NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

SEMESTER I (2004-05)
EXAMINATION FOR

CS2103: SOFTWARE ENGINEERING

November 2004 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of **seven(7)** questions and comprises of **sixteen (16)** printed pages. Answer **ALL** questions.
2. Write your answers in the **blank spaces** in this answer book only.
3. This is an **OPEN BOOK** examination.
4. Please fill in your **Matriculation Number** below. Also write your matriculation number on the top right hand corner of every page.

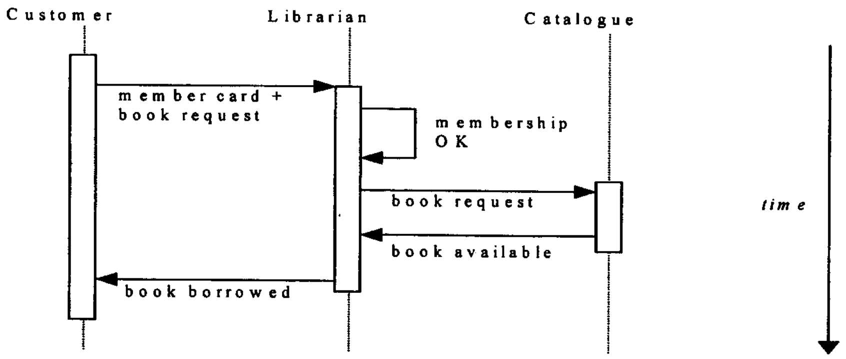
Matriculation Number: □□□□□□□□□□

For Official Use Only :

	Marks
Question 1 (max 10)	
Question 2 (max 15)	
Question 3 (max 10)	
Question 4 (max 15)	
Question 5 (max 25)	
Question 6 (max 15)	
Question 7 (max 10)	
TOTAL:	

Question 1 **10 marks**

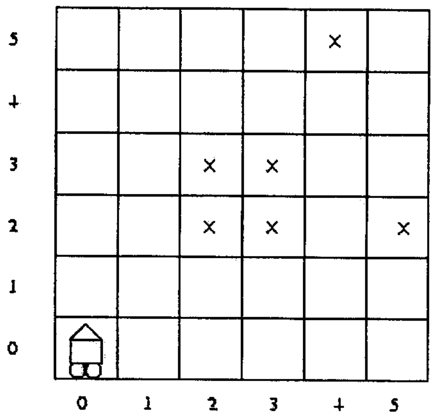
Given the following sequence diagram, draw the equivalent collaboration diagram. Customer, Librarian, and Catalogue represent objects in the following diagram.



Answer 1:

Question 2 **15 marks**

A vacuum cleaning robot moves around to clean a house. The house floor can be divided grid-like into a number of equally sized squares (conveniently corresponding to the unit of movement of the robot, see the example figure below). It can only move to one of its next positions within the floor. If the robot is over a dirty square, then it will clean the square. The robot will move around as long as there is a dirty square in the place. If all squares are clean in the house, then it will rest. Of course, at any time, a square can be made dirty (by external environment). Your task is to build a UML statechart to capture the essential behavior of this system.



Hint: consider the following variables in your system:
vp ---- (Position, Position)
dirtyones ---- Set of (Position, Position)

Answer 2:

Question 3 **10 marks**

A family with a father, a mother, a child and an old grandmother, are going to cross a bridge (a shelter at each end) from the east side to the west side. The bridge is very narrow that allows at most two persons to pass at the same time. It is raining hard that they cannot move without an umbrella (they don't want to get wet). However, the whole family has only one umbrella. So they may need to pass the umbrella back and forth. It takes 1, 2, 5, and 10 minutes for the father, the mother, the child and the grandmother to overpass the bridge respectively, and if two of them walk together, the duration depends on the one who takes longer time.

Draw a Message Sequence Chart (MSC) to capture a scenario that the father is doing all the work (involving with every crossing), then think and answer if this is the most efficient way (fastest). If you think it is, then write a small paragraph to justify why you believe so else if you think there is a better (faster) way, then draw another MSC.

Hint: objects could be East and West, and a message could be e.g. Father(1).

Answer 3:

Answer 3 contd:

Question 4 **15 marks**

Design, in Java, a class hierarchy that includes three kinds of graphical figures: circles, rectangles, and squares as well as a text box which can label the figure or convey a message to the user. Each figure and label has an anchor point whose position determines the position of the entire figure. Each figure and label contains the following functions :

- erase() erases the existing figure or label
- display() displays the figure or label in its position
- reDraw(int x, int y)moves a figure or label on the screen, changing the coordinates of the anchor point by x and y

Assume that implementation of methods display() and erase() already exists.

(a) Write in Java the classes for the figures and label. Include appropriate variables, constructors, and methods in your code. You may omit the method body of the methods display() and erase() i.e. just write display() or erase( ) with appropriate method modifier wherever needed as shown in definition of class Figure below.

You are required to write the implementation for the method reDraw(int x, int y) using the methods display() and erase().

(b) Create a collection that contains a certain number of different type of graphical figures and labels. Write a piece of code, using this collection, that displays all of them on the screen.

