The purpose of this lab is to set up a basic GUI. Also, to get you using GIT on your local machine, and you can then practice using github by puting your lab solution up to your GIT HUB repository.

**Part 1 – Check your GIT and GITHUB set up.**

To use Git/ github – you need:

* An installation of GIT on the local machine, to store you local code.
* A working account on github.com (i.e. the cloud) .. as a destination and central pool for your code.

The lab machines have GIT installed already.

**Steps (the various commands and steps are in the lecture notes):**

* You need to configure your local machine – with your username and email – using the SAME ONES as you used to set up your github.com account.. otherwise they won’t be able to talk to each other.
  + Open GIT locally – find the git bash command line $
  + Type in the two “config” commands explained in the notes
* Set up a local GIT repository. See the notes. You can create a special directory if you want (using mkdir.. see notes). Then use the GIT INIT which will do the actual step of declaring a local git repository.
* Then add files (or directories) that will become part of your repository. As you add them.
* When you’re ready to “commit” a snapshot of your files – execute the Commit command on your local machine. Your code is STILL local.. but you have rubber stamped a particular version of it that is now date stamped.

When you have finished, set up a CLOUD repository.. on github.com – get the UR for it – and on your LOCAL Machine – push your local repository to your cloud repository.

**Part 2 – Code up a screen/GUI**

As from class, create a new screen (using the Jframe class).

Set a layout for it (see the following URL to what the types of layouts are for a Jframe) <https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>

Add the following components to it:

2 Buttons, Textfield, Label + 2 other GUI components of your choice.

See a list of all the classes in the Javax.Swing package from the Java API to give you ideas and let you know the class names to use

https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html

Test your screen by instantiating it from a control class.

(Make sure the frame is set to visible AND it may appear tiny up in the corner of the screen, so you might need to maximise it to see it).

Now use the following methods of the Jframe class to control the appearance and behaviour a little more:

setLocation();

setSize().

setDefaultCloseOperation ();

Change the “Layout” being used so that you can see the impact of different layouts: GridLayout, FlowLayout, BorderLayout etc:

<https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>

Finally – add a bit of behaviour so that your GUI *does* something.

Get your screen class to implement a “listener” that will listen out for user action. Implement the ActionListener interface. This interface **ALREADY** exists in the Java api – you do not need to write it. This action listener “listens” for button clicks.

After implementing the ActionListener interface, you’ll immediately get a compile error to make you put in the unimplemented methods of the interface; actionPerformed(..) method. This is called the “event handler”. Inside that method – put the following line of code:

JOptionPane.*showMessageDialog*(**this**, "MyFirst event!");

In your code, register the listener to any button you have created as follows:

Buttonobject.addActionListener(this);

Run your code – click the button – and make sure you getting your message dialog displayed to you. Have a think about what code you put in in order to make the button click work.

**When you have finished – send the code (for this lab only!) to your github repo.**

**Part 3 – Revisit last week**

We discussed in class why it is better to keep the File functionality in its own class.

Edit your lab code from last week so that you keep all file functionality (read , write etc) in a class called FileManager – and use that from the other classes that need it.