CITY UNIVERSITY OF HONG KONG

Course code & title : CS3342 Software Design

Session : Semester B 2014/15

Time allowed : Two hours

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

1. This paper consists of **6** 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: ______________________

Programme: ______________________

Seat Number: ______________________

	Answer all questions						
	CILO 1	*CILO 2*	*CILO 4*	*CILO 3*	*CILO 4*	*CILO 5*	
Question	**1**	**2**	**3**	**4**	**5**	**6**	**Total**
Max	**10**	**20**	**5**	**35**	**25**	**5**	**100**

Question 1 – Software Engineering Development Processes – CILO1 (10 Marks)

The following shows a list of terminologies, and a table of 10 descriptions.
In the **Terminology ID** column of the table, write down *exactly one* terminology ID that matches the description/illustration.

1. Prototyping
2. Incremental Process Model
3. Evolutionary Process Model
4. Rapid Application Development (RAD)
5. Water Fall Model
6. Concurrent Engineering Model
7. Component-based Software Engineering
8. Software Component
9. Software Engineering
10. Software Design
11. Software Maintenance
12. Software Testing
13. Spiral Model
14. Software Requirement
15. Software Quality Management

Fill in the Terminology ID according to the list above (1 Mark Each):

Terminology ID	Description/Illustration
	Reuse and integration is its core objective where systems are built from integrating existing commercial off the shelf components.
	The study and an application of engineering to design, development, and maintenance of software.
	The ______ is easy to follow, it is structured which provide a template into which methods for analysis, design, code, and testing and maintenance can be utilized. Due to its sequential nature, it does not reflect reality, which relies heavily on completed requirements at start, and problems in the specification may be discovered very late during coding or integration stages.
	_______ provides a service without regard to where the component is executing or its programming language. Their interface is normally published and all interactions are through the published interfaces to allow reuse and integrate with other systems.
	The _______ provides the core functionalities first, and iteratively adds a new functionality in each release cycle until the product is completed.
	The _______ builds on incremental model with emphases on short development cycle. It involves multiple teams, and assumes that the system can be modularized. It will fail if we do not have strong and skillful teams.
	Core requirements are somewhat understood but additional requirements are changing fast. Specification, development and validation activities are concurrently carried out with rapid feedback to customer across these activities, it is suitable for rapidly changing requirements, but time estimation and project completion date may be difficult to forecast.
	It explicitly use the divide and conquer principle in the process, where each team works on its own allocated component in parallel with the other teams, and integrates these components periodically.
	This provides a high level conceptual understanding of the system design, and useful platform to allow better communication with customer to identifying requirements expected. However it may be leading customer with incorrect perception about the final product.
	All the activities involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems following requirements specification and before programming.

Question 2 – User Requirements Specification – CILO2 (20 Marks)

Octopus-enabled ATM system

The **Octopus Card** is a reusable countactless stored value smart card for making electronic payments in online or offline systems in Hong Kong, commonly used for MTR trains and subways. There are some new ATM bank cards integrating the Octopus smart card chipset within, so it can be used as a normal Octopus card as well as a traditional ATM card(i.e. 2 in 1).

The new **Octopus-enabled ATM system** accepts this kind of new ATM card, and allows recharge/top-up the Octopus card from the customer's bank account directly.

Both Customer and ATM Staff operate on the Octopus-enabled ATM system. A Customer, who draws money from his account in cash, or recharges/loads value to the octopus card built into the ATM card. In addition, the customer can also check account balance. ATM staff performs maintenance of the ATM system and fills money into the ATM.

Use cases should include: Withdraw Cash, Recharge Octopus, Check Account Balance, and Maintenance, Fill ATM. Also the system needs to handle a use case when the Octopus Chip is out of order. The ATM needs to handle ATM Exception cases; they are Insufficient Funds when insufficient account balance detected, and Transaction Abort when the customer selected the cancel button without completing the transaction. Remark: it is possible to use inheritance between use cases, and use Include/Extend when appropriate.

a) Based on the above, draw a use case diagram for the Octopus-enabled ATM system. **(10 Marks)**

b) Based on the same case study in (a), complete from Step 6 onward in the following table to describe the use case ***Withdraw Cash*** under typical course of event, and alternative course of events. You may use your own understanding of an ATM system. **(10 Marks)**

