NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

SEMESTER I, AY2009-10
EXAMINATION FOR

CS2103 - SOFTWARE ENGINEERING

Nov 2009 **Time Allowed: 2 Hours**

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **FOUR (4)** questions and comprises **ELEVEN (11)** 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.

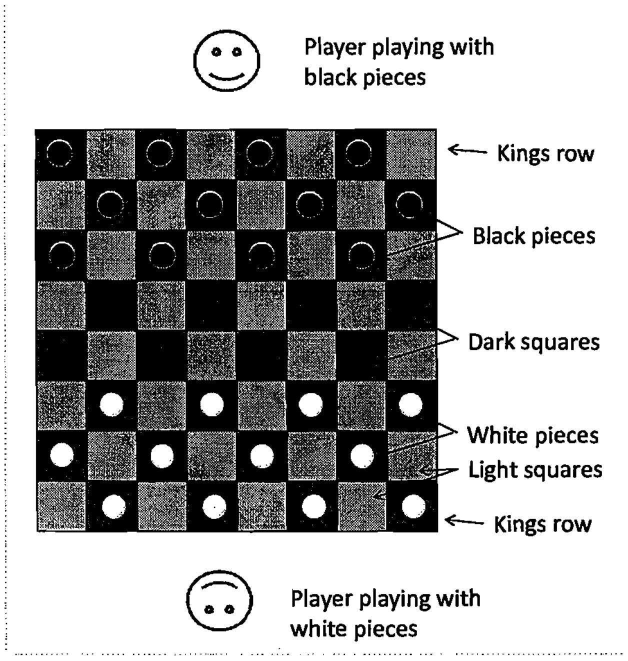
MATRICULATION NO: ____________________

This portion is for examiner's use only

Question	Marks	Remarks
Q1	/13	
Q2	/15	
Q3	/13	
Q4	/14	
Total	/55	

Given below is a description of a simplified Checkers game, adapted from Wikipedia. If any part of this description is not clear, please clarify with the invigilator.

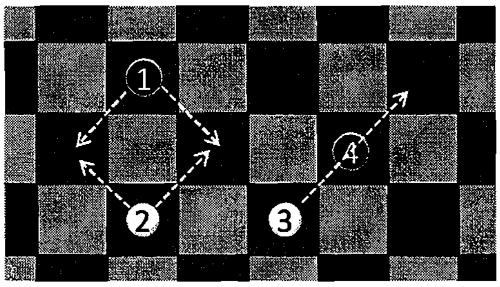
Checkers



Checkers is played by two people, on opposite sides of a playing board. The board is an 8×8 grid, with alternating dark and light squares, called a checkerboard. The pieces are usually made of wood and are flat and cylindrical. A piece can be black or white. One player has black pieces, and the other has white pieces. There are two types of pieces: "normal" pieces and "kings". Each player starts with 12 normal pieces on the dark spaces of the three rows closest to his own side, as shown in the diagram above. The row closest to each player is called the "kings row". The black side moves first.

There are two ways to move a piece:

- A **simple move** involves sliding a piece one space diagonally forwards to an adjacent unoccupied dark square.
- A **jump move** is a move from a square diagonally adjacent to one of the opponent's pieces to an empty square immediately and directly on the opposite side of the opponent's square, thus "jumping directly over" the square containing the opponent's piece. A piece that is jumped is captured and removed from the board.



The diagram above shows some possible “simple moves” for pieces 1 and 2. It also shows how piece 3 can capture piece 4 using a “jump move”.

If a player's piece moves into the kings row on the opposing player's side of the board, that piece becomes a "king" and gains the ability to move both forwards and backwards. A piece is made a king by placing a second piece on top of it.

A player wins by capturing all of the opposing player's pieces, or by leaving the opposing player with no legal moves.

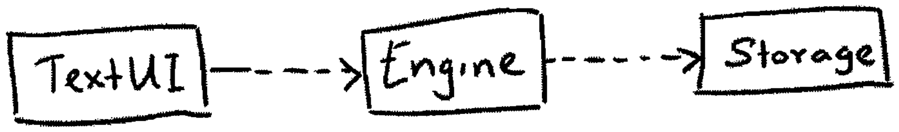
Q1 [13 marks]: Imagine you are planning to implement a computer game called PC-Checkers that is based on the Checkers game explained above. Your first step is to understand and document the problem domain (i.e., the Checkers game explained above).

