OOP Examples Sheet 3

Dr Robert Harle
Michaelmas 2017

Lecture 11/12: Design Patterns

11.1. **(W)** Explain the difference between the State pattern and the Strategy pattern.

11.2. In lectures the examples for the State pattern used academic rank.

(a) Explain the problems with the first solution of using direct inheritance of Lecturer and Professor from Academic rather than the State pattern.
(b) Explain why a call to getRank() could not simply return the mRank object.
(c) Alternatives are to return a separate rank indicator (e.g. an int); to return a *copy* of the rank object; or return an immutable version of the object. Using an appropriate design pattern, which you should identify, show how to achieve the immutable version, and identify any problems that might occur

11.3. A drawing program has an abstract Shape class. Each Shape object supports a draw() method that draws the relevant shape on the screen (as per the example in lectures). There are a series of concrete subclasses of Shape, including Circle and Rectangle. The drawing program keeps a list of all shapes in a List<Shape> object.

(a) Should draw() be an abstract method?
(b) Write Java code for the method called by the main application to draw all the shapes on each screen refresh.
(c) Show how to use the Composite pattern to allow sets of shapes to be grouped together and treated as a single entity.
(d) Which design pattern would you use if you wanted to extend the program to draw frames around some of the shapes? Show how this would work.

11.4. For some applications the Decorator pattern has the weakness that decorated objects can be decorated. For example, wrapped shop items should not be wrapped again! Show how to prevent a wrapped item from being wrapped. It is easy to solve at runtime using a flag—try to solve it at compile time (so the compiler won't allow it rather than the JVM throwing an exception).

11.5. Write a GUI program using Java Swing that presents a window containing a single JPanel. When you press the right arrow it should change the colour of the panel is a red-blue-green repeating cycle. Pressing the left arrow should cycle backwards through the colours. Use the KeyListener class to detect key presses (and note its use of the Observer pattern).