *Ticket Vending Machine* system for **HongKong CityTrain**.

A **Customer** arrives at the train station ticket vending machine:

1. She has 3 options/use-cases allow her to: **buy One-way ticket**, **buy Weekly-pass** or **buy Monthly-pass**.
2. In each of these three options, IF any error occurs, THEN the system must be able to handle using **ExceptionHandling**, there are following different error exceptions (*hint: use-case inheritance*)
   a. **TimeOut** (i.e. the customer took too long to complete the transaction)
   b. **TransactionAbort** (i.e. the customer choose to cancel without completing the transaction)
   c. **OutOfStock** (i.e. the vending machine runs out of Tickets or Passes)
   d. **OtherErrors** (i.e. this is to handle any other errors not covered above)
3. She can choose multiple items. After all the selections of above are completed, she may proceed to the **CheckOut** function/use-case, which (1) will **Calculate the total amount**, and then (2) proceed to the **Payment** screen, where she will be given two options: (*hint: use-case inheritance*)
   a. **Pay by Cash**, the inserted cash note will be validated by a **CashNoteValidationSystem or**
   b. **Pay by Credit Card**, the inserted credit card will be processed by a **CardPaymentSystem**
4. Only IF the payment is successfully completed, THEN it will **issue the tickets**.
5. The customer can now continue her journey with ticket(s) purchased.

Whenever possible, your use case diagram MUST use **<<Extend>>** or **<<Include>>** as well as inheritance techniques to provide a good use case diagram. **(10 Marks)**

<Screen Capture: Place Use Case Diagram Here >



2b). Requirements Specifications **(10 Marks)**

Based on the same case study described above, complete the following table to describe the **Checkout** use case under typical course of events (assuming Customer pay by Cash) AND alternative course of events (assuming customer pay by Credit Card). The situation involves the **Customer** actor, as well as external payment processing systems such as **CardPaymentGateway** and **CashNoteValidator** Systems.

Use Case Name:	**Checkout**	
Actor(s):	Customer	
Description:	This use case describes the process of a customer completing the checkout for the tickets selected. On completion, tickets will be issued.	
Reference ID:	METRO-1.0	
Typical course of events:	**Actor Action**	**System Response**
	Step 1: This use case is initialized when a customer proceeds to checkout and pay for tickets. **Step 3**: The customer choose to pay Cash	**Step 2**: The total purchases will be calculated by invoke the use case *CalculateTotal* **<include: CalculateTotal>,** and displaying the total. **Step 4**: The payment is then processed by invoking the use case *Pay by Cash* <extends the included use case *Payment*>**.**
Alternative course of events:	**Step 3a**: The user chooses to pay by credit card instead of cash. **Step 4a**: The payment is then processed by invoking the use case *Pay by Credit Card* <extends the included use case *Payment*>**.**	
Precondition:	Checkout can only be made after at least one ticket is selected to purchase.	
Postcondition:	The completed transactions will be recorded.	

Question 3 – Object-Oriented Modelling CILO 3 (30 Marks):

3a). The City Library needs to implement a new system to keep track of the borrowings of books by their library members. Each member is able to borrow many books from the Library, and the library also needs to record the due date of each book being borrowed by a member.

For recording information about their members, the Library need to record attributes:

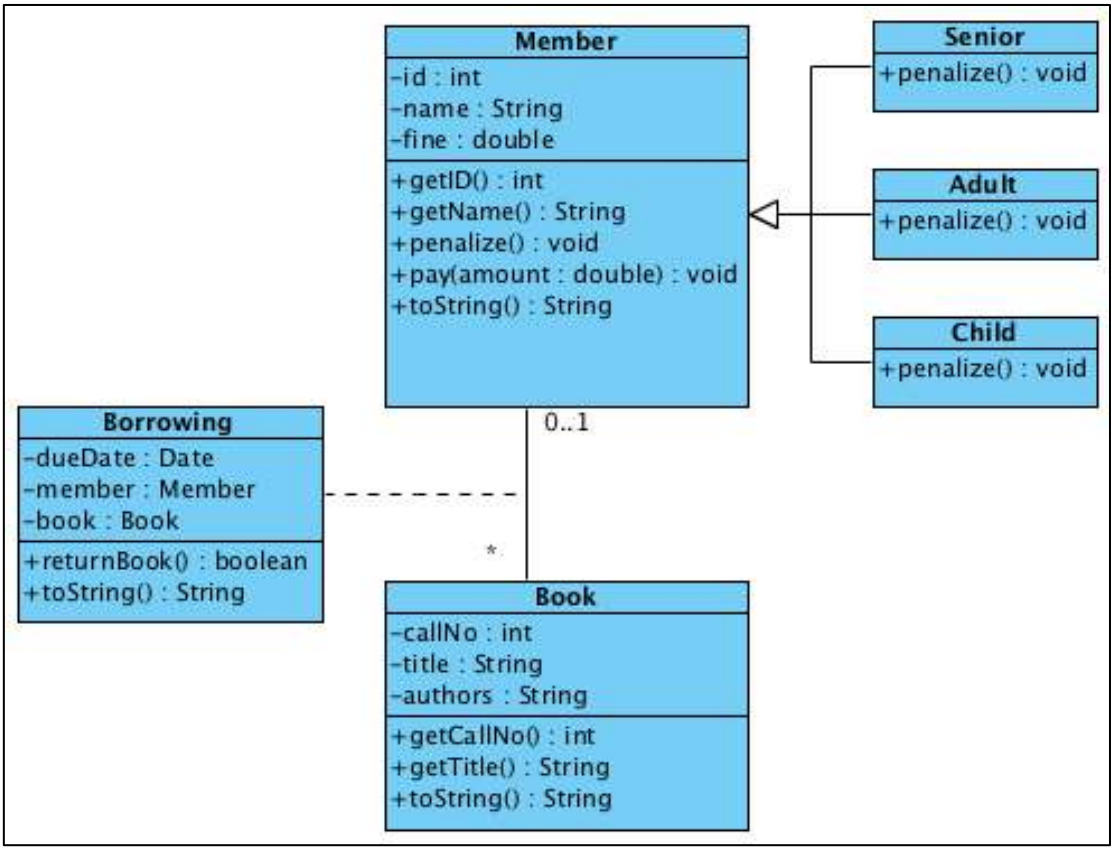
- Member_ID: int
- name: String
- age : int
- As well as operations required accessing these attributes.

For recording information about their books, the Library need to record attributes:

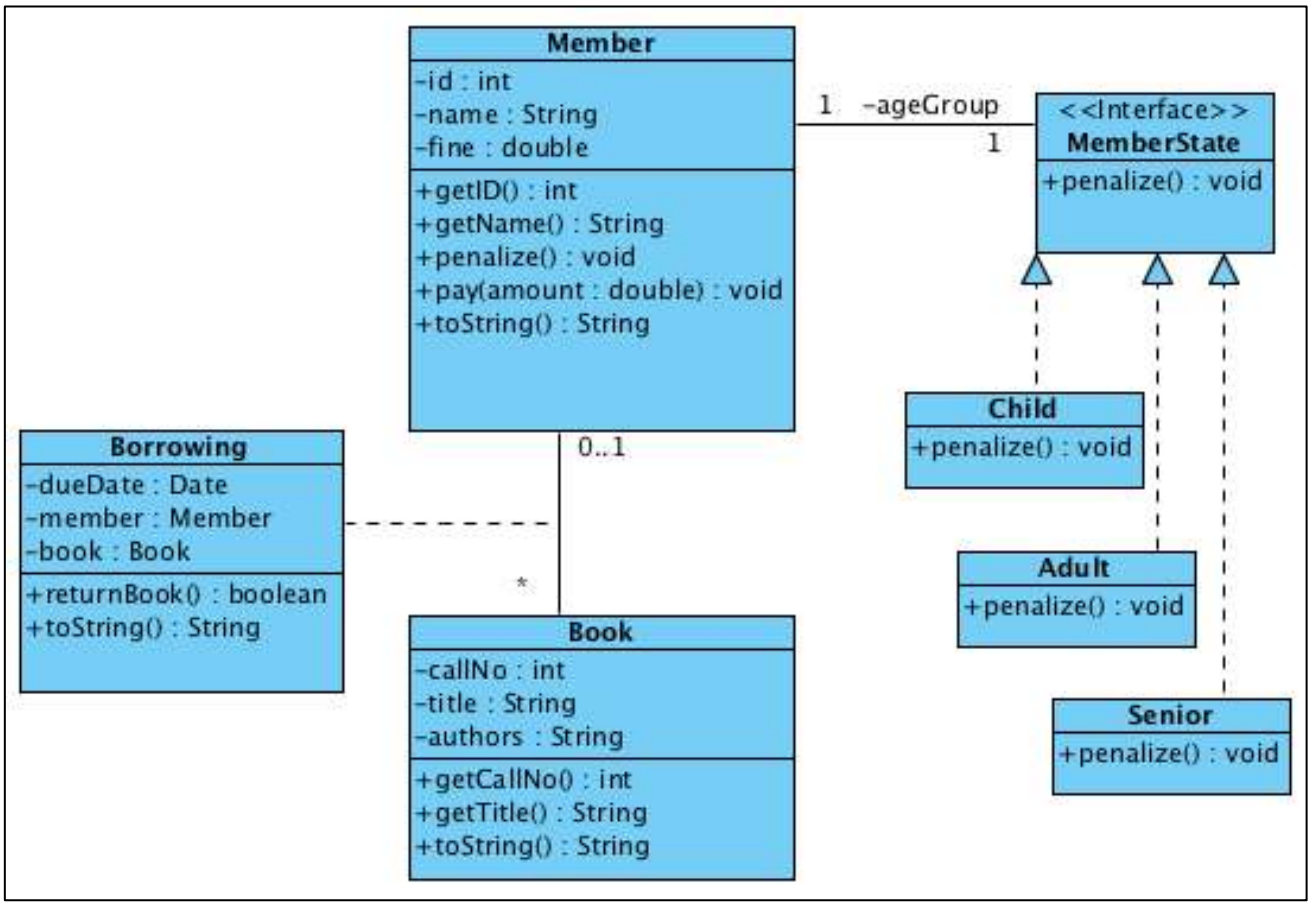
- Book_ID: int
- title: String
- author: String
- As well as operations required accessing these attributes.

The library needs to record information about the book being borrowed by a specific member, as well as its dueDate (book return date). Given that many library members and many books are in the system, how do you correctly model such a relationship using UML class diagram to design the system?

Please show your solution using an UML class diagram, including all attributes and operations required. The class constructor is optional. (10 Marks)



3b). In addition to 3a), the City Library needs to implement three different membership classifications (Student, Adult and Senior) to facilitate the different membership fee charges according to their membership types. i.e. $50/year for Student, $200/year for Adult and $20/year for Senior members, the system should be able to facilitate the changes of membership classifications in a long term. Please provide a solution to extend your class diagram given in 3a).



3c). Please explain and justify in full details whether your solution given in 3b) satisfies the OCP Design Principle? (5 Marks)

If b has State Design Pattern, then it satisfied open for extension and close for modification. Specifically, membership can now be easily changed from Student to Adult, including its membership charges, this satisfies the close for modification purpose. IF the library wants to add a new classification such as a membership class called Child, then it can be easily added to the classification by implementing its interface Membership, this provided necessary structure for OCP.

