NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

SEMESTER 1 (2008/2009)
EXAMINATION FOR

CS2103 – SOFTWARE ENGINEERING

Dec 2008 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **FIVE (5)** questions and comprises **TWELVE (12)** printed pages, including this page.
2. Answer **ALL** questions within the space in this booklet
3. This is an Open Book examination.
4. Please write your Matriculation Number below.

MATRICULATION NO: _____

This portion is for examiner's use only

Question	Marks	Remarks
Q1	/10	
Q2	/10	
Q3	/12	
Q4	/12	
Q5	/11	
Total	/55	

Question 1 (10 marks)

(a) State briefly, any two reasons, for using UML in software analysis and design processes. **(2 marks)**

(b) State two possible reasons for a software failure . You may use an example to support your answer. **(2 marks)**

(c) Behaviour of an object_P when executing operation_1 may depend on properties of object_Q. The questions arises whether the operation should take object_Q as a parameter, or object_Q's property, i.e. operation_1(Class_Q x) vs operation_1(Property_Q x) **(2 marks)**

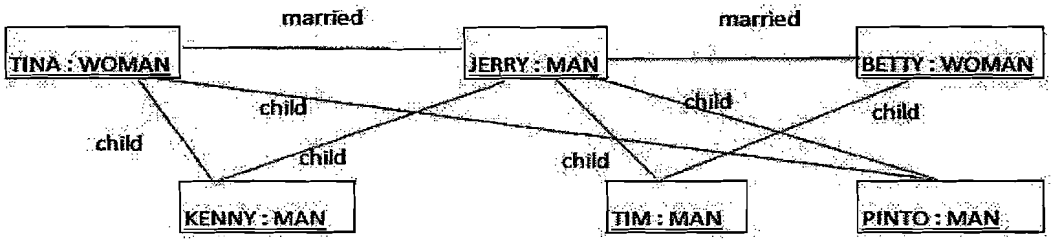
The answer is

The reason is

(d) What are the artifacts or deliverables or documents produced at the end of each of these development phases irrespective of development approach. State one for each phase in the space given **(2 marks)**:

Requirement analysis	
Design	
Verification	
Validation	

(e) Consider the following instance diagram and draw the corresponding class diagram **(2 marks)**.

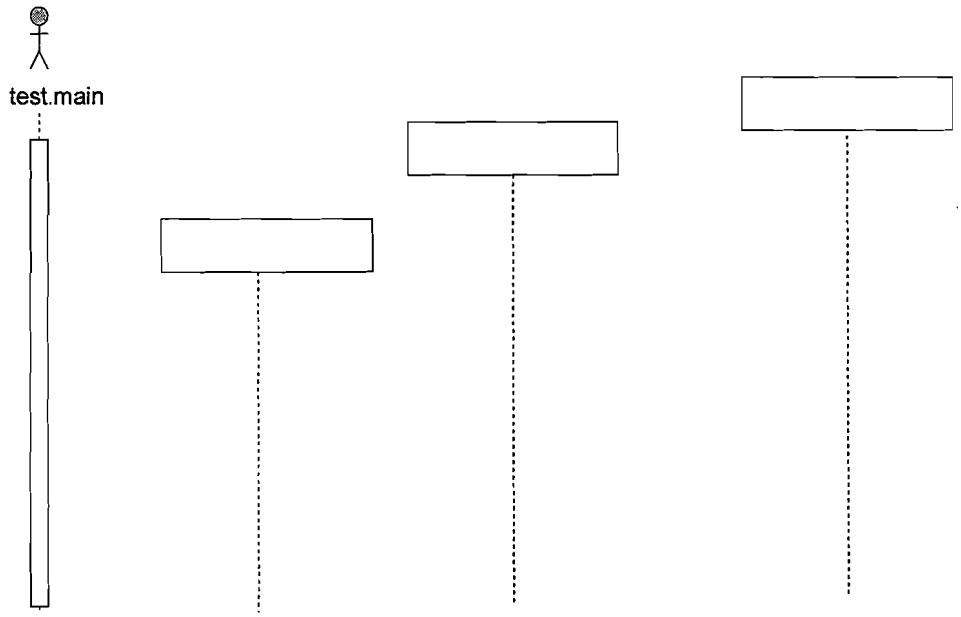


Answer :

Question 2 (10 marks)

