EXAM 13 July 2016, questions and answers

Software Design (City University of Hong Kong)

CITY UNIVERSITY OF HONG KONG

Course code & title	:	CS3342 Software Design
Session	:	Semester B 2015/16
Time allowed	:	Two hours

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 the **Waterfall Model**? (4 Marks)

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

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

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)**

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, CashNoteValidatorSystem, CardPaymentGatewaySystem**	
Description:	This use case describes the process of a customer completing the checkout procedure for the tickets selected. On completion, tickets will be issued if the payment is successfully processed.	
Reference ID:	HKG-CITY-TRAIN-TICKET-1.0	
Typical course of events:	**Actor Action**	**System Response**
Alternative course of events:		
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), you can provide a full or only the extended class diagram to show your solution. (10 Marks)

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

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)

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.

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. (5 Marks)

(2) Describe your idea of improvement, and sketch a new full class diagram to illustrate your design that complies with the ISP design principle. (5 Marks)

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		(1 Mark)
percent		(1 Mark)
sum		(1 Mark)
range		(1 Mark)
percent_conv		(1 Mark)
min		(1 Mark)

In *Sort* Function:

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

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

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

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

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 --