Aston University

School of Engineering and Applied Science

CS2020 - Software Engineering

Stage Two Examination

CLOSED BOOK

Date: 18th January, 2017
Time: 14:00 – 17:00
Duration: 3 hours

Instructions to Candidates

1. Answer ALL questions from Section A. (40 marks)
2. Section A contains EIGHT questions.
3. Answer THREE questions from Section B. (60 marks)
4. Section B contains FOUR questions, each question is worth 20 marks.
5. Use of calculators IS NOT allowed.

Materials provided

1. Answer booklets

This exam paper cannot be removed from the exam room

Section A — Answer ALL questions

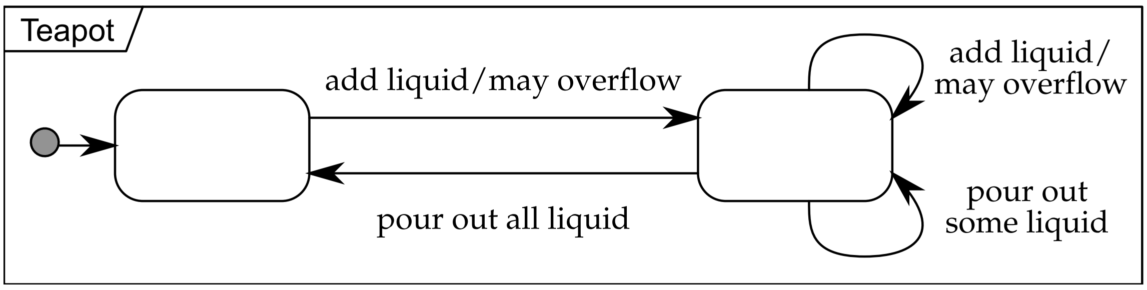
1. a) EXPLAIN the essence of a **cyclic software development process** using a suitable diagram. (3 marks)

   b) STATE ONE significant advantage of **cyclic software development processes** over non-cyclic processes. (2 marks)

2. a) EXPLAIN what kind of information should be shown in a **business process map** AND what kind of information should NOT be shown in it. (3 marks)

   b) STATE THREE typical benefits of creating a **business process map**. (3 marks)

3. Consider the following UML diagram:



   a) STATE the type of the above UML diagram. (1 mark)

   b) SUGGEST helpful names to write inside the two empty boxes. (2 marks)

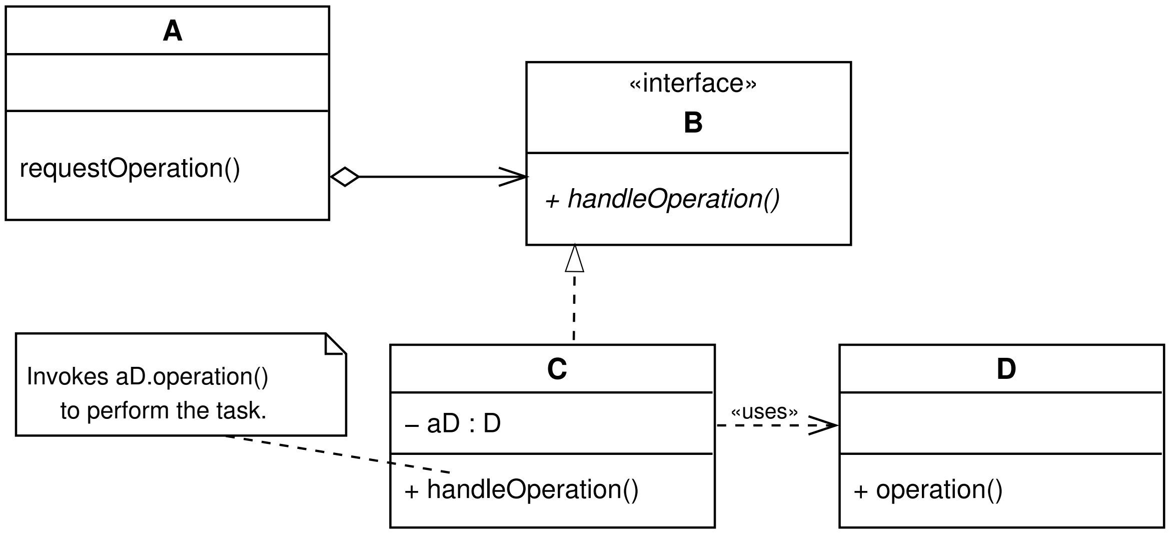
   c) DESCRIBE how to extend the diagram so that it also describes what happens when the teapot is dropped on the floor. (3 marks)

4. REWRITE the following requirement to make it more precise and easier to verify:

   "The user interface should be responsive even on low end PCs."

