|  |
| --- |
| **Session 1 Worksheet** |

Table of Contents

[Setting up and using Git repository 1](#_Toc38847744)

[Create an Android Project 2](#_Toc38847745)

[Section A: LinearLayout & UI Elements 3](#_Toc38847746)

[Section B: Nested Layouts 9](#_Toc38847747)

[Section C: Using strings.xml to store values 11](#_Toc38847748)

## Setting up and using Git repository

1. Please refer to the "**GitHub Setup Guide.pdf**" for instructions on setting up your own GitHub account.
2. From this lesson onwards, check in your projects into Git.
3. Do provide the URL to your GitHub repository here:

|  |
| --- |
|  |

1. For an Android Project to be deposited into the repository, you need to import into a Git repository. Do refer to the guide to import the project into the repository.
2. During the development, you shall commit the changes into the repository regularly.

## Create an Android Project

1. In Android Studio, create a new project with the following information:-

|  |  |
| --- | --- |
| **Project Template** | Empty Activity |
| **Application Name** | Demo My Second App |
| **Package Name** | sg.edu.rp.c346.id<your student ID>.demomysecondapp |
| **Project Location** | D:\C346\Workspace\P02\DemoMySecondApp |
| **Language** | Java |
| **Minimum API Level** | API 16 |

1. What is the default Package name? How is it associated with the application name and Company domain?

|  |
| --- |
|  |

1. What are some of the rules in coming out with a package name? E.g. can it begin with digit? Can special characters like @, # and \_ be used?

|  |
| --- |
|  |

## Section A: LinearLayout & UI Elements

In Lesson 01, you have learnt how to drag and drop UI elements into the layout screen. Another way is by editing the XML codes.

**Note:** We will be using LinearLayout throughout this module. Please use LinearLayout when answering questions and quizzes.

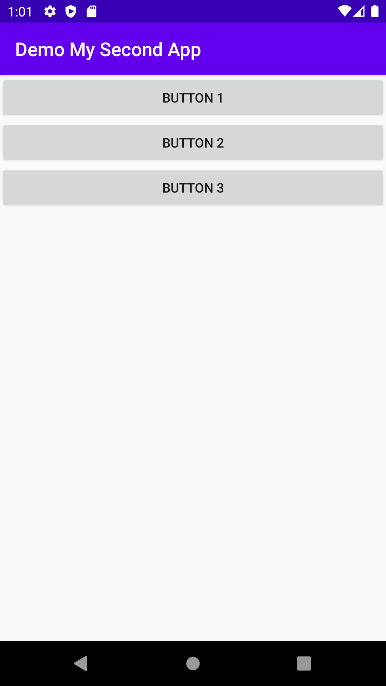
Use the following template for your XML layouts.

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:orientation="vertical"**>   </**LinearLayout**> |

Basic LinearLayout Template

**Exercise 1a**

The buttons are placed in a LinearLayout. What is the orientation for this LinearLayout?



**Answer:**

|  |
| --- |
|  |

**Exercise 1b**

The buttons are placed in a LinearLayout. What is the orientation for this LinearLayout?

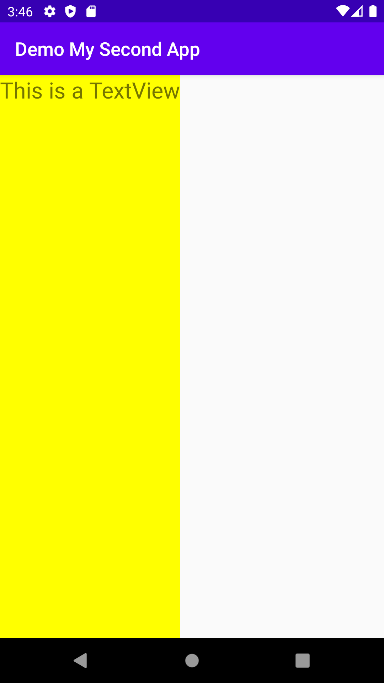


**Answer:**

|  |
| --- |
|  |

**Exercise 2a**

Write the XML codes to display the text as shown. Font size used is 24sp.



**Answer:**

|  |
| --- |
|  |

**Exercise 2b**

Write the XML codes to include the text and button as shown. The default font size is used.



**Answer:**

|  |
| --- |
|  |

**Exercise 2c**

Write the XML codes to include the text and button as shown. Note that the Button takes up the remaining of the width left. The default font size is used.