(a) Draw a domain model that explains objects and relationships in the Checkers game. Show classes, important attributes, associations, multiplicities, association labels, and role names. You may omit the composition symbol.[6 marks]

(b) Explain the high-level game flow in an easy-to-follow way. Your explanation can be diagrammatic, textual or a mix. [5 marks]

(c) Assume you are now brainstorming for potential features for PC-Checkers. One way to categorize such features is as "minimal", "typical" and "unique" as was done in the class project. Propose another way to categorize features (give at least two categories). Give one example for each category [2 marks].

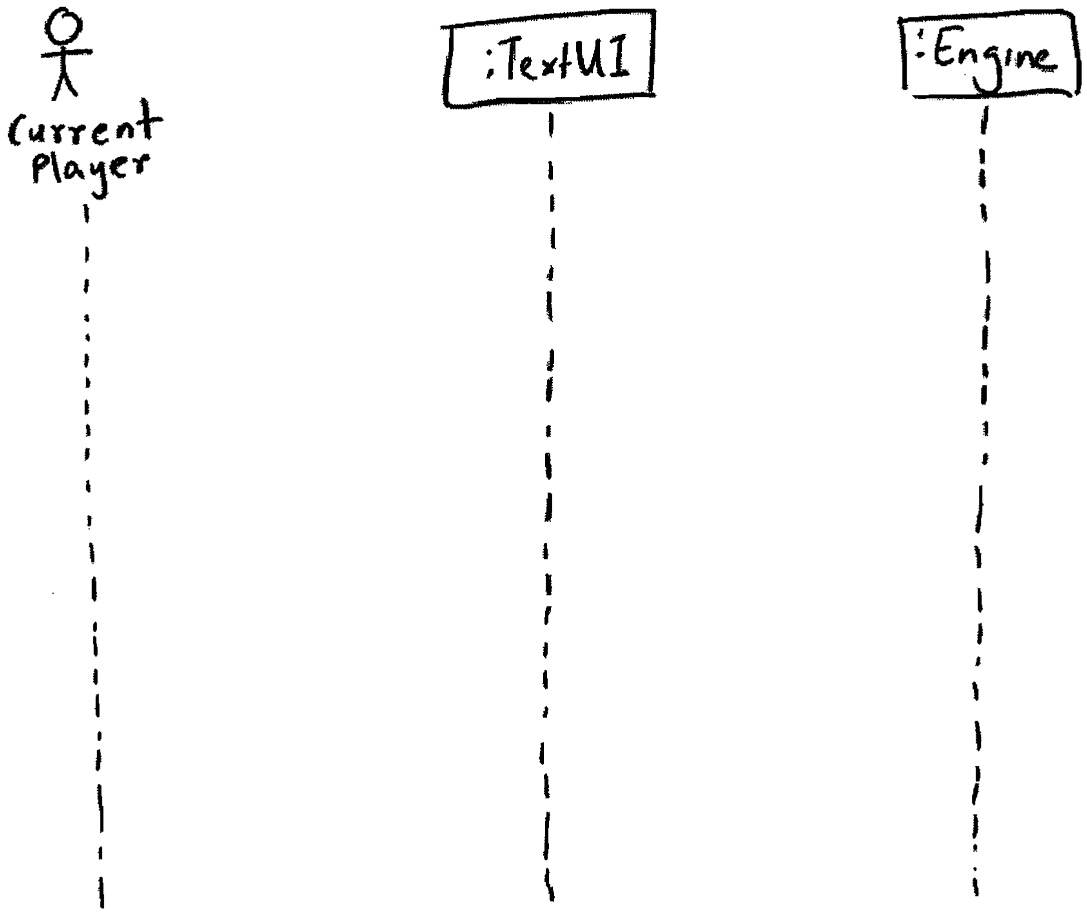
Q2 [15 marks]: Assume PC-Checkers is a text-based game played by two players using the same PC. The two players take turns to make a move. Given below is a possible architecture for such a Checkers game.



