CITY UNIVERSITY OF HONG KONG

Course code & title : CS3342 Software Design

Session : Semester A 2012/13

Time allowed : Two hours

This paper has **16** pages (including this cover page) and 4 pages of Appendix.

---

1. This paper consists of 8 questions.
2. Answer ALL questions in Section A and THREE questions in Section B. All questions are independent of one another.
3. Directly write your answer on this question script.

---

*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				Answer any three				
Question	1	2	3	4	5	6	7	8	Total
Max	10	10	10	10	20	20	20	20	100

Section A (40 marks). Answer <u>all</u> questions. Each question carries 10 marks

1. Software Process (10 marks)

(a) Give an *example* that explains what an incremental development model is. You may illustrate your example via a diagram supplemented with texts. **(5 marks)**

(b) For software development, the use of the *incremental development model* is more suitable than use of the *waterfall model*. Do you agree? Give **<u>two</u>** reasons to support your answer. **(5 marks)**

Reason 1:

Reason 2:

2. Design Principles (10 marks)

Study the Scenario *A* in Appendix 1, and create a design solution in the form of UML class diagram that applies **one** good object-oriented design principle to solve **one** of the following problems on top of the Scenario *A*. Your solution should show all classes, relationships, attributes and methods.

Select **one** of the following problems to solve:

- **Problem 1:** Some methods of the classes `SalesRecord` and `PaymentRecord` are inapplicable to them (e.g., `pay()` in `SalesRecord` and `addItem()` in `PaymentRecord`).

- **Problem 2:** `getTotalAmt()` of `SalesRecord` should not directly retrieve the value of the `Price` attribute of its `SalesItem` objects.

3. Roles of Variables (10 marks)

Study either the following C++ code listing or the following Java code listing, and identify the role of each program variable declared in the code listing.

In C++:

```
#include <iostream>
using namespace std;

class Growth {

public:
static void main(int loan) {

    float current, interest; int max = 10; float half_life;

    current = loan;
    half_life = 0;
    for (int i = 1; i <= max; i++) {
      interest = 0.05 * current;
      if ( current * 2 > loan && half_life == 0)
           half_life = i;
      current = current +interest;
    }
    cout << ("After " + max + "year, the loan is ") << current << endl;
    cout << ("Half-life is") << half_life << endl;

   }
};
```

In Java:

```
public class Growth {

   // Put your loan value into the args[0]
   public static void main(String[] args) {

    float current, interest; int max = 10; float half_life;

    // change the input string into an integer
    int loan = Integer.parseInt(args[0]);

    current = loan;
    half_life = 0;
    for (int i = 1; i <= max; i++) {
      interest = (float) (0.05 * current);
      if ( current * 2 > loan && half_life == 0)
           half_life = i;
      current = current +interest;
    }

    System.out.println("After " + max + "year, the loan is " + current);
    System.out.println("Half-life is" + half_life);

   }
}
```

loan:

current:

interest:

max:

half_life:

i:

4. Professional Ethics in Software Engineering (10 marks)

Study **Case 1** below. **Explain** whether each person below violates or follows any **one** code of ethics in software engineering.

(Note: Appendix 3 lists out the eight areas of code of ethics in software engineering.)

Case 1: *Lenka* supervises both *Leon* and *Gina* in a project.

- ***Leon*** knows that his program has a bug, but he is going to leave the project for further study to improve his Computer Science skills. So, he decides to keep silent because debugging this program takes time.

- ***Gina*** tests *Leon*'s program, and finds out this bug. However, she is not responsible for coding and is aware of *Leon*'s departure. So, she decides to keep silent even though she intends to assist Leon to debug the program.

- ***Lenka*** reviews *Leon*'s program, and spots this bug. She explains the bug to both *Leon* and *Gina*, instructs *Leon* to fix the bug, and instructs *Gina* to test the program more comprehensively.

Leon:

Gina:

Lenka:

Section B (60 marks). Answer any three questions. Each question carries 20 marks.