   (3 marks)

5. Consider the following detailed UML **class diagram** which shows the general form of an object-oriented design pattern as defined by the "Gang of Four". Consider an application that has been designed to work with classes that implement a certain interface. This pattern enables classes which do not implement the same interface to work with the application.



   a) NAME the design pattern described by the given UML **class diagram**.

   (1 mark)

   b) GIVE a suitable name to EACH of the classes `A`, `B`, `C` and `D` shown in the given UML **class diagram**.

   (4 marks)

6. a) Making use of a diagram, describe the core components, their roles, and their relationship within the MVC architecture.

   (5 marks)

   b) NAME the object-oriented design pattern that is typically applied in the implementation of the MVC architecture.

   (1 mark)

7. a) BRIEFLY EXPLAIN the term **User-Centered Design**. (1 mark)

   b) Assume that **User-Centered Design** is to be applied in software development. Which stage(s) of the software development process would likely be influenced by **User-Centered Design**? BRIEFLY EXPLAIN your answer. (3 marks)

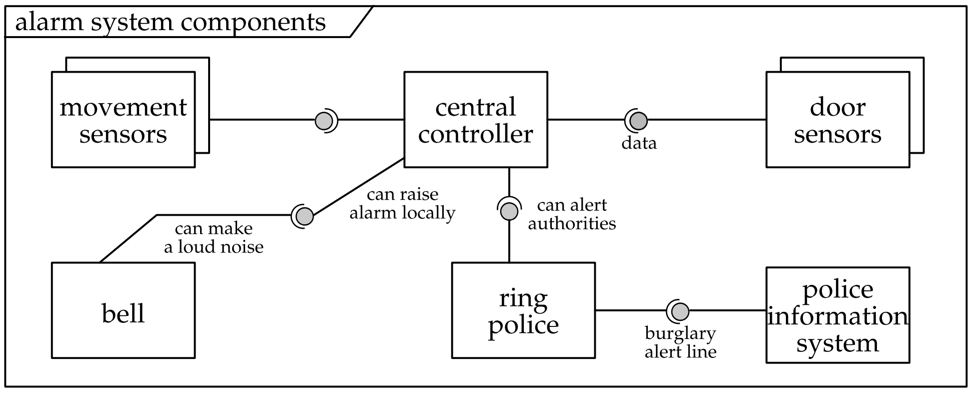
8. a) Making use of a diagram, DESCRIBE the sequence of activities involved in **Test-Driven Development**. (4 marks)

   b) STATE ONE advantage of engaging in **Test-Driven Development**. (1 mark)

END OF SECTION A

Section B — Answer THREE questions

9. Consider the following imperfect **component diagram** showing an imperfect **component model** of an intruder alarm system:



a) The above diagram and the model described by the diagram contain a number of errors. The type and number of some of these errors are specified below. IDENTIFY errors matching the specifications below and describe how to FIX each error:

i) ONE error related to component description (2 marks)

ii) ONE error related to system scope (2 marks)

iii) THREE errors related to component connections (6 marks)

b) Consider the following potential properties or behaviours of the above system:

- “The intruder alarm system can work for up to 3 days without mains power.”
- “The intruder alarm system sounds a bell when a thief enters the property.”
- “The intruder alarm system can be packed in a small box.”

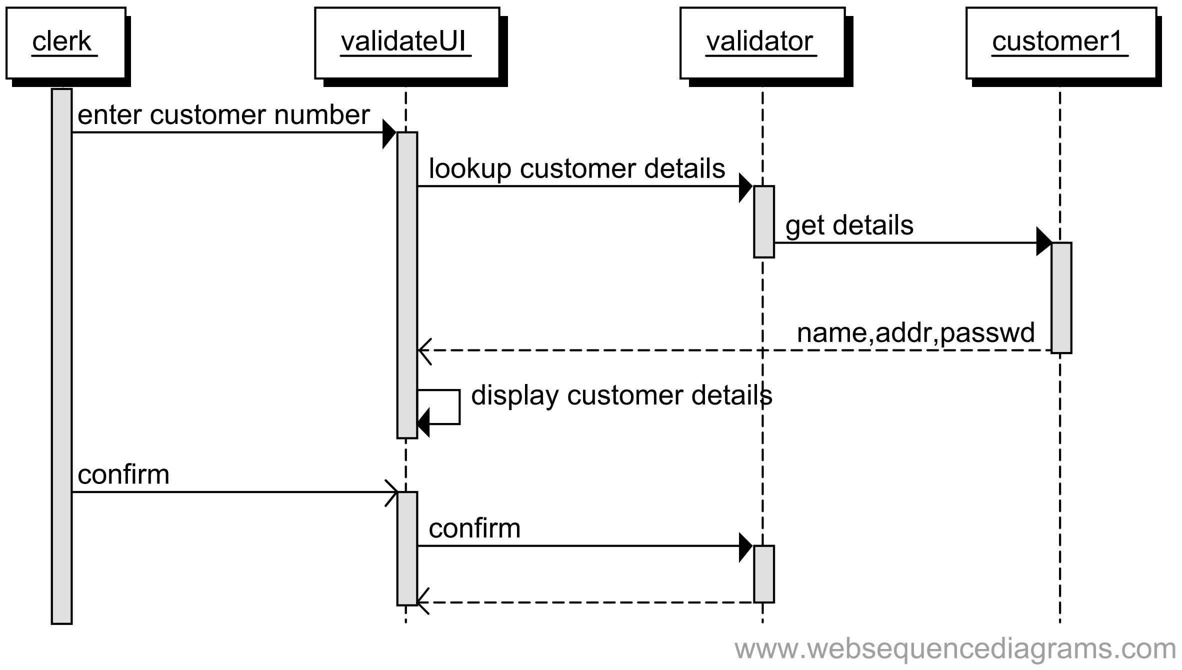
i) INDICATE which of these properties are **emergent** and which are **non-emergent** in the sense used in systems engineering. JUSTIFY your answers.
(3 marks)