(a) Complete the skeletal interaction diagram (on next page) that shows the method calls of the main method given below. **(5 marks)**

```
public class test {
    public static void main(String [] args) {
        String s = "Hello world";
        Scanner in = new Scanner(s);
        while (in.hasNext())
                printer.print(in.next()); //printer is of type Printer
    }
}
```



(b) An XObject is controlled by events: ***powerOn, powerOff, Start, Stop, guardOn, guardOff***
Information about states of XObject are as follows :

- *Safe* – power should be off, and guard should be on
- *Ready* – power to be on, and guard to be on
- *Maintain* – power to be off, and guard to be off
- *UnSafe* – power to be on , and guard to be off
- The object transits to a ***run*** state only when power and guard are on, and a **start** event is triggered. It continues to be in ***run*** state until the **stop** event or **poweroff** event are triggered.
- Assume initial state of the object is ***Safe*** state.

Complete the following skeletal state chart with valid labeled transitions based on the above information. **(5 marks)**

Safe

Ready

Run

Maintain

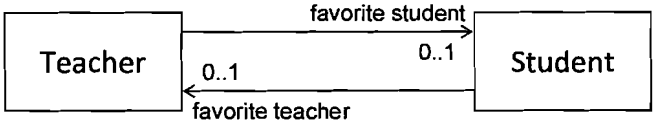
Unsafe

Question 3 (12 marks)

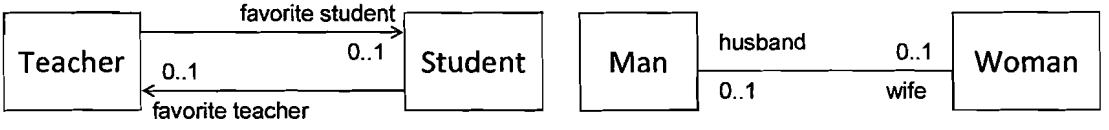
(a) Is the following implementation a good translation of the association links between the Teacher class and the Student class given in the class diagram? Explain your answer in 1 – 2 sentences **(2 marks)**.

```
class Teacher{
  private Student favStudent;
  void setFavStudent(Student s){
    favStudent = s;
  }
}
```

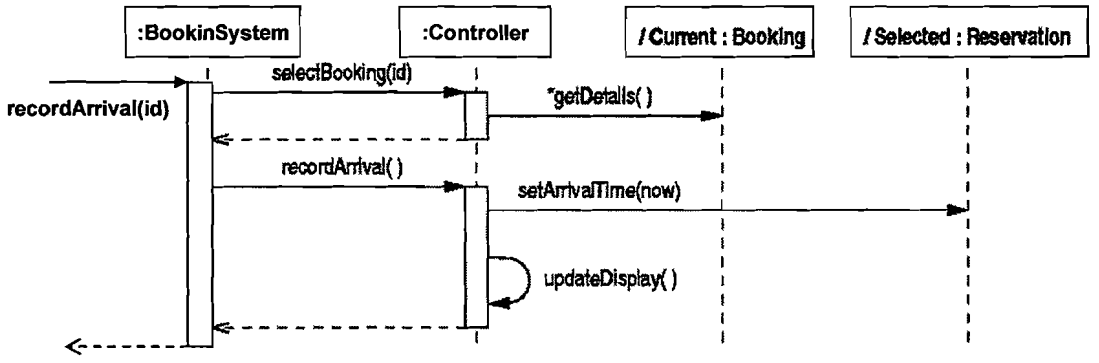
```
class Student{
  private Teacher favTeacher;
  void setFavTeacher(Teacher t){
    favTeacher = t;
  }
}
```



(b) In terms of maintaining referential integrity, what is the difference between implementing the following two diagrams? Explain in no more than 5 sentences. **(2 marks)**