5. Design Patterns (20 marks)

(a) In Scenario A (see Appendix 1), the class `CashierRegister` is responsible to create `SalesRecord` objects and `PaymentRecord` objects. However, `CashierRegister` goes through the `Record` class to use any operation of these two classes. Illustrate a **design pattern based solution** in which `CashierRegister` needs not to know the presence of either `SalesRecord` or `PaymentRecord` but is still able to obtain new objects of these two classes. Show you answer in both UML class diagram and sequence diagram(s) **(15 marks).**

(b) Select any **one** of the following design patterns. Discuss how your selected design pattern can be applied or cannot be applied to Scenario *A* (see Appendix 1). **(5 marks)**

- State Pattern
- Command Pattern
- Observer Pattern

6. Factoring in Design Constraints (20 marks)

Scenario *B*: An online shopping activity is as follows.

- The *shopping cart* contains a number of items, each of which is an item for a customer intended to purchase at the same time.
- A *shopping cart page* is a webpage that lists out the content of a shopping cart, item by item. The page also shows a quantity to be purchased for each item shown.
- A customer may modify the quantity value of each item on the shopping cart page.
- A customer may press the [Revise] button at any time.
- Whenever the [Revise] button is pressed, the system stores the quantity value of each item shown on the shopping cart page, computes the new grant total of the shopping cart, and refreshes the shopping cart page to display the new grant total of the shopping cart accordingly.

(a) Identify all the classes in Scenario *B*. For each of your identified classes, identity its analysis class stereotype such as entity class, boundary class, and control class. State the responsibility of each class according to Scenario *B*. **(8 marks)**

Class	**Analysis Class Stereotype**	**Responsibility of the Class**

(b) Using **either** class diagram **or** sequence diagram to illustrate how your solution factors in the **Design Constraint 1** below on top of **Scenario B**. **Explain** your solution on why it can handle the design constraint. **(12 marks)**

- **Design Constraint 1**: A shopping cart should allow multiple shopping cart pages to show its information; otherwise, if a web user opens the shopping cart page in two browser tab pages, the information shown on the two browser tab pages will be inconsistent.

7. Domain Modeling (20 marks)

Study **Case 2** below.

Case 2 (Lift Control System at CityU Academic 5):

- Each lift serves all floors of the building.
- At each floor, there is a pair of directional buttons (i.e., an [up] button and a [down] button) to call for the service of the lifts.
  - Pressing a button at a floor indicates a request of the lift service at that floor.
  - However, to successfully press a button, a passenger should also key in the floor number that the passenger wants to go.
- When a lift arrives at a floor, it will inform those passengers who want to go to a particular floor to enter the lift.
- If the lift is currently serving traffic in a certain direction, it will only answer the requests for services in the same direction.
- Inside each lift, there is only one door control button to keep its opened door remained open if this button is being pressed. The lift will automatically close the door after an intelligent amount of time period if the button is not pressed.
- For security reasons, the lift access to a particular floor can be disabled.

(a) Assess whether each noun in the following table is a domain entity that we should model it in the domain model of the above lift control system. **(5 marks)**

Noun	Assessment with explanation
Directional button	
Lift service	
Passenger	
Lift	
Security reason	

(b) Perform domain modeling on Case 2. Give your answer in the form of class diagram that shows classes, attributes, and relationships between classes. **(15 marks)**