Use Case Name:	**Withdraw Cash**	
Actor(s):	Customer	
Description:	This use case describes the process of customer withdrawing cash.	
Reference ID:	ATM-1.0	
Typical course of events:	**Actor Action**	**System Response**
	Step 1: The Customer inserts his bankcard into the ATM. **Step 3**: The Customer enters his correct PIN and confirms it. **Step 5**: The Customer selects the "Withdraw Cash" function by pressing the appropriate button.	**Step 2:** The ATM system shows the "*enter PIN screen*" to the Customer. **Step 4**: The ATM welcomes the Customer and opens the main menu. **Step 6:**
Alternative course of events:		
Pre-condition:		
Post-condition:		

Question 3 – Roles of Variables – CILO4 (5 Marks)

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	An array which is only used for rearranging its elements after initialization.

Table 1: Roles of Variables in Software Programs

```
public int CalculateSquaredSum (int max)
{
    int sum = 0;
    int squared = 0;
    Scanner sc = new Scanner(System.in);

    for (int i=0; i<max; i++) {
        System.out.printf ("Enter number %d: ",i+1);
        int input = sc.nextInt();
        sum = sum + input;
    }
    squared = sum * sum;
    return squared;
}
```

Based on the function ***CalculateSquaredSum*** above, correctly classify the role of each variable in the table below. You may use Table 1 for your reference. **(5 Marks)**

Variable	Role of Variable (1 Mark Each)
i	
sum	
squared	
input	
max	

Question 4 – Object Oriented Analysis – CILO3 (35 Marks)

(a) Associations / Aggregation / Composition (5 Marks)

There are three kinds of class linkages in class diagrams. Based on the following code fragments describing the class **Car** and class **Engine** relationship, correctly indicate which one of the three applies (Associations / Aggregation / Composition) by completing the diagram, and state its relationship below.

```
class Engine {
   public Engine() { };
   public void work() {
      System.out.println ("Engine Works...");
   }
}
```

Draw and Determine their correct line/linkage relationship (Associations / Aggregation / Composition) in the following diagrams.

```
class Car {
   private final Engine engine = new Engine();

   public void move() {
      engine.work();
   }
}
```

This is ____________________ relationship.

```
class Car {
   public void move(Engine engine){
     engine.work();
   }
}
```

This is ____________________ relationship.

```
class Car {
   private Engine engine;

   public void installEngine (Engine engine) {
      this.engine = engine;
   }
   public void move() {
         engine.work();
   }
}
```

This is ____________________ relationship.

(b) Object Orientation (20 Marks)

Study the following code fragments and answer the following questions.

```
class Person
{
   private String name;
   private int yearOfBirth;
   private Address address;

   Person(String name, int yearOfBirth){
      this.name = name;
      this.yearOfBirth = yearOfBirth;
   }

   public void setAddress(String street, String town, String country){
      address = new Address(street, town, country);
   }

   public Address getAddress(){
      return address;
   }
}
```

```
class Address
{
   private String street;
   private String town;
   private String country;

   public Address(String street, String town, String country)
   {
      this.street = street;
      this.town = town;
      this.country = country;
   }

   public String toString()
   {
      return street + "\n" +
            town + " " + "\n" +
            country + "\n";
   }
}
```

```
class Staff extends Person
{
   private String room; // Staff Room number

   Staff(String name, int yearOfBirth, String roomNumber)
   {
      super(name, yearOfBirth);
      room = roomNumber;
   }

   public void setRoom(String newRoom)
   {
      room = newRoom;
   }

   public String getRoom()
   {
      return room;
   }

}
```

```
class Student extends Person
{
   private String SID;    // student ID number

   public Student(String name, int yearOfBirth, String
studentID)
   {
      super(name, yearOfBirth);
      SID = studentID;
   }

   public String getStudentID()
   {
      return SID;
   }

}
```

- Based on your observation of the above code fragment, draw a complete class diagram. **(10 Marks)**

- Assuming that there are a number of University Events at the University, and Staffs and Students within the University organize them, which means that there are many Events organized by many People (Staff/Student). **Extend** the existing class diagram design to correctly show classes **EventAllocation** and **Event**. Class **Event** should contain important attributes of Venue, Date, Event_Name. Class **EventAllocation** should record which **Person** is Responsible for which **Event**, more importantly it must contains an attribute for the person's Role (responsibility) in the allocated event as String. **(10 Marks)**

4(b) Class Diagram:

4(b) Class Diagram (con't).

