Labwork 7 Command, Flyweight, Proxy Pattern – Object Orientation with Design Patterns 2015

This lab is worth 6% or 60 points from the total 500 points for labwork this semester (includes 10 marks UML exam preps)

UML DIAGRAM DRAWING EXAM PREPS (10 Points)

Draw a UML diagram for **Labwork 5 Part 2**. This is an example of the Decorator pattern. Include ALL class level details and relationships between participants in the diagram (Note: YOU DO NOT NEED TO HAVE IMPLEMENTED THE SOLUTION FULLY TO DRAW A UML DIAGRAM OF THE SYSTEM)

All pattern participants included\shown	(4 marks)
All relationships shown in correct UML syntax	(4 marks)
Diagram class level details (attributes\behaviours)	(2 marks)

Part 1: Proxy Pattern (15 points)

Create a new project called **Lab7Part1**. Implement a **Proxy** pattern whereby there is a generic class **WebServer** from which the **RealLocalServer** and **ProxyLocalServer** are inherited. Each class will implement a polymorphic method called **getConnection()**. When a call is made to the ProxyLocalServer **getConnection()** method output the message "You are attempting to connect to the web but have connected instead to a Proxy. All website activity will be monitored". When you call the getConnection() method of the RealLocalServer the output should read "You have connected to the real local server directly". Also create a **WebClient** class which will attempt to connect to the RealLocalServer but will in fact connect to the ProxyLocalServer that will forward the connection to the getConnection() method of the RealLocalServer.

- Implement the hierarchy Proxy hierarchy (5 points)
- Implement the polymorphic **getConnection()** method (5 points)
- Call ProxyLocalServer via WebClient and forward to real server (5 points)

Part 2: Flyweight Pattern **(15 points)**

Create a new project called **Lab7Part2**. Create a simple **GremlinGUI** program that applies the **Flyweight** pattern to create multiple Gremlins (minimum 20 shown on GUI in rows and columns with their picture and name: only show a good Gremlin image, all Gremlins will look exactly the same!). Each Gremlin should share the image of the Gremlin (flyweight intrinsic data) but have a unique name shown underneath (flyweight extrinsic data). To add a new Gremlin click the an add button which will launch an input dialog to capture the name of the new Gremlin (remember the image is shared as part of the flyweight, so ensure you do not create multiple images)

Proposed marking scheme:

- GUI to display and add Gremlins (min 20) (5 points)
- Flyweight pattern (intrinsic and extrinsic data) (5 points)
- Test GUI to add new Gremlin instances but share the image (5 points)

Part 3: Command (a basic drawing application which provides undo)

(20 points)

Problem: You have created a basic drawing application that must provide an undo to undo the drawing starting with the most recent

Possible solution: Use the command pattern to store the drawing commands as they are executed thus allowing undo of the drawing from the screen

Create a new project called **Lab7Part3**. Create a JFrame application that will draw shapes three shapes to the screen, e.g., circle, square and triangle. The JFrame has four buttons; one to draw each shape and one to undo the last draw operation. Apply the **Command** pattern to record the drawing of the shapes as commands and **undo** the shape drawings starting with the most recent.

Proposed marking scheme:

- Create drawing application to draw three shapes (3 points)
- Create a super class command (4 points)
- Create the three command subclasses to draw the shapes (9 points)
- Implement the undo operation successfully (4 points)