# Intro to Design Patterns: after-course work

2 July 2010

## 1

This question is based on Chapter 2 of Vlissides' book Pattern Hatching, which is strongly recommended. (For best learning, you should probably do the question first, and read the book afterwards, if at all! We will discuss Vlissides' approach to the problem next time.)

Design an object oriented API for a hierarchical file system. (That is, feel free to ignore low-level implementation issues, do have classes such as File, Directory...) Your design should permit:

- arbitrarily deep hierarchies of directories
- any directory to be able to report the total amount of disk space used by its contents, assuming some mechanism for each file to do so;
- easy implementation of commands to make a new file or directory, whether in the current directory, by a path relative to the current directory, or by an absolute path;
- symbolic links;
- anything else you think is important or interesting.

Describe your design using whatever combination of UML and Java you like. Be clear about what patterns you used and why.

## 2

Study the pattern relationship diagram. Pick an arrow, preferably one connecting two patterns you are not already familiar with. Preferably using a copy of the GoF book, but failing that Wikipedia, read about the patterns at each end of your arrow. Ensure that you understand the relationship indicated by the arrow. Draw one or more UML diagrams illustrating the relationship, e.g., showing how the two patterns can be used together. Sketch corresponding Java code (you might or might not find it helpful to take this to completion).

If you have the time and the inclination, repeat for another arrow.

You may find the diagram helpful for ongoing familiarisation with patterns: one approach is to tick the patterns off as you read them in detail, and again in a different colour as you use them in real software (this could take a while!)

## 3

Read Martin Fowler's short article *Writing Software Patterns*
`http://martinfowler.com/articles/writingPatterns.html`

Consider whether you have seen recurring problems with recurring solutions in your work. If you can think of at least one, draft a description of one in a pattern form, reflecting as you do so on your choice of form, what is the important information to convey, and whether the resulting pattern might be useful to someone else.