**Answer:**

|  |
| --- |
|  |

**Exercise 2d**

Write the XML codes to include the text and button as shown. The default font size is used.

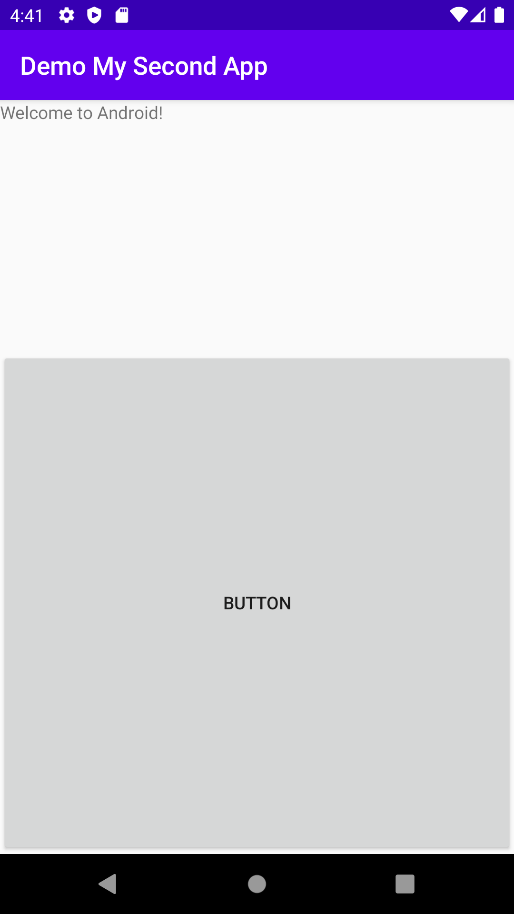


**Answer:**

|  |
| --- |
|  |

**Exercise 3a**

Write the XML codes to display the text and button as shown. The text takes up 1/3 of the screen and the button takes up 2/3 of the screen.



**Answer:**

|  |
| --- |
|  |

**Exercise 3b**

Write the XML codes to display the text and button as shown. The text takes up only the width that it needs. The button fills the remaining space.



**Answer:**

|  |
| --- |
|  |

## Section B: Nested Layouts

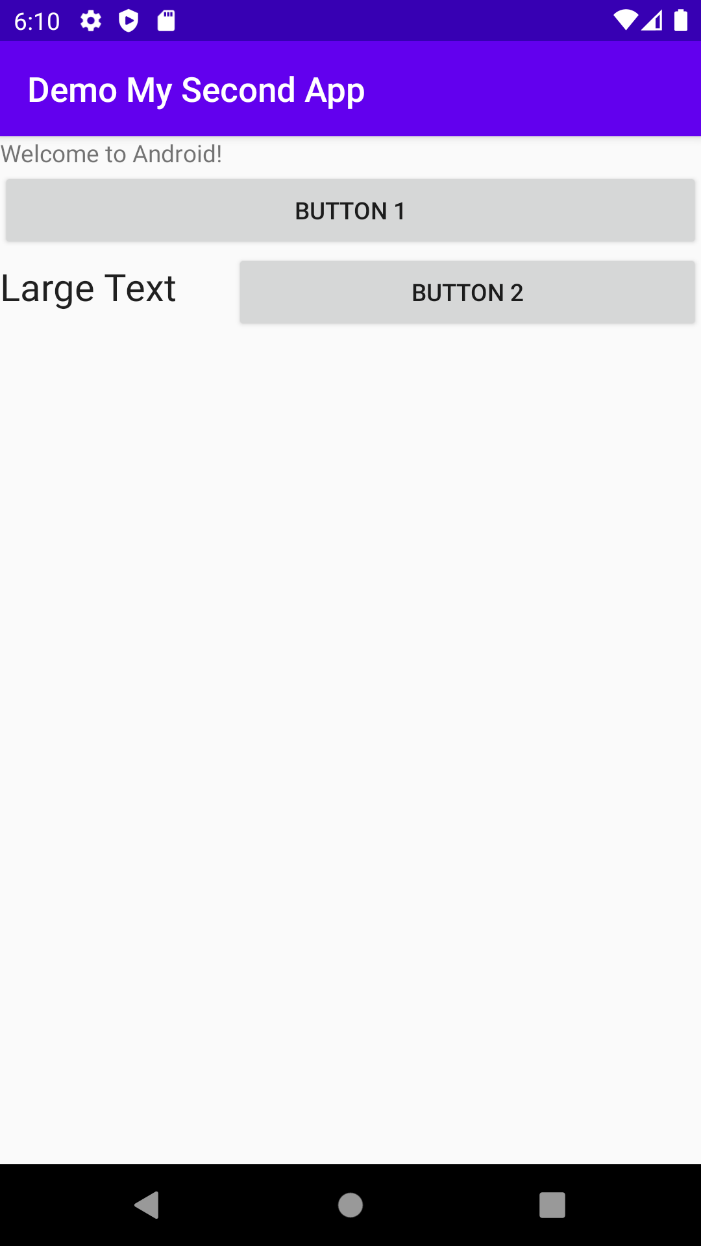
What we have done so far is implementing only one LinearLayout for the App. To solve some of the real life problems, we may need to implement more than 1 layout type or even nested layouts into the App. To support the layout design, Android OS does allow more than one LinearLayout in a canvas.

