Aston University

School of Engineering and Applied Science

CS2020 - Software Engineering

Second Year Examination

CLOSED BOOK

Date: 13th January, 2015
Time: 14:00 – 17:00
Duration: 3 hours

Instructions to Candidates

1. Answer ALL questions from Section A (40 marks)
2. Section A contains 11 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. STATE ONE **emergent property** of a car engine system, and ONE **non-emergent property** of this system. JUSTIFY your answers. (4 marks)

2. STATE the main difference between the goals of the **elaboration** and **construction phases** in the Unified Process. (2 marks)

3. STATE THREE reasons why it is sometimes beneficial to create models of software systems. (3 marks)

4. STATE in what context one prefers vague requirements over precise requirements. (2 marks)

5. a) Briefly EXPLAIN the essence and benefits of **viewpoint-oriented requirements elicitation**. (2 marks)

   b) NAME THREE **viewpoints** relevant to the development of a smart phone educational game. (3 marks)

6. a) EXPLAIN how a **conflict** can arise in a **repository** of a version control system. (2 marks)

   b) Assume that a team of programmers use a version control system to manage their shared source code. DESCRIBE a good practice that the team should adopt to minimise the occurrences and impact of conflicts. (2 marks)

7. COMPARE the characteristics of the **Model-View-Controller** architecture with a simple **Layered** architecture with three layers. DESCRIBE ONE similarity and ONE difference between these architectures. (4 marks)

8. The following Java code for method `createstage` is very difficult to read because it fails to conform to the Java coding conventions.

   STATE FOUR types of issues with the readability of the following Java code.

   (4 marks)

Listing 1: Method `createstage`

```
1   public void createstage(int piles, int cards)    {
2           Stack<Card> temporary = new Stack<Card>();
3           int[] cardT ;
4           int p = (piles*cards);
5           int r = piles-1;
6           if(p % r !=0)
7           { }
8           else{
9                 int g = (p/r);
10                cardT = new int[g];
11                          gdardHolder = new CardHolder();
12              gdardHolder.create(piles);
13  // it creates a number of cards equals to "g" and put in a storage
14                    for (int d= 0;d<g;d++)   {
15                      c. createCards( );
16                      rndNum= rnd.nextInt(51) ;
17                      cardT[d]=rndNum;
18                      temporary.push(c.getSpecCard(rndNum));
19                    }
20                     Set<Integer> numbers = new HashSet<Integer>();
21
22            for(int u = 0;u<piles;u++) // creating number of piles needed
23                      {
24                      c. createCards( );
25                      rndNum= rnd.nextInt(cardT.length) ;
26                      if (numbers.contains(rndNum))  {}
27                      else{
28                      numbers.add(rndNum);
29                      rndNum= rnd.nextInt(cardT.length) ;
30                      temporary.push(c.getSpecCard(cardT[rndNum]));
31                      }
32                              pile = new Stack<Card>();
33                              gdardHolder.addOn(u,pile);
34                      }
35                      while(temporary.isEmpty()==false)
36                      {
37                              for(int t=0;t<gdardHolder.cardHSize();t++)
38                              {
39                                      gdardHolder.put(t, temporary.pop());
40                              }
41                      }
42              }
43            }
```

9. A residential property management company manages various types of residential properties (eg houses, flats) on behalf of their clients. When there is a fault within a property (eg a faulty light switch), the resident reports the fault to a property manager by phone, email or post. Upon receipt of a fault report, the property manager arranges repairs to be carried out as soon as possible. If a fault occurs within a communal area of a property, a property manager may receive multiple reports about the same fault.

   Suppose you have been tasked to design a computer system for residential property management with the following **measurable objective**:

   - **Each property manager will be able to process 30% more fault reports.**

   Briefly DESCRIBE TWO potential ways to achieve the above **measurable objective**.