3d). SOLID Design Principles

What are the first five design principles in object-oriented design commonly referred to the principle of **SOLID**? Please name them. (5 Marks)

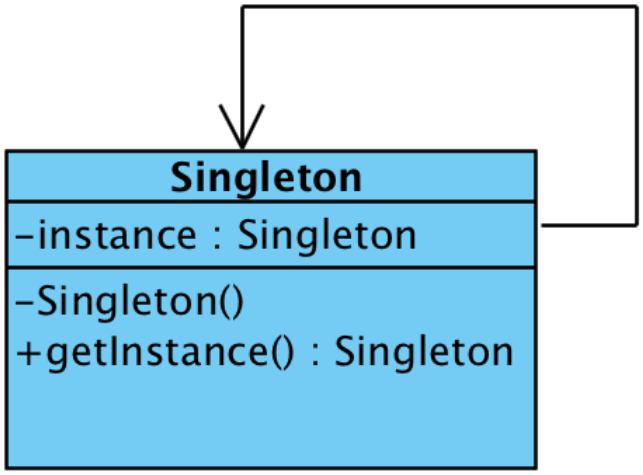
SOLID (**single responsibility, open-closed, Liskov substitution, interface segregation and dependency inversion)**

Question 4 – Design Principles and Design Patterns (35 Marks):

4a). Singleton Pattern (5 Marks)

Please describe the motivation and when to use Singleton Pattern in software design, in addition to your explanation, please draw a simple class diagram to show an example of Singleton Pattern including its essential attributes and operations.

- Motivation/when to use
  - Need to have ONLY one (1) instance for a class.
  - Singleton instantiates itself, and only one(1) of such.
- Intent
  - Ensure that only 1 object instance is ever created.
  - Provide a global point of access.
  - Save the number of duplicated objects being created. Such as in State Pattern.



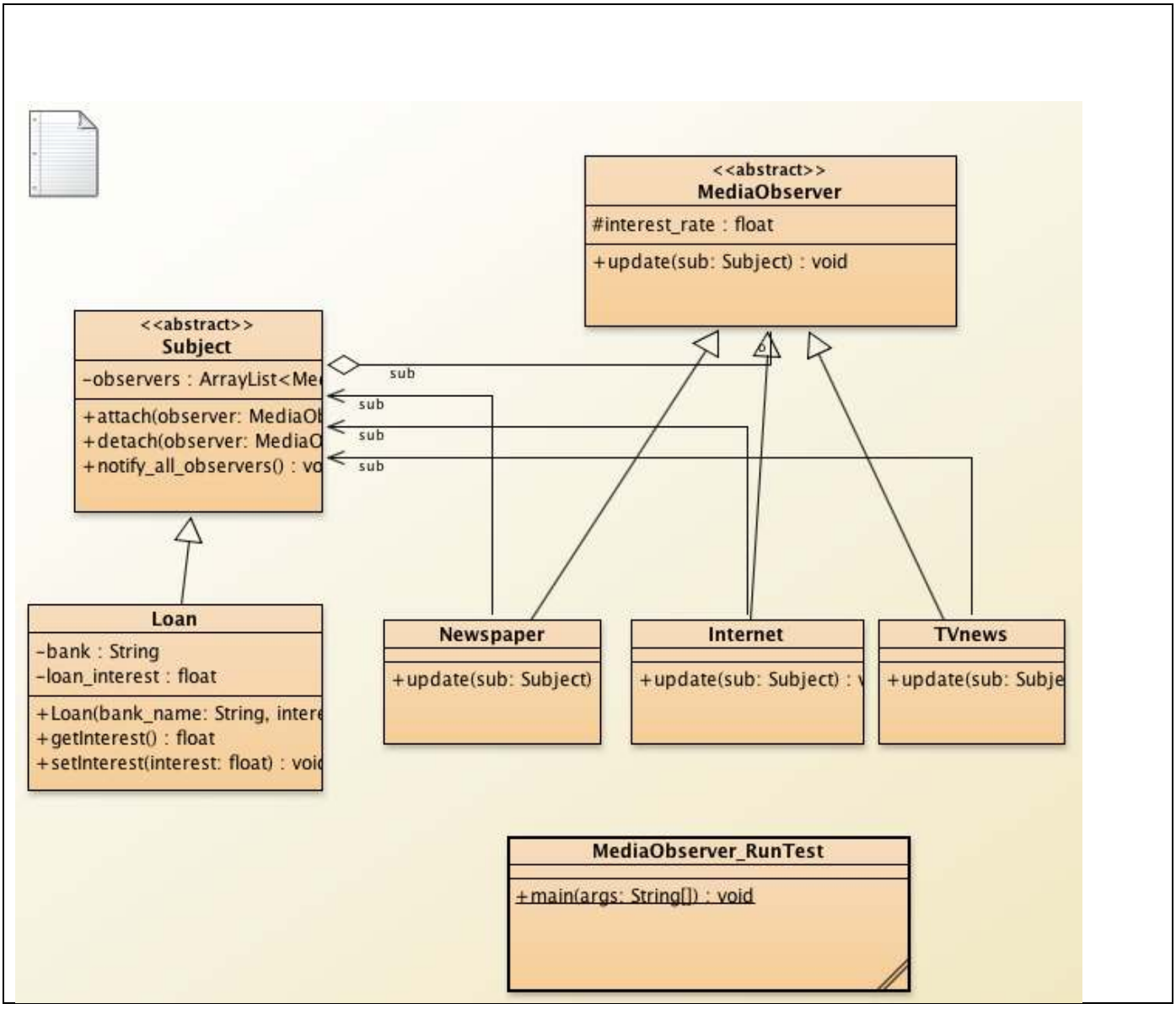
4b). Observer Pattern (10 Marks)

