COMP61532 exam performance feedback 2013

Questions in plain font, original marking scheme in bold, additional comments in bold italic.

Overall, answers to the exam (both parts) were disappointing. It looked like people had not spent enough time revising.

Question A1.d

Explain why design patterns are a tool for communication. Your answer should state who is doing the communicating and should use the Observer pattern as an example. (4 marks)

Design patterns give developers a common language for communicating design ideas. [1] Once two developers know a common set of patterns they can communicate very efficiently by using the pattern names such as Observer. [1] In the case of Observer the designers know they are discussing a flexible way of associating events (state changes) with code which needs to be informed of these events. [1] +1 for any other relevant point (e.g. experienced designers teaching inexperienced ones).

A few people mis-read the question and talked about communicating classes within the Observer pattern. Apart from the fact that the question doesn’t make a lot of sense interpreted that way, It does say *who*, not *what*. Of those who avoided that mistake, many did not say four distinct things and so did not get full marks.

Question A1.e

State the three operations involved in the Observer pattern and where each of them is implemented. (3 marks)

Subscribe - implemented in the observable [1] (although initiated by the observer)
Publish - implemented in the observable [1]
Act - implemented in the observer [1]
There are numerous synonyms for all these terms, all of which are acceptable.

A number of people missed out Act, otherwise answers to this were mostly ok.

Question A1.f

Briefly explain the idea behind the Command pattern and give a specific example of where it might be used. (3 marks)
Represent user-level commands as objects. [1]. These objects contain the data required to undo a command, so we can have multiple levels of undo/redo. [1]. Third mark for any reasonable specific example, e.g. in an aircraft simulator (unlike an aircraft!) we might want to undo control commands to retry.

*Answers to this were very mixed. Many were muddled and some people didn’t seem to know that pattern at all.*

Section C

Question C1

This was the more popular question, by a factor of about 2:1.

You are providing a Computer Assisted Design (CAD) system for Steelworks Ltd, a civil engineering company which specialises in large steel-framed buildings. Frames are constructed from steel bars by using a combination of nuts and bolts and welding. Larger frames are constructed from smaller frames and so on to complete the framework for a building. Other large structures (e.g. air conditioning systems, lifts) are likewise constructed from smaller ones.

a). Draw a UML class diagram which shows how the **Composite** design pattern could be applied to model structures as described about, showing the example structures mentioned. The diagram should indicate how the weight of a structure would be calculated. (6 marks)
The key point is the dual relationship between CompositeStructure and Structure. For full marks the diagram should look very similar to that shown above (inclusion of the Building class is optional). In particular, since very different kinds of composite structures are mentioned, the CompositeStructure class should be abstract.

For solutions with the right overall shape, deduct marks for:

- Leaving out the classes mentioned in the description, or adding extra ones which aren’t there.
- Abuse of UML notation, in particular the diamond for “contains” and the triangle for inheritance.

A solution which has CompositeStructures containing atomic structures (as opposed to structures) should get max 3. A solution which is the wrong shape altogether should get max 2.

_I don’t think anybody got full marks for this. Most people simply ignored the last part of the description and dealt only with frames as composites. A lot of people also failed to describe how weights should be calculated. Abuse of UML notation was fairly rare though._
b). The company has a database of structural components it has in stock. In addition, it has read-only copies of databases from several suppliers which are regularly updated. A requirement for the CAD system is that, to improve responsiveness, all relevant database information is read into memory at startup time, so the database information needs to be represented as objects. For each database, there is a natural translation from the data format into an object structure. However, each resulting object structure is different from all the others. Also, the code which does the translation for each of the external suppliers is not available - they only provide the data, so we will need to write our own.

i). Suggest two ways in which we could structure the code so solve this problem, one of which makes use of the Adapter pattern and one which does not. Briefly explain what criterion you would use to decide which approach to take. (3 marks)

1. Write code which yields the natural object structure corresponding to each database, and the use adapters to convert these object structures into the one the company uses.
2. Write more complex input code which yields the desired object structures directly.
   It would depend how different the natural object structures were; if they were only slightly different, the Adapter pattern would be overkill and approach 2 would be fine.

   A number of people suggested using Interpreter, which I allowed if well explained, although this would probably be overkill. Many people failed to address the last point adequately.

ii). Draw a design class diagram showing the class structure required for the solution using the Adapter pattern. You should give names to associations where appropriate, but you need to show multiplicities, attributes, or operations. (3 marks).

   Actually there is a mistake in this question. It should of course say “you do not need to show multiplicities…” Fortunately, this appears to be the way everybody interpreted it as one problem the answers did not have was being overly cluttered with extra stuff. Not knowing the Adapter pattern was a common problem though.
A number of variants are possible, although the solution above has the nice property that the company’s own object structure is simply a concrete subclass of the abstract adapter (which has been renamed accordingly). As usual, marks deducted for notation abuse, unnecessary clutter etc.

c). The company regularly orders shipments of structures from suppliers, where a shipment may involve several different container loads. A shipment can be in several different states, e.g. ordered, partially delivered (some containers have arrived but not others). Briefly suggest how two different design patterns could be used to help in the tracking of shipments. (2 marks)

State is indicated because the states are non-trivial. [1] (A variant on) Observer could be used to get notification of state changes. [1]

All sorts of suggestions were made here, some plausible, most not.

d).

ii). Briefly explain the principle behind the Flyweight pattern (2 marks)

We distinguish between the intrinsic state of an object, stored within it, and extrinsic state, stored elsewhere. [1] The idea of the Flyweight pattern is to reduce
the amount of intrinsic state, replacing it by extrinsic state, in order to increase sharing and hence reduce the number of objects needed. [1]

Most descriptions only dealt with the simple case where objects don’t have state which needs to be made extrinsic, which is not really the Flyweight pattern.

iii). Explain how Flyweight could be applied to the scenario described above, in a way that minimises the amount of storage required to represent small structural components both when they are being shipped and when they are parts of larger structures.. (2 marks)

We need to find places to put extrinsic state information. When they are being shipped, that state is where they are in the world, and the state would be associated with the container objects, with just a count of each kind of structural component.. [1] When they are part of larger structures that state is where they are in the overall structure which is determined by where they are in the composite tree, and leaf instances can be generated from a factory and shared. [1].