If you wish, you may make use of following (partial) code by modifying it appropriately.

```
public class Figure {
private int row;
private int col;
public Figure (int r,int c)
{    row = r; col = c    };
public void display();
public void erase();
public void reDraw(int x, int y) {
// to be coded
}
}
public class Circle {
 public Circle (int r, int c, float rd) {
super(r,c);
radius = rd; }
}
```

```
public class Rectangle {
public Rectangle (int r, int c, float h, float w)
{
super(r,c);
height = h;
width = w; }
}

public class Text {
// to be coded
}

public class Square {
public Square (int r, int c, float s) {
    super (r,c,s,s); }
    }
```

Answer 4 (a)

Answer 4 (b)

Question 5 **25 marks**

Read the following codes and state the result or errors you may find when the code is compiled and executed :

(i)

```
class Parent {
String name = "Parent";
 public void thisFunction() {
   System.out.println( this.name );
   }
}

class Child extends Parent {
String name = "Child";
}

class StaticThis {
public static void main( String args[] ) {
            Parent p = new Child();
            p.thisFunction();
            Child c = new Child();
            c.thisFunction();
  }
}
```

Answer 5(i) :

(ii)
```
class Parent {

    public void choose() {
    me();
    }

    public void me() {
    System.out.println( "Parent" );
    }
}

class Child extends Parent {
public void me() {
            System.out.println( "Child" );
  }
}

class whichOne {
  public static void main( String args[] )  {
            Parent p = new Child();
            p.choose();
            p = new Parent();
            p.choose();
}
}
```

Answer 5(ii)

(iii)

```
class Parent {
  public Parent( ) {
            System.out.println( "In Parent, No Arg" );
  }

  public Parent( String message ) {
            System.out.println( "In Parent" + message );
  }
}

class Child extends Parent {
  public Child( String message, int value ) {
            this( message );
            System.out.println( "In Child" );
  }

  public Child( String message ) {
            super( message );
            System.out.println( "In Child" + message );
  }
}

class Constructors {
  public static void main( String args[] ) {
            System.out.println( "Construct Child" );
            Child c = new Child( "Start from Child", 10 );
  }
}
```

Answer 5(iii)

(iv) For the following code fill up the table that follows the code. Write ‘ok’ if the statement in the main method executes correctly or else write ‘error’ stating a reason . Assume default access modifier to be public.

```
interface Flyer {
public boolean hasWings();
}

interface Mammal {
static boolean hasLungs = true;
public boolean hasLegs();
}

interface Noisy extends Mammal
{
public boolean hasBark();
}

class Dog implements Noisy {
public boolean hasLegs() {
   return true;
   }
 public boolean hasBark() {
    return true;
    }
}
```

```
class Bird implements Mammal, Flyer
{
   public boolean hasLegs(){
      return true;
      }
   public boolean hasWings() {
      return true;
      }
 }

 class Example {
 public static void main( String args[] )
  {
      Bird chirpy = new Bird();
      Flyer airplane = new Bird();
      Noisy kj = new Dog();
      Mammal lion = new Noisy();
      chirpy.hasLegs();
      chirpy.hasWings();
      boolean live = chirpy.hasLungs;
      airplane.hasLegs();
      }
  }
```

Statement	**Answer 5(iv)**
Bird chirpy = new Bird();	
Flyer airplane = new Bird();	
Noisy kj = new Dog();	
Mammal lion = new Noisy();	
chirpy.hasLegs();	
chirpy.hasWings();	
Boolean live = chirpy.hasLungs;	
airplane.hasLegs();	

(v) For the following code fill up the table that follows the code. Write 'ok' if the statement in the main method executes correctly or else write 'error' stating a reason . Assume default access modifier to be public.

```
class Parent { int data; }
class Child extends Parent { String name; }
class Uncle { String rich; }
class Casting
{
public static void main( String args[] )
          {
          Object object;
          Parent parent;
          Child child = new Child();
          Uncle uncle;

          parent = child;
          object = child;
          parent = (Parent) object;
          child = (Child) object;
          uncle = (Uncle) object;
      }
}
```

Statement	**Answer 5(v)**
parent = child;	
object = child;	
parent = (Parent) object;	
child = (Child) object;	
uncle = (Uncle) object;	

Question 6 **15 marks**

Read the following code fragment. It is for a function which attempts to generate a random number between 1 and 100 and which gives seven tries to a user to guess it correctly.

(i) Draw a flowgraph .
(ii) Based on the flowgraph, find the Cyclomatic Complexity and list all the linearly independent paths.

Line numbers are included for easy reference.

```
1.  display(string msg);  // displays message to user
2.  numOfTries = 0;
3.  i = getRandom(1,100);  // gets a random number between 1 and 100
4.  do {
5.    guess = getUserGuess();       // gets guess of the user
6.    numOfTries = numOfTries + 1;
7.    if  (guess < i ) {
8.          guessTooLow();           // displays that user's guess is too low
9.    }
10.  else
11.       if( guess > i)   {
12.            guessTooHigh();       // displays that user's guess is too high
13.       }
14.       else {
15.           guessed = true;
16.       }
17.  } while ( numOfTries < 7  &&  ! guessed );
    //  loop continues as long as number of tries is less than 7 or  guessed is false.

18.  if (numOfTries = 7 ) {
19.             System.Out.println(" user had 7 tries and answer is ", i);
20.  }
21.  else {
22.         System.Out.println(" user won ");  // not the same as exit node
23.  } //   treat this as exit node
```

Answer 6 :

Question 7 **10 marks**

Consider following descriptions. Suggest to introduce a suitable design pattern in each case and state atleast one benefit such a pattern may provide.

(i)

" I have Client objects that needs to interact with Database, Model, and Element objects. A specific Client must first open the Database and get a Model. Then it queries Model to get an Element. Finally it asks Element for information . "

(ii)

" I have a mazeFactory class that can create components of mazes i.e. it builds rooms , walls, doors between rooms. It might be used by a program that builds mazes randomly. "

Answer 7 :

----END OF PAPER----