To facilitate the needs of information dissemination of banks, the City's Banking & Monetary Authority needs to design a new system to allow all news agencies to receive the latest updates about different bank's financial updates such as their latest interest rates. You are required to use the **Observer Pattern** to create a design and using a complete **class diagram** to show your solution.

Specifically, the system needs to handle at least three different news agencies namely, **NewsPaper**, **Internet, and TVnews**, and your design should ensure its compliance to OCP design principle.

According to the Observer Pattern, the subject of the information update should include the latest information of its **Loan** interest rate and **TermDeposit** interest rate of banks. The system should be able to register and unregister concerning news agencies to such a subject for information of a bank, as well as being able to notify all news agencies about the latest updates of the concerning subject registered.

Please show your solution using Observer Pattern in a complete UML class diagram. You may consider using ArrayList to store all the registered observers.



4c). Software Design Principle

Study the following code:

IN PHONE.JAVA

```
//The class Phone models many functions of the phone devices
public interface Phone {
   //Make a call to the phone whose number equals the phoneNum.
   public void makeCall(int phoneNum);
   //Send a message to the phone whose number equals the phoneNum.
   public void sendMessage(int phoneNum);
   //Take highquality (more than 8M pixels) images
   public void takeHighQualityImage();
   //Access Internet via Wi-Fi
   public void accessWebviaWifi();
}
```

IN MOTO2100.JAVA

```
//The class Moto2100 models the Moto2100 product
public class Moto2100 implements Phone {
   //Make a call to the phone whose number equals the phoneNum.
   public void makeCall(int phoneNum){
      System.out.println("Calling to " + phoneNum + " with a Moto2100");
   }
   //Send a message to the phone whose number equals the phoneNum.
   public void sendMessage(int phoneNum){
      System.out.println("Sending message to " + phoneNum + " with a Moto2100");
   }
   //Take highquality (more than 8M pixels) images
   public void takeHighQualityImage(){
      System.out.println("This operation is not supported in this device");
   }
   //Access Internet via Wi-Fi
   public void accessWebviaWifi(){
      System.out.println("This operation is not supported in this device");
   }
}
```

