ISYS1083/1084 Object Oriented Software Design

TuteLab 6 - Structural Patterns (part A)

1. You are required to model a file system where folders and files can be manipulated using common operations and the folders can be repeatedly nested. The **Composite** pattern is a good candidate here, therefore:

a) Draw a class diagram to model an appropriate solution paying special attention to the methods for files in folders and parent folders for a file.

b) As a Java programming and API challenge implement your solution using the smallest subset of methods from `java.io.File` that you can, whereby the majority of the traversal functionality is handled recursively by your own Composite solution. NOTE: If you get stuck for too long on the implementation move on and do it as a self-study exercise.

c) Test your implementation by listing the contents of a deep (more than 3 levels) folder structure to the console.

2. A rich text editor component requires the flexibility of having multiple combinations of features. Some obvious examples include language support (including spelling), markup/styling support (e.g. CSS, XML, HTML etc.), multiple scrolling and wrapping modes etc.

a) Explain how the **Decorator** pattern could be used to facilitate such complexity. Pay attention to why this approach is better than a complex class hierarchy.

b) Draw a class diagram that identifies the main class and relationships that are required here. You don't need to show all features but should show at least two features with two possible variations.

3. Think about how you could use the **Flyweight** pattern in your assignment 1 and what would be the benefit of doing so?

4. The **Proxy** pattern is a clear example of one of the GRASP patterns.

a) Go back to the topic 2 lecture notes and see if you can identify which one it is? If you can't find it then go back to the Structural Patterns lecture notes, read up on proxy and try again! Still can't find it? Ask your tutor but you will need to convince him/her that you have made a reasonable effort!

b) Ok now you should know the purpose of the proxy pattern, so think of some scenarios where it would be useful and is thus likely to be used in real world solutions. HINT: think middleware/system functionality.