Given this scenario, how many layout(s) is/are required?

|  |
| --- |
|  |

**Exercise 4a**

Write the XML codes to display the screen as shown. For "**Large Text**" TextView, use "**android:textAppearance**" attribute with value "**@android:style/TextAppearance.Large**".



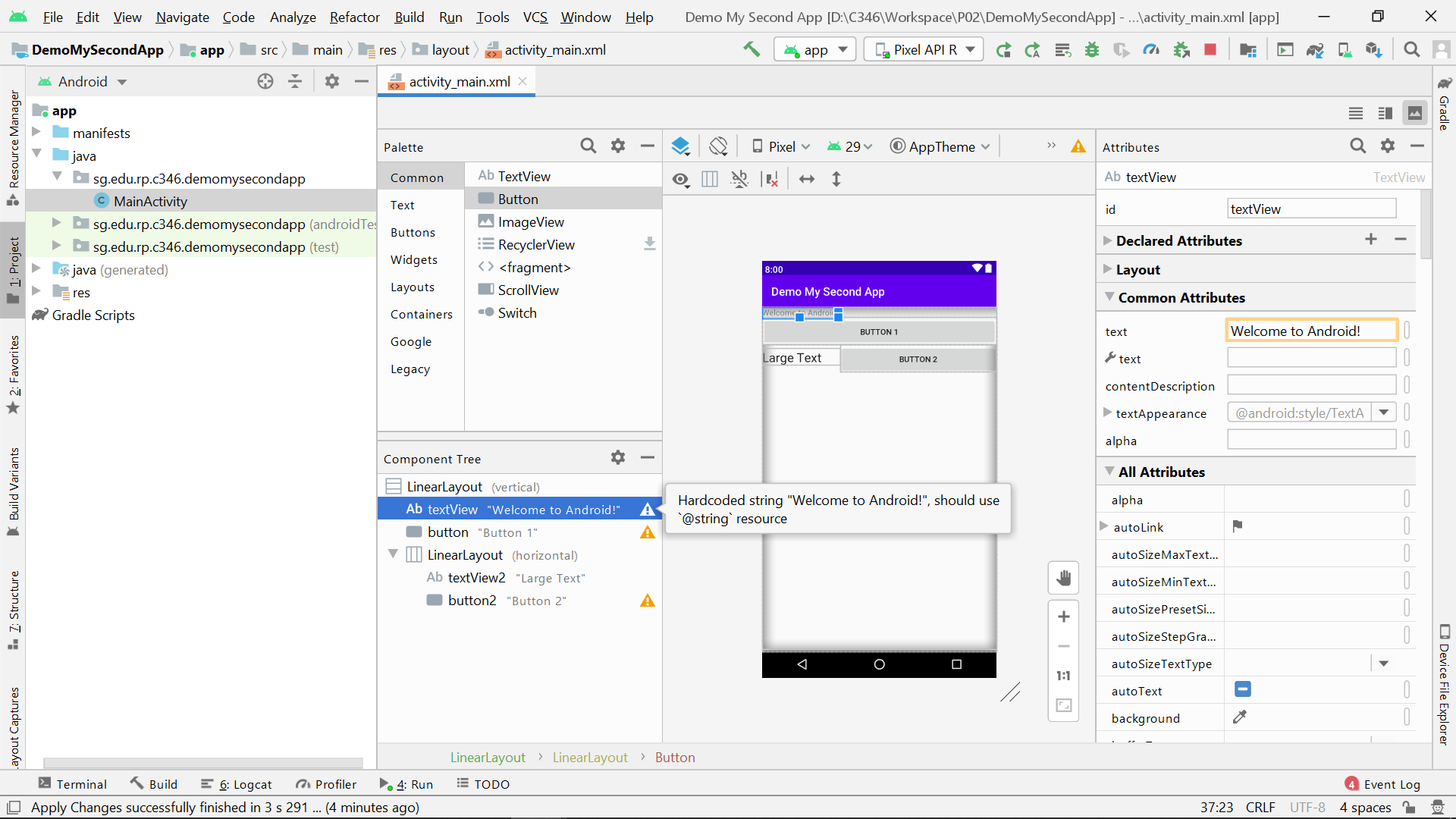
**Answer:**

|  |
| --- |
|  |

|  |  |
| --- | --- |
|  | **Learning Checkpoint** |
| *By now, you should be able to*   * *Use the Android Layout Editor to create a more complex Android UI using nested layouts* * *Add controls into LinearLayout component and make changes to the control element attributes, such as width and height*   **To recap on what we have learned so far,**  A homogenous layout design could be applied for apps with simple UI requirements. To solve some of the real life problems which usually require a more complex UI, we may need to implement more than one layout type or even have nested layouts to either to optimise the screen usage or fundamentally, to improve the usability of the app as a whole. | |

