UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR08019 INFORMATICS 2C - INTRODUCTION TO SOFTWARE ENGINEERING

Wednesday 21st December 2016

14:30 to 15:30

INSTRUCTIONS TO CANDIDATES

Answer QUESTION 1 and ONE other question.

Question 1 is COMPULSORY.

All questions carry equal weight.

CALCULATORS MAY NOT BE USED IN THIS EXAMINATION

Convener: I. Simpson
External Examiner: I. Gent

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. THIS QUESTION IS COMPULSORY

(a) Consider the OO design of a theatre booking system that defines a class for each of the italicised nouns in the following description. The theatre puts on *performances* of shows. For each performance, the system records a show title *string*, a *date* , a start *time*, and the *status* of each *seat* – whether it is booked or not. For each seat, the system tracks its *position* and its *price*. Different seats can have different prices, but let's assume a seat's price is constant across shows and performances. Draw a UML class diagram for this system, showing associations between classes and, as appropriate, presenting each class either as a separate box or as the type of an attribute of another class. Include suitable multiplicities on each association. [*8 marks*]

(b) One of the core practices of Extreme Programming is *collective ownership*.

i. Give two reasons why it is advocated. [*3 marks*]

ii. Name two other Extreme Programming practices that are important for making collective ownership effective. For each, give a one sentence explanation. [*5 marks*]

(c) i. In a JUnit 4 test class, a test method can have the `@Before` annotation. What does this mean? [*2 marks*]

ii. Write a JUnit 4 test class for the following class that includes separate test methods for checking the behaviour of the `update()` method with its `if` condition true and false. Write your tests in such a way that a helpful message will be included in the AssertionError object thrown if a test fails.

```
public class KeepMax {

    private int max;

    public KeepMax(int i) { max = i;}

    public int update(int i) {
        if (i > max) {
            max = i;
        }
        return max;
    }
}
```

There is no need to explicitly give the appropriate import statements. [*7 marks*]

2. ANSWER EITHER THIS QUESTION OR QUESTION 3

(a) Consider gathering requirements for an upgrade to an existing air traffic control system. The upgraded system is to be used by the same air traffic controllers and, of course, safety is paramount. Of the requirements elicitation techniques discussed in lecture, name two that would be particularly useful and in each case briefly explain why. [*6 marks*]

(b) Assume a colleague has written the following Java class for a simple game.

```
public class Game {

    void setup() { ... }

    void makeStep() { ... }

    boolean gameOver() { ... }

    int printScore() { ... }

    void playGame() {
        setup();
        do {
            makeStep();
        } while (!gameOver());
        printScore();
    }
}
```

You recognise that a number of games have a top-level structure just like `playGame()` above, and you would like to set up a game framework that would support both this game and other games you have in mind. What design pattern might be appropriate to use? Following this pattern, explain how you would set up the framework and how your colleague would modify their class to use this framework. [*8 marks*]

(c) What is *regression testing* and why is it useful? [*4 marks*]

(d) A particular make and model of hard drive is specified with a MTTF of 10 years. What does *MTTF* stand for? Say you plan to use the drive for just 5 years. Explain whether this MTTF figure is sufficient to allow you to predict the likelihood of the drive lasting this long. [*4 marks*]

(e) *User familiarity* is one of the principles of user interface design. Using an example, briefly explain what this principle is about. [*3 marks*]

3. ANSWER EITHER THIS QUESTION OR QUESTION 2

(a) Draw a UML sequence diagram for the invocation of the following `addItem` method on some `Item` object. Be sure to show calls of every method and constructor. When a method returns some interesting value, add a reply message with some appropriate annotation to your diagram.

```
public class MultiSet {
    private QuantifiedItem lookup(Item i) {...}
    private void insert(QuantifiedItem qI) {...}

    public void addItem(Item i) {
        QuantifiedItem qI = lookup(i);
        if (qI != null) {
            qI.incrementCount();
        } else {
            insert(new QuantifiedItem(i));
        }
    }
```

[9 marks]

(b) Consider the following simplified Makefile for a C program.

```
OBJS = a.o b.o
CC = gcc

a : $(OBJS)
        $(CC) -o a $(OBJS)

a.o : a.c a.h
        $(CC) -c a.c

b.o : b.c b.h
        $(CC) -c b.c
```

i. Give the general names for the components of a *make* rule, and identify an example of each component from one of the rules in the example Makefile above. *[3 marks]*

ii. Describe how the *make* command decides when to use a rule. *[3 marks]*

iii. Say all the files `a`, `a.o`, `b.o` are up to date and then the file `b.h` is edited. On running the command

```
make a
```

what command or sequence of commands does the Makefile above cause to be invoked, and what is the effect of each of the commands? *[3 marks]*

(c) Why might software be re-engineered? Briefly describe two specific activities that re-engineering might involve. *[7 marks]*