Task 3 - Extending the ShapeParser

Prepared by William Reid
September 18, 2014

1 Modifying Class Functions

1.1 Shape Class

Create a function called *print()* that returns a string containing the information of the shape. It should be in the same format as it was read in:

type: (X1, Y1), (X2, Y2), ... (Xn, Yn), colour

1.2 DrawPanel Class

Create a function (name of your choosing) that returns the list of shapes that the DrawPanel class has stored locally.

1.3 DrawFrame Class

Modify the constructor to get all the shapes stored in the DrawPanel and send them to the ShapeParser's *writeFile()* method. Be careful where you place this code as incorrectly placing the function calls will cause the program to fail.

Verify this works by adding code to the *writeFile()* method that calls (and prints out the returned string) the *print()* method we made earlier for each shape.

2 Writing to File

Now that we can get the information of each shape, we want to write it out to a file. This can be done using Java's **BufferedWriter** class. The following resource provides a good example of how to implement the required features:

http://www.mkyong.com/java/how-to-write-to-file-in-java-bufferedwriter-example/

2.1 Checking the Output

If you have correctly implemented the write methods, when you run the code you should obtain a new text file (that you may name whatever you wish) that looks identical to the sample file provided to you.

2.1.1 Testing - (2 marks)

If everything has been named and implemented correctly, the DrawPanel should now add Shapes to the drawing surface that have been specified in the text file. The code we implemented in the *paintComponent()* method in **Exercise 3.2** should already draw any shapes in the local data structure (that stores Shapes) onto the drawing surface. If this does not occur, find the source of the errors and report the process in a text file named *debug.txt*

To test your program further, try and pass the *ShapeParser* a file that does not exist and see if it behaves as we expect.

3 Checking your Code (10 marks)

**Note: For each file - 1 mark each goes to comments, layout/structure and conforming to the function/variable naming standards. This does not include Draw.java as this has not been changed. DrawFrame.java and DrawPanel.java will not be marked on layout/structure*

Review your code for layout, structure, comments and how it measures against the supplied standards. When you have completed this check, ensure it is uploaded to the Git repository (you should be committing your work once you have completed each class).

Notify me that you have finished and I will check your work, giving feedback on how things can be improved, including your total mark out of **90**. After I have given feedback you may be asked to modify the code (based on my comments). Once you have completed this and notified me, will move on to the next task. Good luck!

4 Program Understanding

Do you know what we have achieved in these classes? Do you understand how files are read in what we can achieve with exceptions? Create a document called *Questions.txt* that lists questions of things you need clarification on. It may be useful to make this document before you start such that you can add and remove to it as you continue through these tasks.