IN GALAXY.JAVA

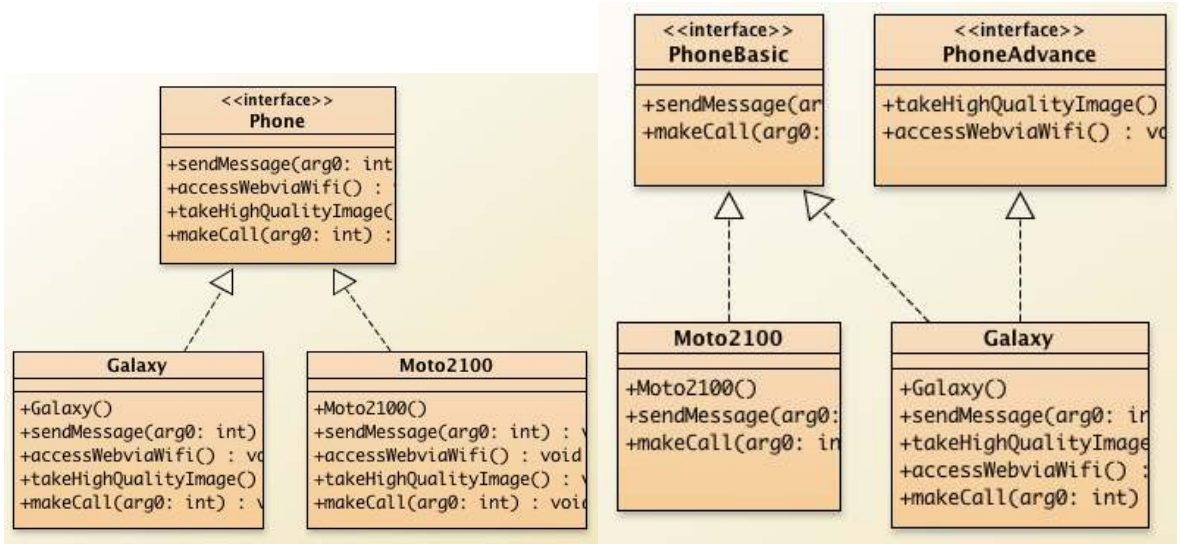
```
//The class Galaxy models the Galaxy product
public class Galaxy imeplements Phone{
   //Make a call to the phone whose number equals the phoneNum.
   public void makeCall(int phoneNum){
      System.out.println("Calling to " + phoneNum + " with a Galaxy");
   }
   //Send a message to the phone whose number equals the phoneNum.
   public void sendMessage(int phoneNum){
      System.out.println("Sending message to " + phoneNum + " with a Galaxy");
   }
   //Take highquality (more than 8M pixels) images
   public void takeHighQualityImage(){
      System.out.println("Take highquality image with the Galaxy technology");
   }
   //Access Internet via Wi-Fi
   public void accessWebviaWifi(){
      System.out.println("Access web via Galaxy Wifi technology");
   }
}
```

The class **Phone** models the general functions of various phone devices. On top of Phone, the classes of **Moto2100** and **Galaxy** implement their specific behavior. Study the java code above and point out the bad design within that violates the ISP design principle. Then you should propose your improvement on the current (undesirable) design. Specifically, your answer should meet the following two requirements:

(1) Point out and explain why these codes represent a violation of the ISP design principle.

The current design enables the class Phone to contain many functions (e.g., *makeCall()*, *takeHighQualityImage()*). However, **some functions are not supported** in some of its subclasses (e.g., *takeHighQualityImage()* and *accessWebviaWifi()* are not supported in subclass Moto2100). Nonetheless, the subclass Moto2100 still has to implement those unsupported function. This means many functions inherited from this base class can be redundant for its subclass. This **design violates** the Interface Segregation Principle (ISP). To enable the design to meet ISP, we need to re-organize the original base class Phone such as producing a new class entitled PhoneBasic to guarantee that it has the minimal useful functions (e.g., *makeCall(int phoneNum)* and *sendMessage(int phoneNum)*) for all possible subclasses (e.g., Moto2100 and Galaxy). Then, for all these functions (e.g., *takeHighQualityImage()* and *accessWebviaWifi()*) that are not included in this newly created base class, we design various classes (e.g., PhoneAdvance) to include them. In this way, a subclass of the original Phone class can flexibly extend the new base class with implementing different interfaces (e.g., Moto2100 only extends the PhoneBasic, Galaxy extends the PhoneBasic and also extends the class PhoneAdvance). The improved code is as follows.

(2) Describe your idea of improvement, and sketch a new class diagram to illustrate your design.