   (4 marks)

10. a) STATE what is meant by EACH of the following terms within the context of a **persistence framework**:

      i) **materialization** (1 mark)

      ii) **dematerialization** (1 mark)

      iii) **lazy materialization** (1 mark)

   b) NAME the object-oriented design pattern that is suitable for supporting **lazy materialization**.

      (1 mark)

11. a) STATE the purpose of a **test fixture**. (1 mark)

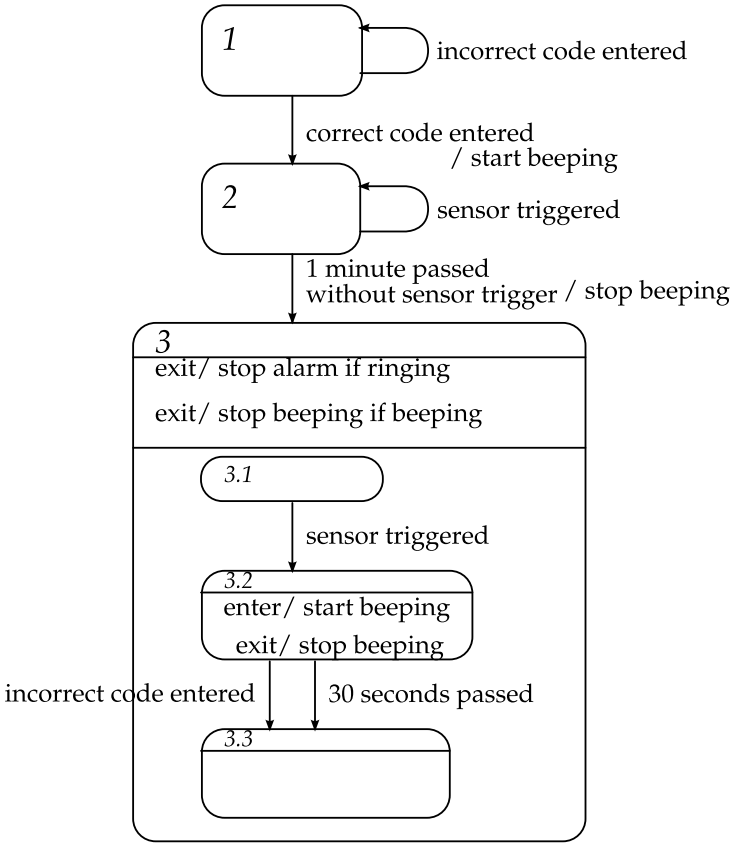
   b) DESCRIBE the THREE steps involved in setting up a **test fixture** in **JUnit**.

      (3 marks)

END OF SECTION A

Section B — Answer THREE questions

12.a) Consider the operation of an intruder alarm system in a house. The system is pre-activated by entering its code. The system becomes active one minute after it stops sensing movement in the house. If movement is detected when the system is active, it beeps for 30 seconds or until the code is entered. If the code is not entered within 30 seconds, it starts ringing loudly until the code is entered. The following **state machine diagram** attempts to describe the behaviour of this system, but it is not complete:



i) DESCRIBE how to complete the diagram above so that it represents the specified behaviour. In particular, describe how to add: initial state indicator(s), an important transition, and an important action. (6 marks)

ii) EXPLAIN why the states in a **state machine diagram** do not have to be named. (2 marks)

(question continues on next page. . .)

(Question 12 continued. . .)

b) Consider the following objects:

- pen
- ink cartridge
- company logo printed on a pen
- paper

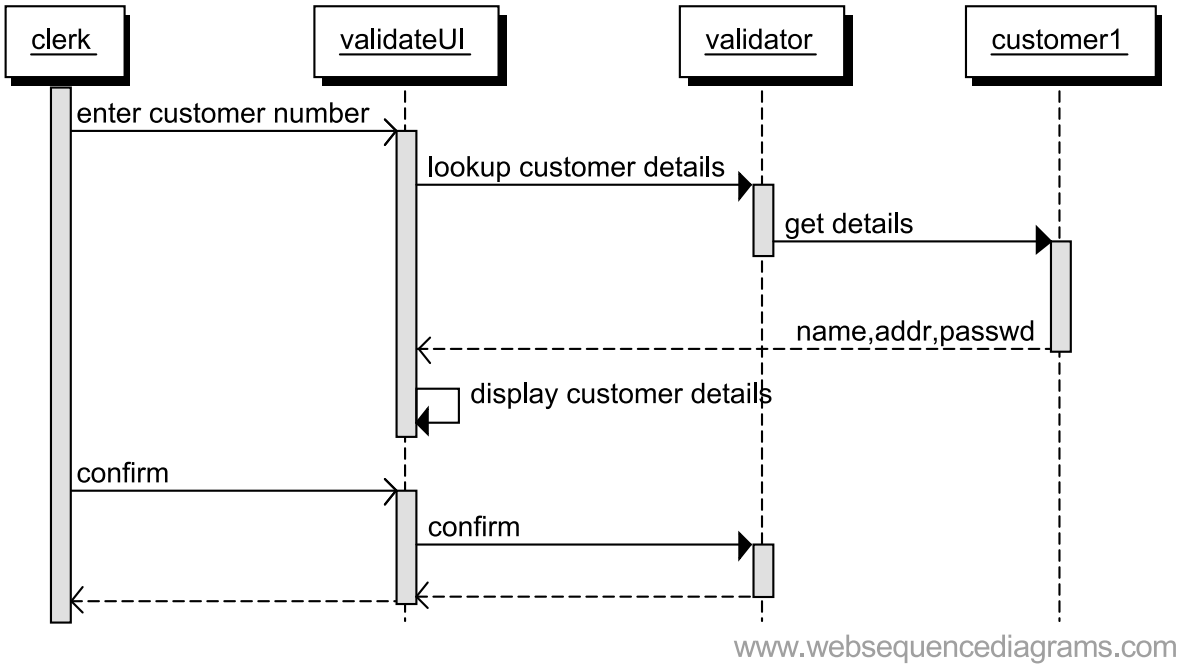
i) Identify among these objects:

- a **composition** (2 marks)
- an **aggregation** that is not a **composition** (2 marks)
- one **association** that is not an **aggregation** (2 marks)

ii) DRAW a **class diagram** showing the THREE relations identified in (i), including their **multiplicities** in both directions. (3 marks)

c) EXPLAIN why **business analysis** is relevant in software engineering, using examples from the context of developing either a software simulator of a car or a software system for assisting domestic cooking. (3 marks)

13.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 13 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 a 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	
Alternatives:	

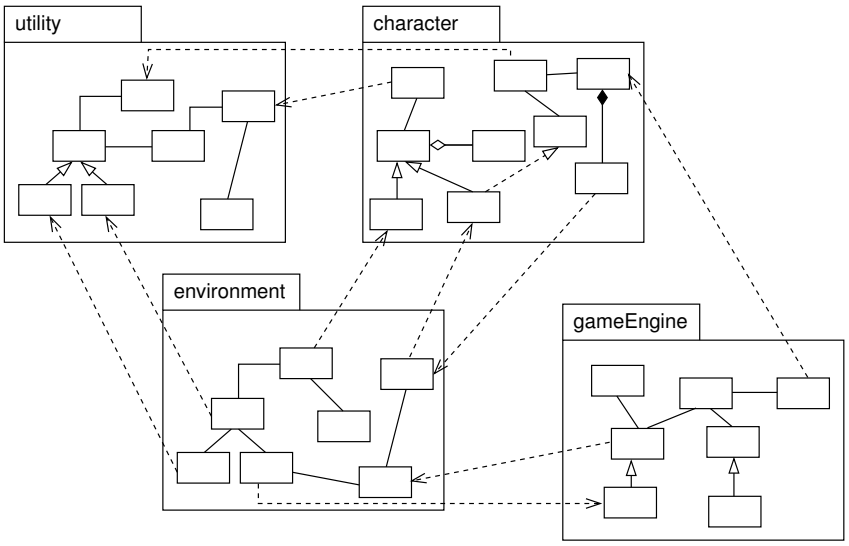
i) DRAW a systems analysis sequence diagram for the primary path of this **use case**.