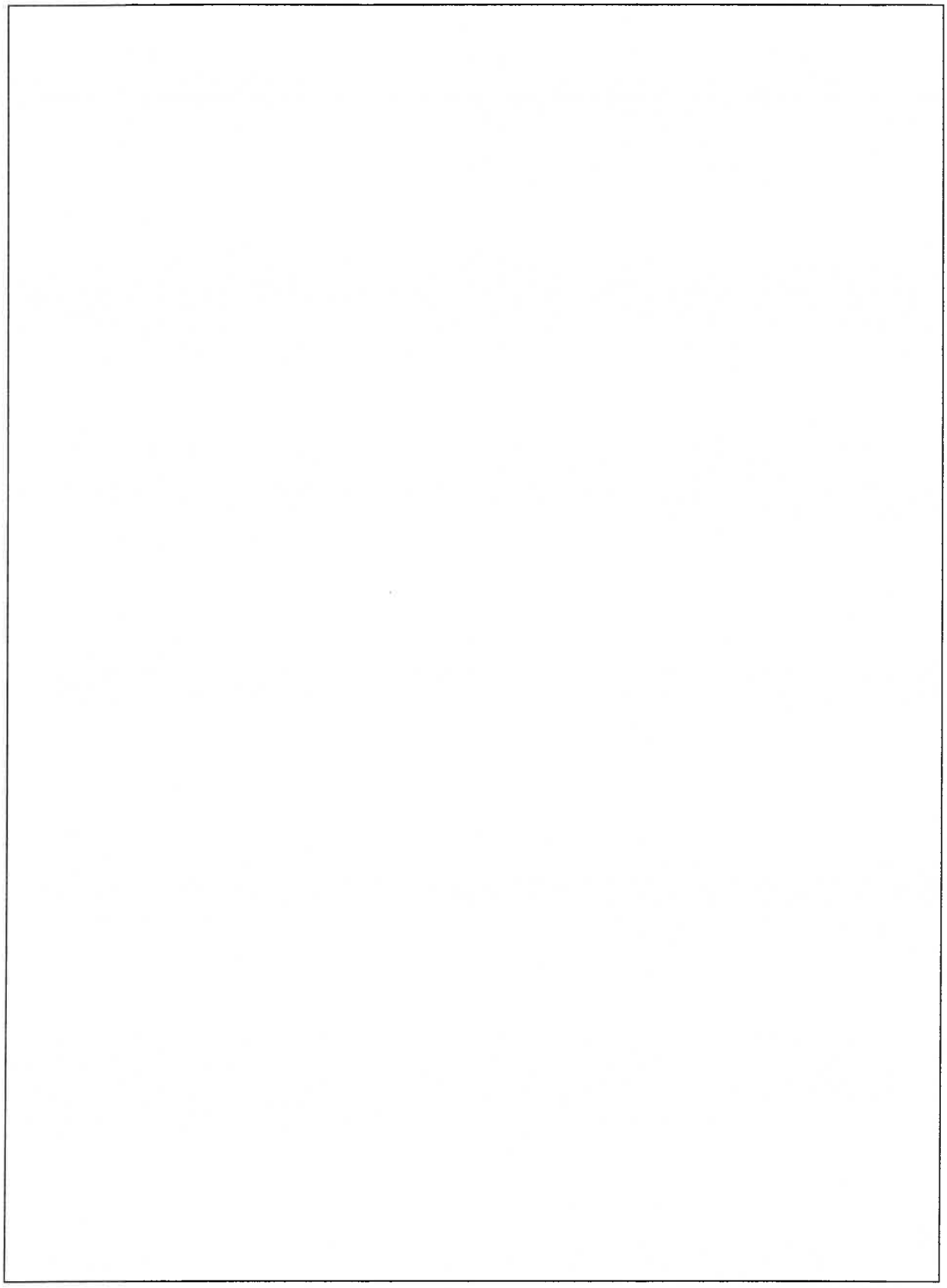
8. Requirement Capturing and Specification (20 marks)

(a) After receiving a use case, a programmer jumps to immediately implement the use case. Explain **two** steps related to requirement capturing and specification that may have been missed by the programmer's approach. **(5 marks)**

(b) **Write the alternative flows** of the UC1 so that the use case can handle all of the following newly discovered requirements. **(15 marks)**

- **Comment 1:** A course instructor may prepare multiple versions of the same examination paper.
  - She/he may predefine a particular student to take a particular examination paper version.
  - She/he may want to firstly know which particular student wants to get an examination paper (after confirming the status of the coursework requirement of the student) then give out an appropriate examination paper version to this particular student to attempt.

- **Comment 2:** A student may go to the restroom during the examination period.
  - The screen of the computer should be blackened and the keyboard should be locked whenever the camera of the computer cannot detect the face of the student.
  - The system should automatically brighten the screen and unlock the keyboard whenever the same student is detected by the camera.



- END -

Appendix 1: **Scenario *A***

In a supermarket, each cashier operates a Cashier Register, which supports a number of operations:

- The `newSales()` operation creates a blank new `SalesRecord` of the current date.
- The `completeSales()` operation calls the `endRecord()` operation of `Record`.
- The `addSalesItem()` operation calls the `addItem()` operation of `Record`.
- The `makePayment(x, payno)` operation creates a blank new `PaymentRecord` of the current date and with the payment number `pno` followed by calling the `pay(x)` operation of `Record`. In `PaymentRecord`, `pay(int amt)` simply assigns `amt` to its attribute `PaidAmount`. The constructor of `PaymentRecord` assigns the input value to the attribute PaymentNo.
- The `cancelPayment()` operation also calls the `endRecord()` operation of `Record`.
- The method bodies of both `pay()` of `SalesRecord` and `addItem()` of `PaymentRecord` contain no any code.

