In general the level of English was poor, spelling and grammar was a problem in most cases. Some answers made no sense at all. For A1, the overall performance for the students was impressive. Most students have demonstrated their knowledge and understanding of the P4eb approach. However, there seems to be confusion between how students understood the terms business interactions and business patterns. Some students have used the two terms interchangeably while others did not. In A2 there was a major disparity in students’ performances. About 60% have performed exceptionally well, while the rest performed poorly. A number of students have failed to answer the question, and gave unnecessary information instead. Most students understood the fundamental assumptions of the P4eb approach and had the at least an idea of the steps in this approach, but very few attempted the last part of the question for associating the steps with their target architectural layers.

A3 Explain why design patterns are a tool for communication. Your answer should state who is doing the communicating and should include two specific examples of design patterns. (4 marks)

Design patterns enable quick communication between experienced developers. They also allow experienced developers to communicate best practice to inexperienced ones. An example of the former is “do you think we should use a Factory to create the Adaptors”? An example of the latter is that you might draw a class diagram to show the Strategy pattern, and giving an example of its use.

A number of people misunderstood the use of the word “communication here. It should be obvious that we're talking about people, rather than objects, communicating because the questions says “…who is doing the communicating…”

A4 Explain how the Visitor design pattern works (4 marks)

Visitor allows us to manipulate an object structure with code outside of that structure. A visitor class has a method visitC for every class to be visited. Each such class has a single extra method accept(visitor) which runs the relevant visitor method. An object structure may have multiple visitors, in which case there is an interface which all the Visitor classes implement and the classes in the object structure still need only a single accept method.

A lot of people were caught out by this and didn’t really know the pattern.

A5 Under what circumstances is it appropriate to use the Visitor pattern? (2 marks)
Visitor inverts the normal OO structure and collects the whole of an algorithm in one class. This is appropriate when the object structure is stable but the algorithms to be applied to it are complex and liable to change. Otherwise it’s best to make each class responsible for its part of the algorithm in the normal OO way.

*I gave marks here for anybody who got the right general idea.*

Section B

_Students seemed to be remembering/imagining actors and use cases instead of analyzing the given scenario. This could be explained in two ways: 1) either by the fact that the exam is not suitable for online assessment or 2) the software tool is not suitable for the design type of exam. In general students' answers seemed to be characterised by at least one of the following attributes: unfocused, unorganized, short and lacking detail. None of the students managed to offer a complete and correct description of their model. None of the students produced any convincing arguments in favour of their solution. The lack of graphical representation is a major issue for this particular exam, as it forces the student to detail the solution using natural language. As a consequence most of the time answers were shallow, inaccurate and difficult to read/understand due to a poor command of written English language.

John adds: the comments about being unfocussed etc. and English language issues also apply to section C._

Section C

Answer ONE question from this section. The questions relate to the design of an online assessment system such as the one you are using.

C1

   a) “The server returns copies of the question paper created by a **Factory** which is a **Singleton**”. Explain this statement and how it relates to the use of design patterns for communication.

   (4 marks)

Instead of creating copies of QuestionPaper objects directly, this task is delegated to a separate factory class, which will create them. [1] This provides a separation of concerns and makes it easy to create exam papers flexibly, for instance we might randomly or adaptively select from a question bank, possible giving different papers to different students. [1] We only even need one instance of this factory class, hence it’s a singleton. [1] Design patterns provide a common language for so for instance here an important design decision is conveyed in a single sentence [1]
Note that the fact that the factory is a singleton does NOT mean all the question papers are the same. If anything the use of a factory suggests the opposed (as above) although there is nothing explicit in the question about the properties of question papers.

b) Show the important elements of this class in Java syntax (4 marks)

```java
public class QuestionPaperFactory {

    private static QuestionPaperFactory instance; [1]
    // Constructor is private so we can only create instances internally
    private QuestionPaperFactory() { … } [1 bonus]

    // Eager instantiation is also acceptable
    public static QuestionPaperFactory getInstance() { [1]
        if (instance == null) instance = new QuestionPaperFactory(); [1]
        return instance;
    }

    // Methods to create question papers. [or similar, 1]
}
```

Note that it’s the Factory which is the Singleton, not the question paper (by definition there must be multiple instances of that - they even reside on different physical machines).

c) Suppose the application is being extended so that it will accept question papers written in a variety of different formats and convert those to the format used by the exam software. Draw a UML class diagrams showing how the Adapter pattern can be used to organise this part of the application. (4 marks)


Note; in practice there will be many variations and this simplistic marking scheme will not always work.