4(c) Sequence Diagram (10 Marks)

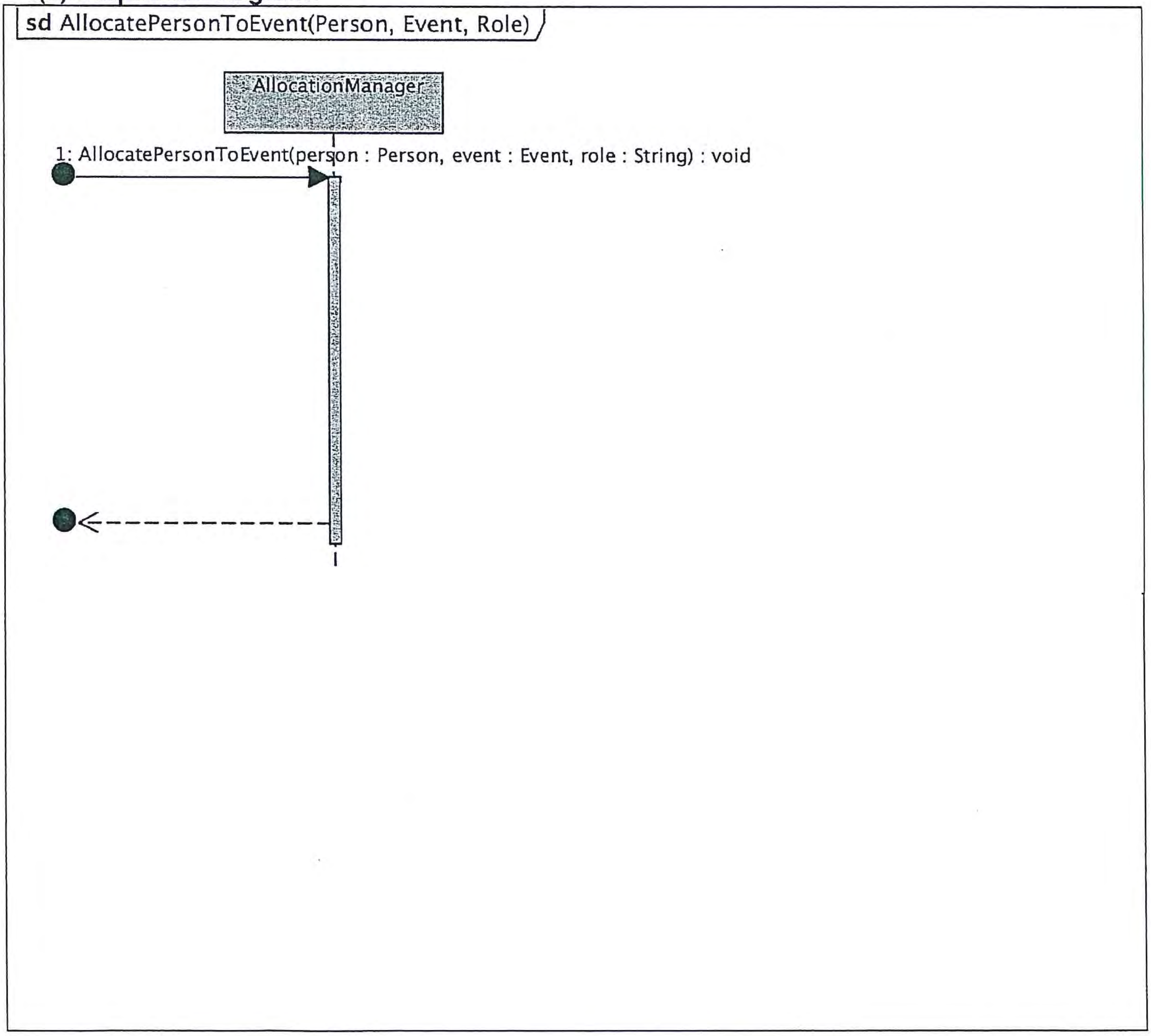
Continue from the previous sections. There is an **AllocationManager** class keeps a <u>List of **EventAllocations**</u>. Based on the following coding sequence draw a complete sequence diagram to describe the execution sequence of **AllocatePersonToEvent(). (10 Marks)**

```
import java.util.*;

public class AllocationManager
{
   private List<EventAllocation> events_alloc_list = new ArrayList<EventAllocation>();

   public void AllocatePersonToEvent (Person person, Event event, String role)
   {
     EventAllocation alloc = new EventAllocation (person, event);
     alloc.setRole (role);
     events_alloc_list.add (alloc);
   }
}
```