Most descriptions were much more vague and general than this. Few addressed the question of where to put the extrinsic state, which is what you have to do any time you apply Flyweight.

iv). How is the Flyweight pattern related to GRASP principles? (2 marks)

Flyweight itself isn’t. It is an optimisation which is actually likely to increase coupling and generally complicate the design so actually goes against GRASP principle [1] It does rely on a factory which is a Pure Fabrication which increases cohesion and reduces coupling etc. [1] (Any sensible variant on the second point is ok, but the first point must be there.

Sadly, quite a number of people said Flyweight improves cohesion etc.

Question C2

The ABC exam software has a tool which allows students to draw diagrams which consist of boxes joined by connectors. Since diagrams have a graph structure, we refer to them within the project as graphs. We can also take a complete graph and enclose it within a box, which is called a graph box. Naturally we use the Composite design pattern to represent these graph structures.

a) The Strategy, Visitor and Template Method patterns address different aspects of the same general problem. Briefly explain what that problem is, giving an example not shown on the course or on this exam paper (3 marks)
The problem of implementing complex algorithms in OOD [1]. By default an algorithm will be distributed over a number of classes defining an object structure, which can make it hard to understand. [1] Example [1]

Answers to this were frequently muddled. Apart from the example this is bookwork.

b) Since diagrams have a graph structure, we can compare them by graph matching which potentially gives us the ability to do some semi-automatic marking. This turns out to be very complex. No single graph matching algorithm will do the job, we need several algorithms, each of which may be complex.

Briefly explain how the Strategy design pattern can be used to help manage the complexity involved. (3 marks)

Each algorithm is encapsulated within a class. [1] We have an abstract superclass and a subclass for each algorithm (strategy). [1] This gives us a simple, uniform interface for our algorithms and gives us a convenient way to switch between algorithms at runtime. [1]

Most answers were along the right lines, although many failed to explicitly state the first point above.

c) Explain how the Visitor pattern could be used to help solve the graph matching problem. Your answer should state whether the situation described is one where use of Visitor could be considered appropriate. (4 marks)

First, yes it is appropriate, because the object structure (boxes, connectors etc.) is simple and stable, whereas the algorithms are complex and variable. [1]. We would define an abstract class, e.g. GraphItemVisitor with subclasses for each algorithm. [1] Within each of these classes there would be a visit method for each kind of graph items, hence collecting all the code for each algorithm in one class. [1]. Each graph item class, e.g. Box, need only implement a single method accept(GraphItemVisitor v) which calls back to the corresponding method in the visitor class. [1]

An acceptable variant on the first point is “we don’t know because we don’t know how stable the object structure is”. There were some good answers to this, but others were muddled.

d) Explain how the Template Method class could also be used to help structure the graph matching code. (4 marks)

Template Method is a way of structuring code in any class hierarchy, so could be used along with either Strategy or Visitor. [1] The idea is that the methods which make up the public interface of the superclass are templates. [1] They provide common code (e.g., to match a box we might always want to get the text inside it)
and then make use of abstract, protected methods to do the subclass-specific work (in this case the actual matching. [1]. The result is that programmers of the subclasses know exactly what they need to do - i.e. fill in the “slots” in the template and the compiler will even tell them this. [1].

Note: answers are likely to be more generic than this, which is acceptable so long as they are in some way related to the particular scenario.

Answers to this were remarkably poor considering TM is a very simple pattern. Few people got full marks; some people didn’t even attempt this part.

e). If we use the Strategy pattern, we may only need a single instance of each strategy. Show, in Java code, how this can be achieved, assuming that if we don’t need a particular strategy it is never created. (4 marks)

This is the lazy initialisation of the Singleton pattern. What they need to figure out is that this has to be for a particular concrete strategy, e.g.:

```java
public class StrategyX extends Strategy {
    private static StrategyX instance;

    // Constructor private

    public synchronized StrategyX getInstance() {
        if (instance == null) instance = new StrategyX;
        return instance;
    }

    // Code to implement the strategy
}
```

3 marks if they show the singleton pattern correctly without noticing the problem, as they won’t be able to answer part f.

A number of people had a Factory which was a Singleton, but the question says it’s the Strategies which need to be singletons. Only one or two people realised that here we have Singleton combined with inheritance, which is tricky to handle. Most people got three marks.

f). What is the problem with the implementation shown in part e, and how can it be avoided?

We need to replicate the singleton code for each concrete strategy because the return types are different each time. [1] We can avoid this by having a StrategyFactory which creates strategy objects on demand and caches them. [1] (There are also more exotic solutions involving AOP).
Anybody who figured this out in part e and wrote the corresponding code gets all 6 marks.

But nobody did. This was tricky, intended to distinguish distinction-level performance from the rest (C1d(ii) had a similar purpose), but it’s disappointing that nobody figured it out.