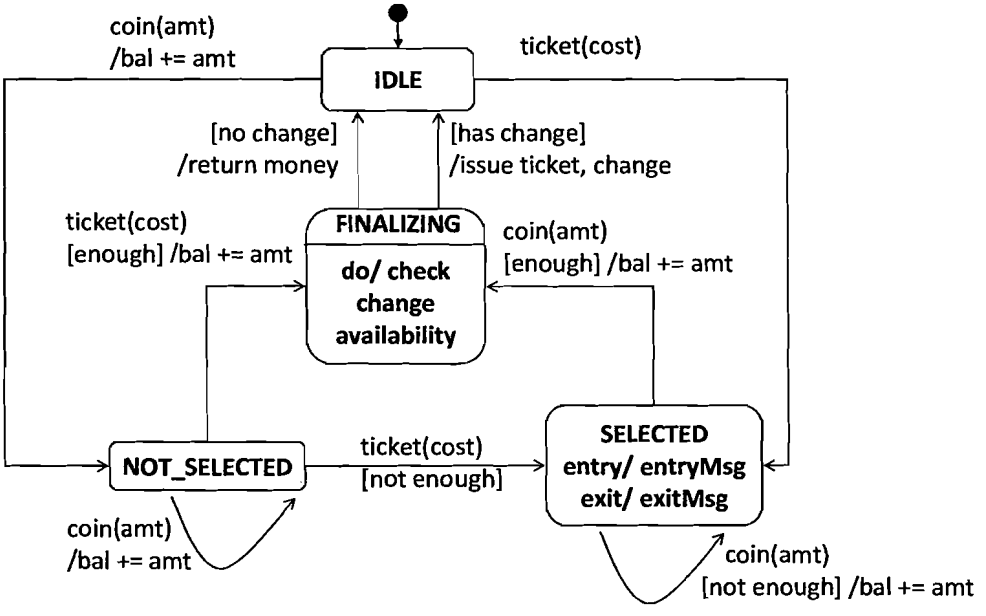
(c) Translate into code the recordArrival(id) method of the BookingSystem class shown below **(2 marks)**




```
Controller c;
...
void recordArrival(int id){
        //write your code here
```

}

(d) Given below is a simplified state chart for a ticket machine, and the partial code (next page) for the coin(double) and the ticket(double) methods. Complete the two methods by inserting calls to entryMsg() and exitMsg() methods where appropriate. No other modifications are required. You are to use the blank space between the given code lines to add your own code. Note that the method doFinalizing() implements the activity state FINALIZING. Consult the invigilator if you do not understand the syntax of the code **(6 marks)**



```
// variables
   Enum States {IDLE, NOT_SELECTED, SELECTED}
   private States state ;
   private double bal; //amount of money inserted
   private double cost; //value of the ticket selected
     ...
```

```
public void coin(double amt) {

     switch (state) {
       case IDLE:

          bal += amt;

          state = States.NOT_SELECTED;

          break;
       case NOT_SELECTED:

          bal += amt;

          break;
       case SELECTED:

          if( (bal+amt) >= cost ){

                bal += amt;

                doFinalizing();

          } else {

                bal += amt;

          }

          break;
     }

  }
```

```
public void ticket(double cost) {

   switch (state) {
     case IDLE:

        this.cost = cost;

        state = States.SELECTED;

        break;
     case NOT_SELECTED:

        this.cost = cost;

        if(bal<cost){

           state = States.SELECTED;

        }else {

           doFinalizing();

        }

        break;
     case SELECTED:

        break;
   }
```

Question 4 (12 marks)
Assume you are testing the add(Item) method specified below.

ItemList
add(Item):void contains(Item):boolean size():int

Assume i to be the Item being added.
preconditions:
i != null [throws invalidItemException if i == null]
contains(i) == false [throws duplicateItemException if contains(i) == true]
size() < 10 [throws listFullException if size() == 10]

postconditions:
contains(i) == true;
new size() == old size()+1

Invariants:
0 <= size() <= 10

(a) What are the equivalence partitions relevant to testing the add(Item) method? **(3 marks)**.

(b) What are the boundary, and non-boundary, values you will use to test the add(Item) method? **(3 marks)**.

(c) The code below shows how the add(Item) method deals with the preconditions.

```
void add(Item i) throws Exception{
        if(isNull(i)) throw new invalidItemException();
        if(contains(i) == true) throw new duplicateItemException();
        if(size() == 10) throw new fullListException();

        //code for adding i to an internal data structure
}
```

Assume methods isNull(Item), contains(Item) and size() are working correctly (already tested). Design a set of test cases to test the add(Item) method by considering the equivalence partitions and boundary values from your answers to (a) and (b) above. For increased efficiency in testing, you should avoid testing redundant combinations that are unlikely to discover any new bugs not discovered by the rest of the test cases. Test case 0 is for illustration only, not counted as part of your answer. **(6 marks)**

	Test input : i	Test input: ItemList object	Expected output
0	Any non-null Item	5 Items in the list, contains(i)==false	contains(i)==true size()==6
1			
2			
3			
4			
5			
6			
7			
8			

Question 5 (11 marks)

(a) Draw the CFG for the following code. Consult the invigilator if you do not understand the syntax of the code (**2 marks**).