4d). Software Design and Roles of Variables (10 Marks)

Study the following Java codes and identify the role of each variable declared in the code listing by completing the tables below (you may refer to the roles of variables in Appendix I):

```
_____
public class SortPercentage {
    public static void main(String[] args) {
        int intArray[] = new int[]{5,90,35,45,95,3,45,19,62,73};
        double intArrayPercent [] = new double[intArray.length];
        double sum=0, max=0, min=0, range=0;
        double percent=100;

        Sort(intArray);

        for (int i=0 ; i< intArray.length; i++) {
            double percent_conv = (double)intArray[i] / percent;
            intArrayPercent[i] = percent_conv;
            sum = sum + percent_conv;
        }
        max = intArrayPercent[intArrayPercent.length-1];
        min = intArrayPercent[0];
        range = max - min;

        System.out.println("Size of Array is: \t" + intArrayPercent.length);
        System.out.println("Sum is: \t\t"+ sum);
        System.out.println("Max is: \t\t"+ max);
        System.out.println("Max is: \t\t"+ min);
        System.out.println("Range is: \t\t"+ range);
    }
    private static void Sort(int[] intArray) {
        int n = intArray.length;
        int temp = 0;

        for(int i=0; i < n; i++){
            for(int j=1; j < (n-i); j++){
                if(intArray[j-1] > intArray[j]){
                    //swap the elements!
                    temp = intArray[j-1];
                    intArray[j-1] = intArray[j];
                    intArray[j] = temp;
                }
            }
        }
    }
}_____
```

In *Main* Function:

Variable	Role	(6 Marks)
intArrayPercent	Organizer	(1 Mark)
percent	Constant	(1 Mark)
sum	Gatherer	(1 Mark)
range	Transformation	(1 Mark)
percent_conv	Transformation	(1 Mark)
min	Most Wanted	(1 Mark)

In *Sort* Function:

Variable	Role	(4 Marks)
temp	Temporary	(1 Mark)
intArray	Constant	(1 Mark)
n	Constant	(1 Mark)
j	Stepper	(1 Mark)

Question 5 – Software Engineering Professional Ethics (5 Marks):

5a). Why is Software Engineering Code of Ethics important to Software Engineers? (3 Marks)

Codes, if carefully written and properly promoted, can be powerful instruments in the drive for professionalism and in establishing safeguards for society. They do not have to be and should not be sterile, which is often the perception that people have of them. This draft code evolved after extensive study of several computing and engineering codes[1]. All of these codes try to educate and inspire the members of the professional group that adopts the code. Codes also inform the public about the responsibilities that are important to a profession. Codes instruct practitioners about the standards that society expects them to meet, and what their peers strive for and expect of each other. Codes are not meant to encourage litigation, and they are not legislation; but they do offer practical advice about issues that matter to professionals and their clients and they do inform policymakers. These concepts have been adopted in the development of this code. Based on the feedback from readers of this publication and from other sources, a final draft of the code will be developed and presented to the Steering Committee for approval.

5b). What are the x8 ACM Software Engineering Code of Ethics for Professional Practice? Detailed Explanation of each code is not required. (2 Marks)

1. PUBLIC - Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Appendix I

Roles of Variables

Role	Description
Constant/ Fixed value	A variable which is initialized without any calculation and whose value does not change thereafter.
Stepper	A variable stepping through values that can be predicted as soon as the succession starts.
Most-recent holder	A variable holding the latest value encountered in going through a succession of values.
Most-wanted holder	A variable holding the “best” value encountered so far in going through a succession of values. There are no restrictions on how to measure the goodness of a value.
Gatherer	A variable accumulating the effect of individual values in going through a succession of values.
Transformation	A variable that always gets its new value from the same calculation from value(s) of other variable(s).
Follower	A variable that gets its values by following another variable.
One-way flag	A two-valued variable that cannot get its initial value once its value has been changed.
Temporary	A variable holding some value for a very short time only.
Organizer	A data structure, which is only used for rearranging its data and object elements after initialization.

-- END --