It turns out that I unintentionally made this question too easy, as it’s essentially the example of Adapter I used in the course. As a result I marked it strictly, in particular I only gave full marks for solutions which showed that the adapter classes all have the same interface (derived from the AbstractAdapter class) while the classes being adapted have different interfaces.
d) Given an example (not one given in the course) of another application in which the adapter pattern could play a useful role.

**Marks for any sensible example with the role of the adapter clearly explained. E.g. adapting between different product description formats in a retail application.**

There were a lot of very good answers. I gave full marks only to those answers which showed clearly how the full adapter pattern (with multiple concrete adapters implementing a common interface) would be useful, as opposed to merely a single adapter class.

e) Explain how the Factory and Adapter patterns are related to GRASP principles. For full marks you should include four different GRASP principles in your answer.

(4 marks)

High cohesion, e.g a Factory creates objects and does nothing else.
Low coupling, e.g. Adapter decouples code within the application from external question formats
Both are Pure Fabrications – they are there to improve the design and are not related to domain classes
Adapter makes use of Polymorphism to provide Protected Variations
1 mark for each, or for other valid points

C2

When an exam is taking place, student activity is monitored by a tool, which shows a dynamically updated table of the students taking the exam, with information such as when the most recent backup occurred for each student, and whether any students have attempted to cheat by accessing applications or web pages other than the exam. The monitoring tool works by sending HTTP messages to the server, receiving data in response, so to a firewall it looks just like a web browser. In other words, the tool is polling the server at intervals for batches of information, rather than the server providing each piece of information as soon as it’s available.

a) In the first version of the monitoring tool, there was a 1-1 mapping between exams and invigilator tool instances. It was not possible for one exam to be monitored by many tools, or many exams to be monitored by one tool. Also, we expect in the future to have different types of tools for different users, e.g. invigilators in a room vs. managers overseeing the whole exam process. We might even have different types of servers, e.g. local vs. remote. Explain how the Observer design pattern can be used to remove these restrictions and provide the basis of future developments. (4 marks)

The monitoring tool is the Observer and the server the Observable. Hence the tool registers with the server to receive messages, and the server notifies the tool when an event occurs. This provides flexible communication between servers and tools. In
addition, we can have a MonitoringInterface (an ObserverInterface) which different
monitoring tools implement, and likewise for servers, giving the full Observer
pattern.

It’s also acceptable to consider the exam clients as the observables, even though the
are not directly associated with the monitoring tools as in the normal observer
pattern.

b) Draw a UML diagram to illustrate the Observer pattern as applied to this situation. 6
marks

Diagram should show the above in correct UML notation. Observer/observable 2
marks, register/notify 2 marks, interfaces 2 marks. Max 3 marks if the generic
Observer pattern is shown, rather than specifically related to the application.

Many people lost marks though misusing the notation, and quite a few didn’t really
understand Observer.

c) In what way does the method of communication between the tool and the server
described above require changes to the standard Observer pattern? 2 marks.

Normally, notification is per-event, and immediate (a method call). In this case the
server must store events until the tool asks for them – the opposite way round from
the standard pattern.

d) Suggest another way in which the Observer pattern could be used in the monitoring
tool. 2 marks.

The obvious one is in the user interface – the Java event model. Could also be used
to chain monitoring tools together.

e) Briefly suggest how two other design patterns might be used in the monitoring
process. 2 marks

e.g. Strategy for different monitoring strategies, Proxy for indirect server access,
Composite for dealing with the monitoring data hierarchically in a large
organisation… marks for ay sensible suggestions.

f) Explain how the Observer pattern is related to GRASP principles. For full
marks you should include four different GRASP principles in your answer.

Uses polymorphism. Provides flexibility – PV. Low coupling between observers and
observables. Likewise high cohesion. Etc.

This should have been easy but quite a few people failed to mention 4 GRASP patterns.