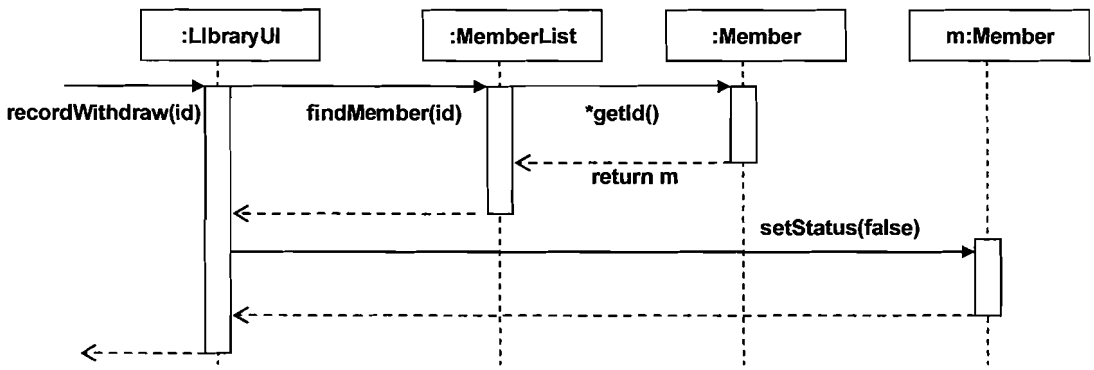
```
void displayOdd(int limit) {
    int i  = 0;
    do {
        print ( i%2==0 ? "-" : i );  //this is a shorthand if-else construct
        i++;
    } while (i <= limit);
}
```

(b) What is the MCC for the above CFG? (**1 mark**)

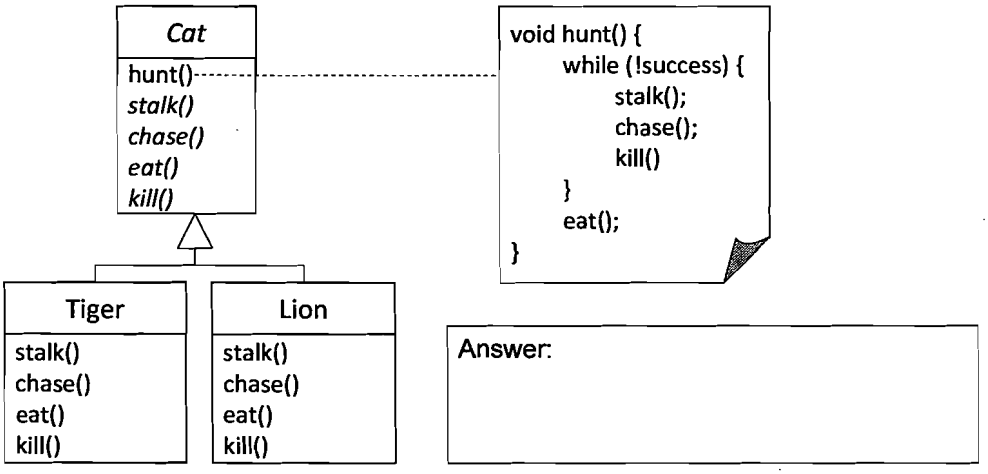
(c) List independent basis paths for the above CFG. For each realizable basis path, also give test data to execute the path, and the expected result. **(3 marks)**

path	realizable (Y/N)?	limit	Output

(d) Given below is a sequence diagram from a library system. Redraw this diagram after applying the Façade pattern to the system. The objective of applying the pattern is to shield the UI layer from the complexities of the business logic layer **(2 marks)**.



(e) Identify the GOF design pattern used in the below diagram. Hint: while all cats follow the same high-level algorithm in hunting, they may differ in how each step is carried out. Methods/classes in italic are abstract (**1 mark**).



(f) Give two arguments in support and two arguments against the following statement.
Statement: Assuming there is no external client, it is OK to use big bang integration for CS2103 course project (**2 marks**).

Arguments in support (give two):

Arguments against (give two):

----------**End of Exam Paper**----------