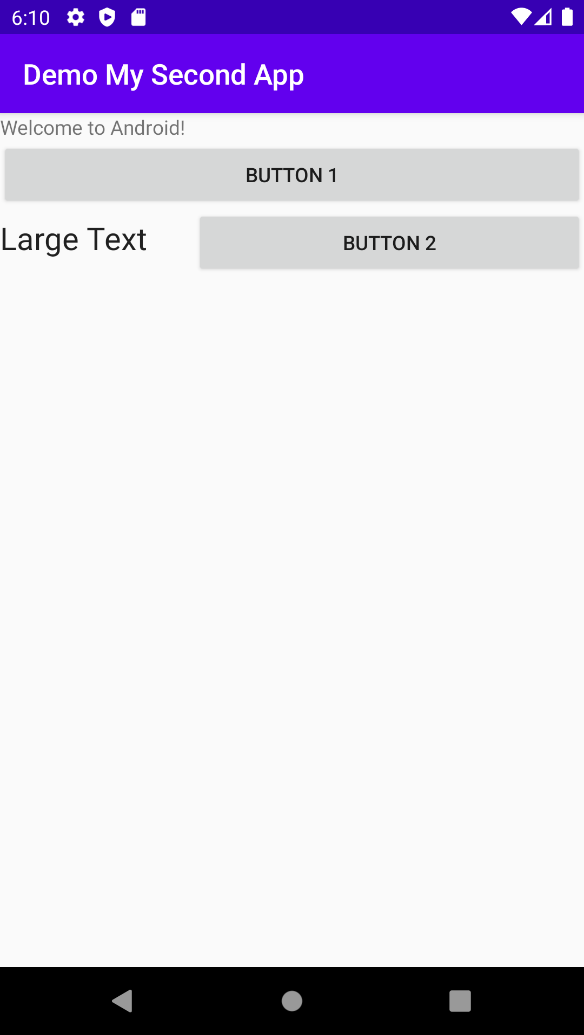
## Section C: Using strings.xml to store values

In this section, we will learn how to change the display label by changing the value of the control’s Attribute in the Design Mode. Furthermore, we will see how to use string resource to manage values better. This is good practice. When the values are hardcoded, Android Studio will give a warning as shown below.



**Exercise 4b**

Remove the hardcoded text messages and replace them using string resources.



**Answer:**

|  |
| --- |
|  |

strings.xml

|  |
| --- |
|  |

activity\_main.xml

|  |  |
| --- | --- |
|  | **Learning Checkpoint** |
| *By now, you should be able to*   * *Define XML element in the string resource file* * *Apply string resource in the layout file*   **To recap on what we have learned so far,**  It is not a good practice to hard code strings in your layout files which does not promote reusability of code. Programmer should add the display strings in the string resource file and reference them from your layout file. | |