University of Glasgow

Friday, 22 May 2015

9.30am – 11.00am

(1 hour 30 minutes)

DEGREES of MSc in Information Technology and MSc in Software Development

Software Engineering (M)

Answer All Three Questions

This exam paper is worth a total of 60 marks

INSTRUCTIONS TO INVIGILATORS

Please collect all exam question papers and exam answer scripts and retain for school to collect. Candidates must not remove exam question papers.

1. This question concerns *object oriented modelling with UML*.

Scenario: A new restaurant want to develop a new software system for managing their daily business in the restaurant. The restaurant owner lives locally and wants to be involved in designing the system. At this point, he is sure that the system must manage the following information:

Staff: There are three kinds of staff members, managers, waiters, and hosts. All the staff members have a staff ID and a telephone number. Managers are also able to add and remove waiters and hosts from the system.

Tables: The tables in the restaurant each of an ID number and a number of seats. Each table is assigned to a waiter, and waiters can have any number of tables. Each table will have a number of covers (see below) that must not exceed the number of seats. The Table class should have a method to check if it is a valid seating.

Covers: Each customer that comes into the restaurant should be represented as a cover. Each cover should have a cover ID, a table number, and a food order.

a) Give a UML class diagram that represents the proposed system. Show the relationships described and include all attributes, methods, and multiplicities.

[10]

b) The relationship between tables and covers is described in the scenario above. What kind of relationship is this and how would you represent this in your Java implementation of this system?

[2]

c) For this part of the question, you will need to write JUnit code to test the relationships between tables and covers. Write a set of test cases that would ensure this requirement:

- A table cannot have more covers than the number of seats

[4]

d) The team developing this system will use agile development methods. Describe two potential advantages or disadvantages of using this approach in this context.

[4]

2. This question is about *object oriented design principles.*

a) A software engineer should always seek to reduce coupling in software designs. Describe **two** kinds of coupling, giving a description of the coupling, how you would identify it in UML, and how you would reduce it in implementation.

[10]

b) What is meant by the design principle "keep the level of abstraction as high as possible", and how is this used in object oriented software engineering?

[2]

c) Describe two techniques you could use to design software for testability.

[4]

d) Consider the code excerpt below for a system that logs user interaction using a BufferedWriter object called 'log'. What kind of coupling can you identify in this code and how would you refactor this code to reduce this coupling?

```
//  When an employee logs into the system, make a log of their Employee I
D number and the time.
log.write(System.currentTimeMillis().toString());
log.write ("Employee Login: ");
log.write(Employee.getID().toString());
log.writeNewine();
log.flush()

//  When an employee logs out the system, make a log of their Employee I
D number and the time.
log.write(System.currentTimeMillis().toString());
log.write ("Employee Logout: ");
log.write(Employee.getID().toString());
log.writeNewine();
log.flush()

//  When an employee changes their password, make a log of their Employee
ID number and the time.
log.write(System.currentTimeMillis().toString());
log.write ("Employee Password Change: ");
log.write(Employee.getID().toString());
log.writeNewine();
log.flush()
```

[4]

3. This question is about *design patterns.*

a) Describe the Abstraction-Occurrence pattern. Provide:

- a description of the pattern context
- what problem this pattern solves
- how to pattern solution is implemented
- the graphical representation of this pattern
- an example of an anti-pattern for Abstraction-Occurrence.

[10]

Scenario: The University is creating a new system for managing student enrolment. In this system, the University is represented as an object that should only ever have one instantiation. The University should be represented as a class called University.

b) Which design pattern would you use to represent this relationship?

[2]

c) Provide source code in Java for how you would implement the University class.

[8]