ii) DEFINE the concept of **emergent** property/behaviour of a system.

(3 marks)

c) EXPLAIN why **systems engineering** is relevant in software engineering, using examples from the context of developing a software controller for an intruder alarm system.
(4 marks)

10. a) Consider the following **sequence diagram** that contains TWO mistakes that make the diagram invalid:



IDENTIFY both mistakes and for EACH mistake DESCRIBE a way to fix it.

(6 marks)

b) DRAW a **communication diagram** showing the same messages as the corrected sequence diagram from part (a).

Pay particular attention to the numbering of the messages. (6 marks)

(question continues on next page. . .)

(Question 10 continued. . .)

c) Consider the task of developing a software that manages a collection of features at an industrial site. The system should have a graphical interface where a user views the features on a map and can edit them using the mouse.

Assume that you are to perform systems analysis for a use case called "delete feature" with the following (partial) description:

Use case number:5	**Name**: delete feature
Goal: To delete a feature from the site.	
Brief description: The engineer can delete a feature by clicking on the feature, and hitting the delete key or choosing "delete" from the drop-down edit menu	
Actors: site engineer	
Frequency of execution: up to 5 times per minute (when making frequent mistakes)	
Scalability: once at a time	
Criticality: Not too critical as the same can be achieved by pen on a printout and by saving and reloading intermediate versions.	
Other non-functional requirements: none identified	
Preconditions: A site map is loaded into the system and has at least one feature in it.	
Postconditions: The selected feature is no longer associated with the site in the repository.	
Primary path: 1. User clicks on the feature on the backdrop. It is highlighted. 2. User hits the delete key or chooses "delete" from the drop-down edit menu. 3. The system disassociates the item from the site and removes it from the screen.	
Use cases related to primary path: none	

i) DRAW a systems analysis sequence diagram for the primary path of this use case.

(6 marks)

ii) For each object in the diagram INDICATE whether it is a boundary, control or entity object using the standard UML stereotype symbols for this purpose.

(2 marks)

11.a) STATE what is meant by **materialization** within the context of a **persistence framework**.

(1 mark)

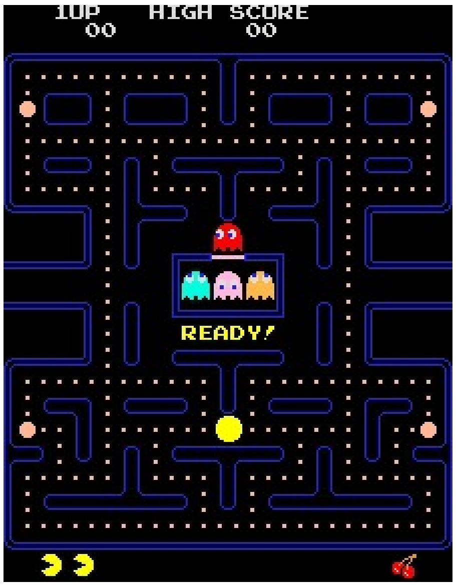
b) NAME the object-oriented design pattern that is often used to support materialization in a **persistence framework**.

(1 mark)

c) Making use of an example, EXPLAIN why the design pattern you stated in part (b) is often used to design a **persistence framework**.

