Part 1 - XML – No more than 20 minutes on this XML part.

Using a simple editor such as notepad+ or textpad, type up XML that describes a sample "student" that encodes 5 aspects of a student: name, age, course, city of study and institution. Include the following attributes :

- Gender attribute, to qualify the name tag;
- Full or Part-time attribute to qualify the course tag.

The data to be stored is `John Murphy, age 21, goes to DIT Dublin, and he is doing DT228.`

Validate using the online W3C XML validator at:
http://www.w3schools.com/xml/xml_validator.asp

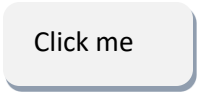
You will sometimes need this validator when programming in Android, as Android Studio does not pick up all XML grammar errors and your project cannot compile in Android if you have any XML grammar errors (e.g. in your XML screen layout).

Part 2 – Android Studio – Event programming – 1st Way: using explicit listener interfaces

Open Android Studio – and Set up a new project. Choose "Empty activity" when you are prompted during the project set up.

Investigate the MainActivity.java shown in the java directory. Make sure you understand what each line of the onCreate() method is doing. For the layout(s), make sure to note the point in blue[1] below

Add a button to the screen



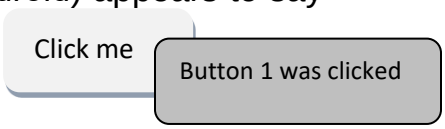
Add a button anywhere in your activity_main XML layout, addition to the textView that is already in there by default when you created the project.

We're not focussing on positioning of views (GUI components) so don't worry too much about where the button is unless you've time.

[1] Note: If you version of Android Studio that creates two XML layout files (activity_main, content_main or similar), the content_main contains the parts of the screen that are in the middle – as an "include" within the activity_main file. For various reasons, it is easier to just take the contents of the content_main (minus the XML declaration) and embed them directly into the activity_main file for now. i.e. just use one XML layout file.

Add functionality to response to button clicks

Implement a listener for the button as an interface within the class using the "implements" keyword. Add functionality so that when the button is clicked it is clicked, a popup message (called a "Toast" in Android) appears to say "Button 1 was clicked".
Use the makeToast() method of the Toast class.



- Run your app (any of emulator/3rd party emulator/ phone) to test it

Add a second button
Add a second button, and get the listener to "listen" to it too.

- Change your code so that when either button is clicked, the Toast message displays a **different message**, depending upon which button was clicked. i.e. "Button 1 was clicked".. "Button 2 was clicked"

Part 3 – Event programming – 2nd Way: using anonymous classes

Change your code (but keep the code from Part 2 as comments so it can be graded) so that instead of declaring a listener interface, anonymous classes are used for both buttons

Part 4 – Event programming – embedded call back method XML

Change your code (but keep the code from Part 2 and 3 as comments so it can be graded) so that you embed the callback method in XML using the onClick attribute.

Part 5 – Event programming – different widget (if you have time, for practice)

Add an editable (EditText) field and a label (TextView) to your layout. Add functionality so that when text is entered, it's value is assigned to the label.