Two hours

Question ONE is COMPULSORY

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Software Design using Patterns

Date:    Friday 24th January 2014
Time:    09:45 - 11:45

Please answer Question 1 and two other Questions

This is a CLOSED book examination

The use of electronic calculators is NOT permitted
Question 1

This question is COMPULSORY

a). “Design patterns are a tool for communication”. Explain this statement, using an example. Your explanation should state who is doing the communication and what is being communicated.

(4 marks)

b). In all organisations where the exam software is used, there is a requirement to store marked exam scripts for future reference. Suppose that this facility already exists, in that scripts can be stored in the format required by the University of Mancunia (UoM). However, the Royal Society of Double Accountants (RSDA) uses a different format, and the University of Lochaber (UoL) uses yet another, for example. Code exists to store the data in each of these formats, but the interface of that code is different in each case, and incompatible with the project’s existing code base.

Draw a UML diagram to show how the Adapter pattern can be used to solve this problem. You should assume that the UoM storage code has a single public operation storeScripts() and that the storage code for each organisation can be represented as a single class which you should show on the diagram.

(5 marks)

c). The Façade pattern has some similarities to Adapter. Explain what the similarities and differences are between the two patterns, and briefly discuss whether Façade would be a sensible alternative to Adapter for the problem of part b).

(5 marks)

d). Somewhere, there needs to be code that decides which adapter to create, depending on the organisation we are storing scripts for. Where would it be sensible to put this code, and how might it make that decision?

(2 marks)

e). What are the consequences of using the Adapter pattern for this task in terms of cohesion and coupling?

(4 marks)
Question 2

a). One key idea in the Agile UP is that all the UP artefacts that are not part of the final product are optional. In what other ways is the Agile UP agile? (4 marks)

b). A slogan for agile software development is “embrace change”. One kind of change is requirements change caused by stakeholder feedback. What other kinds of change can happen (to requirements or otherwise) during a substantial software project? (4 marks)

c). You have been hired by the University of Mancunia to implement a university-wide timetabling system, to go on line for the start of the next academic year, in 9 months time. Currently, timetabling is done in various different ways in different schools, and over 200 admin staff and academics are involved. Under the new system everybody will use the same software and the timetable information will be stored in a central database which will form the “single point of truth” for all timetable information in the University. The software will be based on an existing system, which has only been used by two schools so far,

What will be the main risks associated with this development, as perceived by the following groups of stakeholders?

i). Students (1 mark)

ii). IT support staff (1 mark)

iii). University management (1 mark)

iv). Admin staff responsible for timetabling (hint: consider both short-term and long-term risks from their perspective). (3 marks)

d). Of the four groups of stakeholders mentioned above, briefly state for each whether it would be important to have a representative interacting frequently with the development team. (4 marks).

e). One of the principles of agile software development is “Customer collaboration over contract negotiation”, leading to the agile practice of a customer representative working closely with the development team.

Why is the word “customer” not strictly accurate here? Hint: the timetabling scenario above is an example of the problem. (2 marks)
Question 3

a). It’s often said that design patterns are a tool for communication. Explain with examples why the same is true for GRASP principles.  

(4 marks)

b). “Being agile, we Embrace Change. Therefore we should always design to maximise protection against variation”. How accurate is this statement?  

(2 marks)

c). Explain the relationship between the Creator principle and the use of a Factory.  

(2 marks)

d). Give two examples of Pure Fabrications (other than UI classes) which would probably be useful in an aircraft design application. Hint: such an application would need to deal with a large number of parts, many of which would be similar to each other.  

(2 marks)

e). You have a junior developer who has been reading about GRASP, and you are ready to set him a problem, about representing members of staff in the system. Different people use the system for different purposes, for example setters, markers, invigilators and admin staff. He comes back to you with:  
"This looks like an obvious use of Polymorphism - we have an abstract class StaffMember and subclasses Setter, Marker etc."  
Why is this "obvious" use of Polymorphism actually a mistake?  

(2 marks)

f). Draw a UML class diagram which does use Polymorphism to represent this information given in part e), but in a more appropriate way. You should show the classes and the relationships between them, but no details of the internals of the classes.  

(4 marks)

g). State two ways in which your design in part f) protects against variation.  

(2 marks)

h). In practice, the problem is complicated by the fact that we not only want to restrict what staff can do, but also what they can do it to. For example a setter or marker only needs to set/mark on certain course units, while an invigilator may be able to invigilate any exam in the University. To solve this problem, we can make use of the fact that organisations like universities have a hierarchical structure. A University has faculties, which have schools which (for the purposes of assessment) have course units. What data structure, making use of Polymorphism, would you add to your design to deal with this extra complexity?  

(2 marks)
Question 4

Note: a significant part of this question requires answers in terms of actual Java code.

a). The Strategy, Visitor, and Template Method patterns address different aspects of the same general problem. Briefly state what that problem is, and the role that each pattern plays.

(5 marks)

b). Show, in terms of actual Java code, how you would use Template Method and Strategy together to deal with the following hypothetical problem.

We are doing research into automatic marking of text answers. We have a number of different algorithms for this, but they all have the same structure. They first have a setup() part, which is always the same. Then they have analyse() and decideMark() parts which are different for each algorithm.

You should show the template method class, called MarkingStrategy (7 marks) and an example concrete strategy class called MarkingStrategy1 (3 marks). You should assume that all methods take an Answer object as a parameter. Setup() and analyse() return a modified Answer object while decideMark() returns an int, as does the main markAnswer() method. You should show the complete code for the Template Method itself. For the other methods required, indicate where they have a concrete implementation by a comment
// Concrete implementation here.

Minor errors in Java syntax will not be penalised, but for full marks you should show correct visibilities as well as what is abstract and what is concrete.

(10 marks)

c). A function of the exam software is to produce reports. However, these reports may be in many different formats, which we can’t anticipate in advance. Outline how the Visitor pattern could be used to scan a data structure containing answers to generate a report.

(3 marks)

d). Is this likely to be an appropriate use of Visitor? Hint: we don’t add new answer types very often.

(2 marks)
Question 5

a). Explain, using an example from the exam software, the problem which Aspect-Oriented Programming is designed to solve.  
(4 marks)

b). Briefly explain what the two main constructs within an aspect in AspectJ are, giving an example of how they could be used to deal with the example problem of part a). Accurate AspectJ syntax is not required provided the meaning is clear.  
(6 marks)

c). State the two main reasons why attempts have been made to improve the implementation of design patterns using aspects, and what AspectJ constructs are required in each case.  
(4 marks)

d). Is the use of AOP likely to improve an implementation of the Adapter pattern? Briefly explain why or why not. 
(3 marks)

e). State three disadvantages of AOP which should be taken into account when deciding whether or not to use it on a project.  
(3 marks)