(5 marks)

d) Consider the following partial description of the classic computer game Pac-Man:



The player controls Pac-Man through a maze, eating pac-dots. When all pac-dots are eaten, Pac-Man is taken to the next stage.
At each stage, four ghosts (Blinky, Pinky, Inky and Clyde) roam the maze, trying to catch Pac-Man. If Pac-Man is touched by a ghost, he loses a life. When Pac-Man loses all of his lives, the game ends.
Near the corners of the maze are four larger, flashing dots known as power pellets. As soon as Pac-Man has eaten a power pellet, he enters a temporary state of power-boost. Pac-Man is then able to eat the ghosts. The ghosts, on the other hand, turn deep blue (i.e. in a state of distress), reverse direction of travel and usually move more slowly. Once the time for the power-boost is up, Pac-Man returns to its normal state.

(question continues on next page. . .)

(Question 11 continued. . .)

i) DRAW a detailed UML class diagram to show how the **state** design pattern can be applied in class `PacMan` in a game of Pac-Man.

Your class diagram should include the use of the UML comment notation to outline how the operation(s) that are typical to the ***state*** *design pattern would be implemented.*
Your class diagram may omit details that are irrelevant to the application of the ***state*** *pattern.*

(9 marks)

ii) Making reference to what is meant by **interaction coupling**, evaluate your class design in part (i) in terms of **interaction coupling**.

(4 marks)

12. a) Consider a Java application which contains the following outline Java code with six classes and one interface. This Java application has been developed using an **agile** software development methodology.

Listing 1: Class `Module`

```java
public class Module {
    private static int MAX = 2;

    private String code;
    private Assessment[] assessments;
    private Staff owner;
    private Student[] students;

    public Module(String code) {
        assessments = new Assessment[MAX];
        // other details omitted
    }

    public void assignOwner(Staff owner) {
        // details omitted
    }

    // Other method details omitted
}
```

Listing 2: Class `Assessment`

```java
public abstract class Assessment {
  // other details omitted
}
```

Listing 3: Class `Coursework`

```java
public class Coursework extends Assessment {
    // other details omitted
}
```

Listing 4: Class `CwkSubmission`

```java
public class CwkSubmission {
    private Coursework cwk;
    private Student author;
    private int grade;

    // other details omitted

    public int grade() {
        return grade;
    }
}
```

(question continues on next page. . .)

(Question 12 continued. . .)

Listing 5: Class `Student`

```java
public class Student {
    // other details omitted
}
```

Listing 6: Interface `Tutor`

```java
public interface Tutor {
  void gradeCoursework(Coursework cwk);
}
```

Listing 7: Class `Staff`

```java
public class Staff implements Tutor {
  // other details omitted
}
```

DRAW a detailed UML **class diagram** containing ALL of the above classes and interface, showing ALL attributes, operations and relationships with appropriate multiplicity, defined in the given Java code. (8 marks)

(question continues on next page. . .)

(Question 12 continued. . .)

b) Consider the card game Uno. A deck of Uno cards consists of 108 cards, of which there are twenty-five of each color (red, green, blue, and yellow), each color having two of each rank except zero. The ranks in each color are zero to nine, “Skip”, “Draw Two” and “Reverse” (the last three of these being classified as “action cards”). In addition, the deck contains four each of “Wild” and “Wild Draw Four” cards. Examples of Uno cards are shown below:

In a round of Uno, seven cards are dealt out to each player and the top card of the deck is flipped over and set aside to form the discard pile. The players take turns to:

- play a card from their hand that matches the criteria specified by the top card of the discard pile, or
- draw the top card from the deck.

The first player to get rid of their last card wins the round and scores points for the cards held by the other players.

(question continues on next page. . .)

(Question 12 continued. . .)

Consider the following definition of class `Player`. This class models a player in a game of Uno written at the first **sprint** of an **agile** development process.

Listing 8: Class `Player`

```java
import java.util.*;
public class Player {
  private String name;
  private List<String> hand; // the Uno cards held by this Player

  public Player(String name) {
      // details omitted
  }

  public String playCard(String topCardOnDiscardPile) {
      // detail omitted
  }

  public String getCard(String newCard) {
     hand.add(newCard);
  }

  // other details omitted
}
```

i) Briefly DESCRIBE what is meant by **refactoring**. (2 marks)

ii) It has been decided that the second **sprint** of the development will focus on recording the number of points won by each player.

DESCRIBE ONE **refactoring** that is necessary for the above partial definition of class `Player` so as to prepare it for use in the second **sprint**.

(3 marks)

iii) Making reference to **agile** software development, EXPLAIN why it is beneficial to perform the **refactoring** in part (ii) on class `Player`.

(4 marks)

c) **Enforce Intention** is one technique for Defensive Programming. IDENTIFY where the **Enforce Intention** technique can be applied to the Java code in Listing 8, and briefly EXPLAIN why this technique is used. (3 marks)

END OF EXAMINATION PAPER