Assume that squares of each color are numbered 1 to 32. Pieces of each color are numbered 1 to 12.Here's a partial API for the Engine component.

```
newGame(): void
simpleMove(piece_color, piece_number, destination_square_number): void
jumpMove(piece_color, piece_number, destination_square_number): void
...
```

(a) Complete the following sequence diagram that describes a single turn. Participants are the current player (the player who has the turn), the TextUI, and the Engine component. You sequence diagrams need not follow the full UML notation as long as it is clear enough. [6 marks]



(b) Complete the following operation contract (from the Engine API). [2 marks]

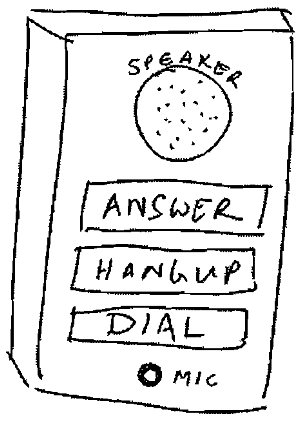
Operation	simpleMove (piece_color, piece_number, destination_square_number):void
Overview	Moves the piece denoted by piece_color and piece_number to the square denoted by destination_square_number.
Preconditions	
Postconditions	

(c) What are the equivalence classes for piece_color, piece_number, and destination_square_number of the simpleMove operation described in (b) above? [5 marks]

piece_color	
piece_number	
destination _square _number	

(d) In addition to piece_color, piece_number, and destination_square_number, name another object or an attribute in the system that should be considered when testing simpleMove operation (b) above. Assume we are using a defensive approach to testing. [2 marks]

Q3 [13 marks] :

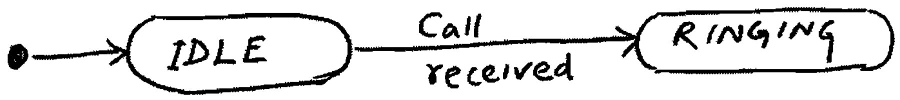


You are designing software for an emergency phone that will be installed in security posts around a high-security zone. It has a mouth piece (microphone), a speaker, and the following three buttons:

- *answer* button – answers an incoming call
- *hangup* button – terminates an ongoing call
- *dial* button– dials head office.

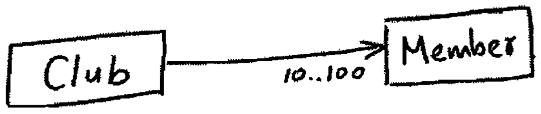
The phone rings when it receives a call. It can receive calls from anywhere. However, it can only make calls to the head office.

(a) Model the behavior of this phone using a state machine diagram. Use the partial state machine diagram given below as your starting point. It should capture information such as how the telephone will respond when the answer button in pressed while the phone is ringing. [6 marks]



(b) Assume the telephone is controlled by a class called Telephone. How would you apply the state pattern when implementing the Telephone class? In your answer, it is enough to show the resulting class diagram containing relevant classes, associations, attributes, and essential operations. [4 marks]

(d) Give a defensive implementation of the remove(Member m) operation of the Club class that removes member m from the Club. You may use Java, C++ or pseudo-code in your answer. [3 marks]



Q4 [14 marks]:

(a) What is the lesson to be learned from the following quote? [2 marks]

About 90 percent of the downtime comes from, at most, 10 percent of the defects.
--Barry Boehm

(b) Argue both against and in support for the following statement. [3 marks]

When to use iterative development? You should use iterative development only on projects that you want to succeed.
--Martin Fowler

(c) Wally is building a software to manage the claim approval process for an insurance company. He is using an OO approach. Explain the relationship between the following three things. [3 marks]

i. How the claim approval process is currently being done – let us call this CAP
ii. The domain model Wally drew during his requirements analysis – let us call this DM
iii. The class diagram of the system Wally built – let us call this CD

[write your answer in the next page]

(d)
Bob: I think we are supposed to draw all UML diagrams *before* we write the code.
Alice: No, we should generate UML diagrams from the code. That way, diagrams will always match the code.
What is your advice to Bob and Alice? [3 marks]

(e) One of your teammates is proposing to use a recently released "cool" UI framework for your class project. Give 4 reasons why this may not be a good idea. [3 marks]

---End of paper---