AC12001: Week 6 Threads and GUIs

Multithreading and GUIs

Dr Iain Martin 2017

Please note that the topics for this week will not be included in the degree exam and there is no assignment to submit this week. However, learning about multi-threading in Java and GUI programming in general should be useful for your personal development and they may be useful for extending your Team Project application in a few weeks.

Overview:

- Create a multithreaded example
- Modify an existing application to use another thread
- Run Swing GUI examples
- Extend a GUI drawing program

Part 1: Multithreading

Write your own example that uses more than one thread. There is helpful code in the slides to do this.

1. Create a test class

Create an empty Java project. Write a standard Java Test class that creates an instance of Test in the main method and runs a method called `runnableTest`. Add a for loop in the main method to print out a message lots of times. This is to show when the main thread is running. E.g.

```
for (int i=0; i< 30; i++){
      System.out.println("Main thread: " + i);
}
```

2. Create a class that implements the Runnable interface

Create a new class that implements the Runnable interface. (How? Look up Runnable and find out what methods you need to implement!

Add a String field called message and in the run() method. Write a constructor which contains a String parameter that sets the initial value of a String message when an object of this class is created. Write a loop that prints out the message 10 times, like the loop in main method, but uses a field.

3. Test our Runnable thread

Add code to `runnableTest` to use our `RunnableClass` to create and start a thread. We need to do three things:

1. Create an instance of our `RunnableClass`.
2. Create a Thread object, passing our runnable class object to the constructor of the Thread.
3. Start the Thread.

Here's the code to do this. Please look at this code and try to understand it, don't just copy it and run it without thinking about it.

```
RunnableClass runnable = new RunnableClass("Runnable 1");
Thread thread = new Thread(runnable);
thread.start();
```

Run this test and you should see by the messages outputted, the different threads running and interrupting each other.

4. Create a class that extends Thread

There's another way to create a Thread (as there often is with Java!). We can extend Thread (create a subclass (child) of Thread). Create a new class called `MyThread` that extends `Thread`. Add a String message parameter and a constructor to set this message. Add a method called run() that prints out the message 30 times like the runnable class.

5. Run all three Threads

Create a method in your Test class called `CreateMyThread`(), in this method create an instance (object) of your `MyThread` class (with a suitable message) and start the thread. Call this method from main and test it. There should now be three threads running!

6. Thread information

Write a method to output the current number of threads and call this from your main method after creating the three threads. Not that the code to do this is shown in the Thread slides.

Part 2: Swing GUIs

The goal of these exercises is to run and modify some basic GUI examples. If you would prefer to experiment with a different set of Java GUI examples (e.g. from your textbook or an online resource) then that would be fine. If you don't have such a resource in mind, then I recommend working through the exercises below.

1. Run some Swing examples

Download the **PanelsDemo** GUI example from Blackboard and set up a new Eclipse project with it. Play with the code and make modifications to experiment with these examples. For example, change the window size, add more components etc.

2. Set up the drawing project

I have started to code a drawing program (adapted from examples in the book: Programming with Java: a Multimedia approach, Chapter 9), but you can make it better! Download the **GUI_lab_start** example from Blackboard, create a new Java project and add the GUI_Drawing and DrawingPanel classes (from GUI_Lab_start) to it. Run the program and look through the code. See where the JFrame is created, look at the JPanels being created.

See also where the JTextField, JButton and JRadioButtons have been added to the side panel. Look at the action listeners.

Have a look at the DrawingPanel class, this extends (is a child class of) JPanel so that we can Override the Paint method and do our own drawing!

Try to change the Window size. The try to change the background colour. Experiment with different sizes and colours.

3. Load a background image.

Add the following code to the constructor of the DrawingPanel but set the filename to be the filename of an image on your computer from the relative path. If you don't have any images, use one of the images I included in the GUI_Start folder.

```
bimage = new javax.swing.ImageIcon("../images/image.jpg").getImage();
```

Then add the line below to the paintComponent() method. Run the program!

```
g2.drawImage(bimage, (int)image_xpos, 0, null);
```

3. Use a JButton to toggle the image on or off

Let's use the JButton to toggle this image on or off. First add a Boolean field to DrawingPanel, e.g.

```
boolean image_visible = true;
```

use this Boolean to draw the image when it is true and not draw it when it is false.

Then make the JButton toggle this Boolean on or off with each click by writing a mutator method in DrawingPanel to toggle (switch on or off) `image_visible`. Call this from the actionListener for the JButton. Test it!

Change the title of the JButton from "Click Me" to something more appropriate for the action it now does.

4. Move the image.

Add the following code to the actionListener of your JButton.

```
float number = (Float) textField.getValue();
```

```
drawingPanel.setImageXPos(number);
```

Look through the code for `image_xpos` to see how it works. Now let's improve it. Add a second JButton with the title "Move" to move the image. Add its own action listener by copying the current JButton action listener. Then move the code to get the image xpos position from the previous JButton to your new one. Test this to see if it works. You should see you image move in the x direction by the number of pixels in your text field.

Change the label of the JTextfield to something more appropriate.

5. Move the image in the y axis.

Add a second JTextField to the side panel (copy the code for adding the first JTextfield with different names for the variables) and use this to move the image in the y-direction. You will need to create an image_ypos field in DrawingPanel, add a mutator method like I did for image_xpos and add similar code to step 3 to get the value from the text field when the Move button is clicked.

6. Change the image from the File Chooser

Find the section of code that runs when the user selects File Open from the Menu and selects a valid file. Add the following two lines of code then test it by adding a different image.

```
Image image = new javax.swing.ImageIcon(file.getPath()).getImage();
```

```
drawingPanel.loadImage(image);
```

7. Add a button enable and disable the display of the shapes.

Add a second JButton to the side panel and when clicked, it should make the shapes drawn, visible or not visible in a similar way as the image. Note that you will need to create another JButton, another

action listener, another Boolean field in DrawingPanel, another mutator method in DrawingPanel and an if statement in the paint method but you can copy this code from the way this is done for the image.

8. Draw a different object

I have added radio buttons which modify shapeNumber in DrawingPanel. So now you could use this to switch between drawing a rectangle and an ellipse. Look at how the rectangle gets drawn (ask me if you don't understand) and try to complete the work to switch to drawing an ellipse when the ellipse radio button is selected. Look up classes and methods from the Oracle Java class library on the internet to help you.

9. Optional extras

Could you implement a GUI for your Lottery assignment or the RPN Calculator?