4(c) Sequence Diagram:


sd AllocatePersonToEvent(Person, Event, Role)
: AllocationManager
1: AllocatePersonToEvent(person : Person, event : Event, role : String) : void

Question 5 – Software Design Principles and Patterns – CILO4 (25 Marks)

Based on the same case study of Question 4, **EventAllocation** is a class to record different _Roles_ of a **Person** being allocated to a specific **Event**. The current roles include **Manager, Assistant and Helper**. There should be a function in **EventAllocation** *getWorkHours()* for different roles, for example Manager needs to work for 20 hours, Assistant and Helper need to work for 15 and 10 hours respectively. The challenge is, from time to time, their roles/responsibilities are subject to change by using a function called *setRole(role),* where role is {*Manager, Assistant, Helper*}.

(a) One of the possible solutions is to use the *State Pattern*, draw a class diagram of **EventAllocation** and all the related classes required using *State Pattern* for the design. **(10 Marks)**

(b) Based on the design in (a), there may be an issue of memory efficiency **in creating multiple state objects**. How would you improve this situation using a design pattern, and which design pattern would you apply? Use a class diagram to show and illustrate your understanding together with your answer (only a selected class example is sufficient). Write code fragments to illustrate your implementation and understanding (only a selected example is sufficient). **(7 Marks)**

(c) Continue from the previous case study. Each **Event** has a Newsletter available to all students and staffs, subject to their subscriptions. The Event class now has a new attribute called *newsletter: String*, with two additional functions: ***setNewsLetter(String news)*** and ***getNewsLetter()***.

- How would you design and extend the system so that both Students and Staff would be able to subscribe and unsubscribe (needs to provide these functions) to individual event's newsletter, and when *newsletter* is updated by an event, all the subscribers will be notified for updates?
- Describe your answer with a suitable Design Pattern.
- Provide code implementation examples to support your answer. **(8 Marks)**

Question 6 – Professional Ethics in Software Engineering – CILO5 (5 Marks)

Banana Software Company produces their famous ***Banana Web Browser*** to all Internet users. ***Pico*** is the programmer and ***Mac*** supervises ***Pico***.

- ***Pico*** is responsible for the rendering engine of the web browser application, and he has discovered a minor bug within the software that was under his responsibility. The deadline for the next release is in two days, and he knew that he could not fix the bug and retest the entire application.
- ***Pico*** evaluated the severity of the bug, and learned that there is only 1% chance that the bug would cause the application to crash or malfunction.
- ***Pico*** reports to ***Mac*** that the application has no bug and will be fully functional for the next release.
- ***Mac*** is surprised by ***Pico***'s report because ***Mac*** also discovered the bug himself, and it was verified with ***Pico*** that the existence of the bug was not reported accordingly.
- ***Mac*** therefore coaches ***Pico*** about professionalism, and suggests ***Pico*** to fix the bug and postpones the release date if it cannot be fixed before the scheduled release date.

Explain at least **two codes of ethics** *Pico violates*; and explain at least three codes of ethics *Mac follows.* (Appendix 1 lists out all the eight areas of code of ethics in software engineering for your reference.)

Pico **violates (2 Marks):**

Mac **follows (3 Marks):**

Appendix 1 - Code of Ethics in Software Engineering

Software engineers shall, in their work capacity,

1) [*Public interest*] Act consistently with public interest
2) [*Client and employer*] Act in the best interests of their clients and employer
3) [*Product*] Develop and maintain the product (e.g., software and documentation) with the highest standards possible
4) [*Judgment*] Maintain integrity and independence (of oneself)
5) [*Management*] Promote an ethical (e.g., equal opportunity, match task against skill level instead of friendship) approach in management of subordinates (who are managed by you)
6) [*Profession*] Advance the integrity and reputation of the profession as software engineers
7) [*Colleagues*] Be fair and supportive to colleagues
8) [*Self*] Participate in lifelong learning (as technology changes fast)

--- END ---