Task 1 - Drawing Surface

Prepared by William Reid
September 10, 2014

1 Quality and Readability Standards

Read the file *Standards Plan.pdf* located in the *Documents* folder on the repository. Take careful note of the variable and function naming standards - these are important! In addition, you must comment your code as you go. References for every function you need are in the Java API:

http://cs.adelaide.edu.au/docs/java/

Anything in this document in **bold** is a keyword and may refer to an operation or something to search for.

2 Program Driver (2 marks)

Setup a file *Draw.java*. This will be where the program is launched from - At this stage just needs a constructor and a main function.

3 Setting up a JFrame (12 marks)

Create a file called *DrawFrame.java*. This class must **extend** the JFrame class. You will need to define two constructors: one will be the default constructor (no parameters) and one will take two integers (width, height). The constructor for this JFrame must fulfil the following requirements (1 mark for each):

1. Set the default close operation as **exit on close**.
2. Set the frame title to a sensible name of your choosing.
3. Set the size of the frame (default is 800 x 600).
4. Set the background colour to white.
5. Set the **relative location** such that the frame is centred on screen.
6. Ensure the frame is **visible**.

All of the above functions CAN be done in the two constructors, but since both constructors share all but *one* set-up operation, can you think of a way that would involve less duplicated code? (2 marks)

3.1 Adding to the Driver (4 marks)

Test your new class by calling both DrawFrame constructors inside the constructor of your Draw class. Do you see a white coloured interface appear? Does the program terminate when you click the close button?

4 Setting up a JPanel (8 marks)

Create a file called *DrawPanel.java* that **extends** the JPanel class. The constructors for this class will include the default constructor and another that takes two integers (width, height). It is important that when setting up your constructors, the call to **super** sets the **layout manager** to NULL - we will be setting the position of objects on the JPanel ourself. Ensure your JPanel fulfils the following requirements (1 mark each):

1. Set the size of the frame (default is 800 x 600).
2. Set the background colour to white.

4.1 Adding your Panel to the Frame (4 marks)

Now we want to add our newly created DrawPanel to the DrawFrame. In the constructors, add the new DrawPanel to the **content pane** of the DrawFrame. Vary the colour of the panel to check that it is correctly showing up on screen. If the DrawPanel is not appearing you may need to set its **bounds**.

4.2 Adding a line to your JPanel (6 marks)

Now we want to add a method that will draw a *green* line from (50, 50) to (100, 100).

Everything that is viewed by the user must be **painted** onto the screen. Every GUI object inherents various paint methods, of which the most important to us is the **paint-Component(Graphics g)** method. This is where you will add the code for drawing your line - see the following two sites to see how it can be used:

http://stackoverflow.com/questions/5801734/how-to-draw-lines-in-java

http://docs.oracle.com/javase/tutorial/2d/geometry/primitives.html

Notice how the method is **protected** - this is because we do not want classes that are not children (i.e. not GUI objects) to be able to use or call this method. This is also partially due to the fact that all paint related methods are called automatically by the parent object, so there is not need for you to make calls to this new function anywhere in your code.

Ensure your *paintComponent()* method places the line onto the DrawPanel and test it by checking it appears on screen when you run the driver (Draw.java). Prove that it works by changing the length and position of the line.

5 Checking your Code (9 marks)

**Note: For each file - 1 mark each goes to comments, layout/structure and conforming to the function/variable naming standards.*

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 **45**. 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!

6 Program Understanding

Do you know what we have tried to do with **extending** these classes? 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.