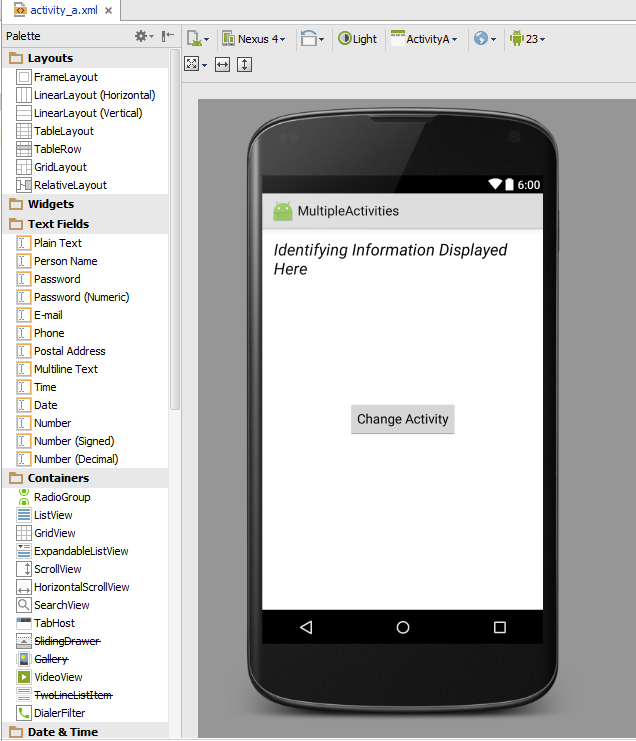
# IN721 2019 3.1 - Multiple Activities

## Task 1: Exploring Intents

Build an app which has three activities, each with screen layout as shown:



This is a RelativeLayout containing a TextView and a Button. If you wish, you can use the same layout.xml file for all three activities, passing it into the setContentView command in each of their onCreate methods.

Use the TextView to provide identifying information for each screen so you know where you are. (**Before you start coding, decide where in the code you will set the contents of that TextView.**)

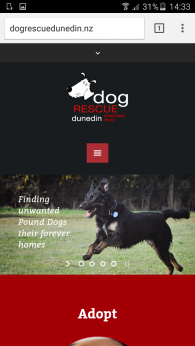
The functionality on button press is:

|  |  |
| --- | --- |
| **When On Screen** | **On Button Press** |
| A | Go to screen B |
| B | Go to screen C |
| C | Go to a website of your choice |

Screenshots of my application when running are shown below. Button press moves between screens.

On Application Launch After Activity A Button Press

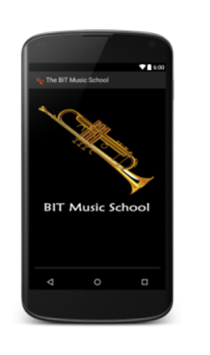
After Activity B Button Press After Activity C Button Press

**Exploration:** What is the effect of clicking the phone's Back Arrow, for the various activities in this application? What does this tell you about the **Activity Stack?**

## Task 2: Adding a Welcome Screen to the Music School app

Extend your Music School app so that upon launch, the user sees an attractive, graphically appropriate welcome screen, which provides clear navigation to the enrolment screen.

**Optional Fun Challenge Task**: Implement your welcome as a *splash screen* that automatically transfers control to the enrolment page after 5 seconds, eliminating the need for direct user navigation (i.e. clicking on the screen or other control). Ignore for now the argument about whether splash screens are a good interface feature. My version:



The most straightforward way to implement this functionality is to add code to the splash Activity's onCreate which uses an Intent to transfer control from the splash Activity to the enrolment Activity. You then briefly delay the execution of this code by pausing the main (UI) processing thread[[1]](#footnote-1). When working in .NET we do this quite directly by calling System.Threading.Thread(msec). In Android it is more complicated. You will need to look at the **Handler** class and the **Runnable** interface.

1. There are more elegant techniques that don't block the UI thread, but let's stick with this direct approach for now. [↑](#footnote-ref-1)