(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)

14. Consider the following UML **class diagram** which shows the design of a computer game.



a) Which object-oriented design pattern can be applied to the above design in order to loosen the **coupling** between the system components? (1 mark)

b) Making use of a UML **class diagram**, SHOW how the design pattern identified in part (a) can be applied to the above design in order to loosen the **coupling**.

(7 marks)

c) Making use of a detailed UML **class diagram**, DRAW the general form of the **composite** design pattern as defined by the "**Gang of Four**". (5 marks)

d) Making reference to its intended purpose, briefly DESCRIBE TWO applications of the **composite** design pattern. (3 marks)

e) Briefly EXPLAIN what is **inheritance coupling**. (1 mark)

f) EVALUATE the **composite** design pattern in terms of **inheritance coupling**.

(3 marks)

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

Listing 2: Class `Location`

```
1  public class Location {
2    private int x;
3    private int y;
4    private Occupant occupant;
5
6    // other details omitted
7  }
```

Listing 3: Class `Maze`

```
1  public class Maze {
2    private Location[] locations;
3
4    // other details omitted
5  }
```

Listing 4: Interface `Occupant`

```
1  public interface Occupant {
2    void makeVisible();
3    void makeInvisible();
4  }
```

Listing 5: Class `Explorer`

```
1  public class Explorer implements Occupant {
2    private static final int MAX_TREASURE = 15;
3
4    private Stone[] bag;
5    private String name;
6    private int lives;
7
8    public Explorer(String name) {
9      this.name = name;
10     bag = new Stone[MAX_TREASURE];
11     lives = 3;
12   }
13
14   public void move(int direction) { ... }
15   public boolean pickUp(Stone treasure) { ... }
16   public Stone drop() { ... }
17
18   // other details omitted
19 }
```

(question continues on next page. . .)

(Question 15 continued. . .)

Listing 6: Class `Stone`

```
1 public class Stone implements Occupant {
2   // details omitted
3 }
```

Listing 7: Class `Diamond`

```
1 public class Diamond extends Stone {
2   // details omitted
3 }
```

Listing 8: Class `Species`

```
1 public class Species {
2   private String name;
3   private int strength;
4
5   public Species(String name, int strength) { ... }
6   public int strength() { ... }
7 }
```

Listing 9: Class `Snake`

```
1 public class Snake implements Occupant {
2   private String colour;
3   private Species species;
4
5   public void attack(Explorer explorer) { ... }
6
7   // other details omitted
8 }
```

a) 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)

b) Briefly DESCRIBE what is meant by **refactoring**. (1 mark)

(question continues on next page. . .)

(Question 15 continued. . .)

c) Class `Snake` in Listing 9 is the result of a refactoring. Initially, class `Snake` was defined as shown in Listing 10:

Listing 10: Class `Snake` before refactoring

```
1 public class Snake implements Occupant {
2   private String colour;
3   private String species;
4
5   public void attack(Explorer explorer) { ... }
6
7   // other details omitted
8 }
```

NAME the type of **refactoring** that has been applied to the version of class `Snake` in Listing 10. (1 mark)

d) Making reference to **agile** software development, EXPLAIN why it is beneficial to perform the kind of **refactoring** in part (c) on class `Snake`. (3 marks)

e) GIVE a BRIEF definition of **Defensive Programming**. (1 mark)

f) **Enforce Intention** is one technique for Defensive Programming. Briefly DESCRIBE TWO examples of how to enforce intention in Java. (4 marks)

g) The Defensive Programming technique **Enforce Intention** has been applied to the Java code given in this question. IDENTIFY where the **Enforce Intention** technique has been applied and briefly EXPLAIN why this technique is used.

(2 marks)

END OF EXAMINATION PAPER