All classes except `Record` in the scenario are concrete classes.

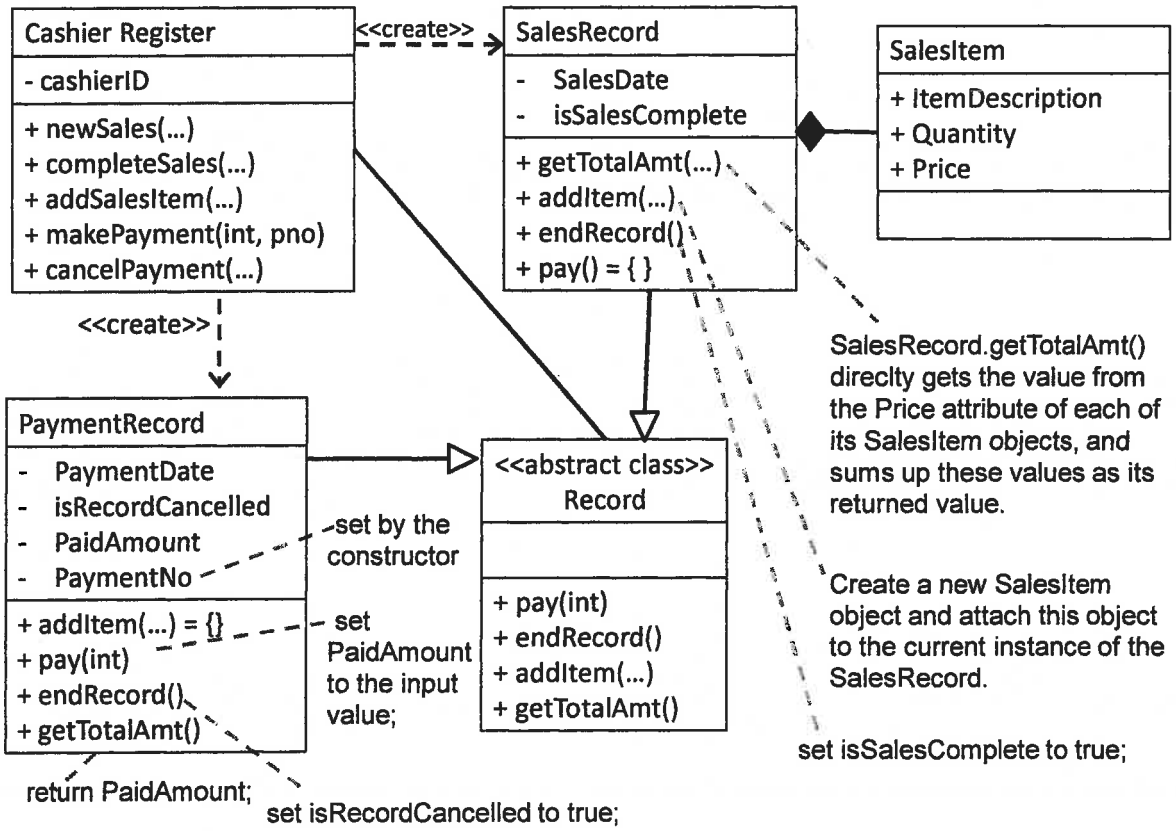


Figure 1. A Class Diagram for Cashier Register Domain

Appendix 2: **Popular Roles for Variables in Programs**

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.

Appendix 3: **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)

Appendix 4: **A Use Case**

Use Case UC1: Take Computer-based Examination	
Basic Flow	
Actor Action (or Intention)	*System Responsibility*
1. *Student* arrives at an examination venue and signs on the system with the student ID	
	2. Recognize the student's face according to the student ID. 3. Verify the student identity being valid for the examination venue. 4. Confirm that the student has met the coursework requirement of the course. 5. Randomly present an applicable examination paper to the student.
6. *Student* fills in the answers for the examination paper within the allowable time period of the exam	
	7. Record the student answers as a part of the student performance for the course 8. Calculate the mark that the student receives from the filled answer. 9. Map the examination mark to a grade. 10. Notify *Course instructor* about the performance of the student 11. Generate a grade sheet to the student
12. *Student* leaves the examination venue with a grade sheet.	

- END OF APPENDIX-