Queens University Belfast

Android Studio

CSC3054 / CSC7054

XML Exercises

Queens University Belfast

XML Attributes

A view in Android represents a widget e.g. a button or a layout manager. The Android SDK provides standards views (widgets) e.g. via the `TextView`, `EditText`, `Button` and `ImageView` classes. This tutorial will focus on the standard widgets and their properties and will consider two layouts `RelativeLayout` and `LinearLayout`.

Exercise 1 – Creating an App with two Views

In this app, you will create two Views. An `ImageView` (to hold an image) and a `TextView` (to hold some Text).

Before You Begin

Open `Android Studio` and create a new project called `"XMLExample1"`. Refer to the `'Creating your first project'` tutorial to help you create a project. Once created your project should look like figure 1. Switch from `Design` view to Text `view` .

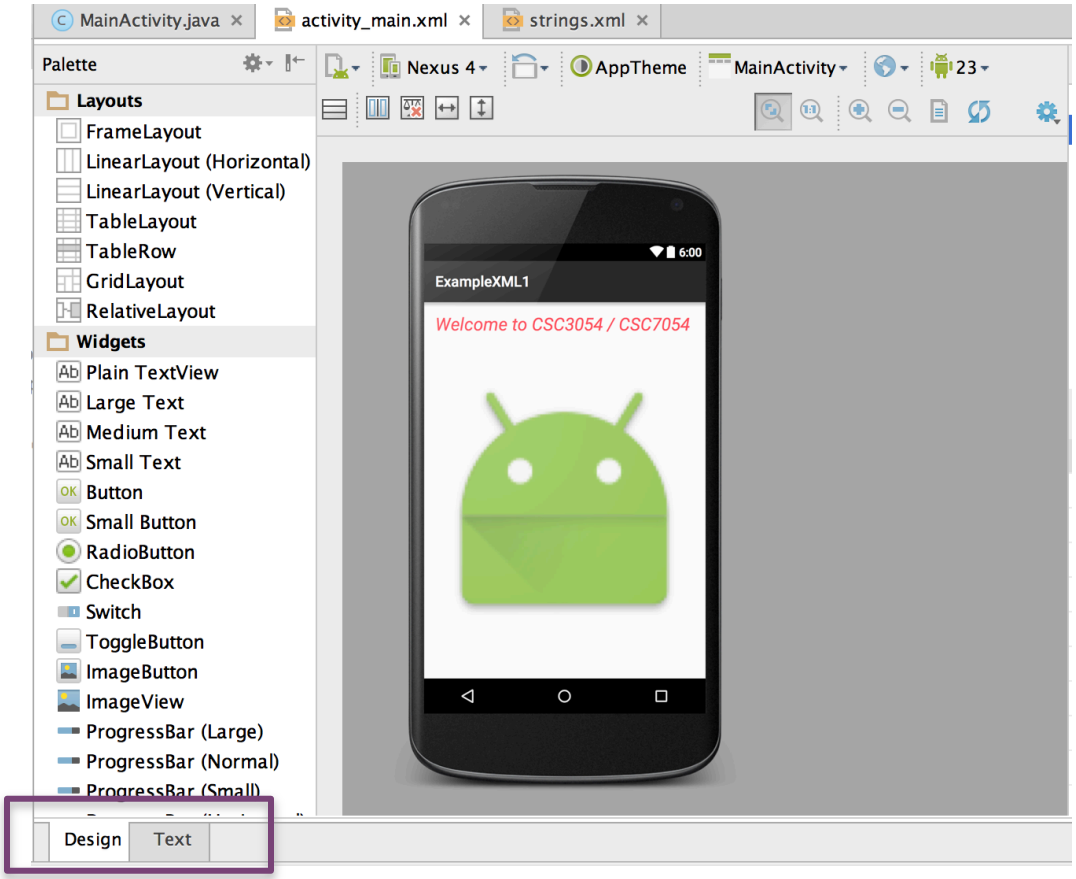

FIGURE 1 - OPEN PROJECT

Step 1: Change the Layout

First change the layout from `RelativeLayout` to `LinearLayout` and set the `orientation` attribute to `vertical`

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity">

</LinearLayout>
```

Notes on `LinearLayout`

This is a view group that aligns all children in a single direction, vertically or horizontally. This layout respects *margin*s between children and the *gravity* (right, center, or left alignment) of each child.

Attribute	Notes
`android:orientation`	• You can specify the layout direction with the attribute. • Children are stacked one after the other. • Vertical list - will only have one child per row, no matter how wide they are • Horizontal list - will only be one row high (the height of the tallest child, plus padding).
`android:layout_weight`	• This layout supports assigning a *weight* to individual children. • This attribute assigns an "**importance**" value to a view in terms of how much space it should occupy on the screen. • A larger weight value allows it to expand to fill any remaining space in the parent view. • Child views can specify a weight value, and then any remaining space in the view group is assigned to children in the proportion of their declared weight. Default weight is zero. •

Using the `android:layout_weight` attribute

If there are 3 text fields

- 2 of them declare a weight of 1
- The other is given no weight

The 3rd text field without weight

- will not grow
- will only occupy the area required by its content.

The other two

- will expand equally
- to fill the space remaining
- after all three fields are measured.

If the third field is then given a weight of 2 (instead of 0)

- It is now declared more important than both the others
- So it gets half the total remaining space
- while the first two share the rest equally

How do I get equally weighted children?

To create a linear layout in which each child uses the same amount of space on the screen

set the

- android:layout_height of each view to "0dp" (for a vertical layout)
- or the android:layout_width of each view to "0dp" (for a horizontal layout).

set the

- android:layout_weight of
- each view to "1".

Step 2: Add a `textView`

Add a `textView` to your XML within the `LinearLayout`. Set the following attributes:

Attribute	Values
`android:text`	`="@string/message"`
`android:layout_width`	`="wrap_content”`
`android:layout_height`	`="wrap_content"`
`android:textStyle`	`="italic”`
`android:textSize`	`="24sp"`
`android:textColor`	`="#ff364c"`
`android:gravity`	`="center"`

Note on the `android:text` attribute

All your applications strings should be placed in the various `/values/strings.xml` files in your app. This is the best practice, gathers all texts, and easily enables you to translate your app into other languages.

While developing apps, it might be easier to just hardcode the string values. Set the cursor inside the text you want to extract (here the text is the title `“Welcome to CSC3054/CSC7053”`) and hit `ALT + Enter`. Select `“Extract string resource”` as shown in Figure 2.

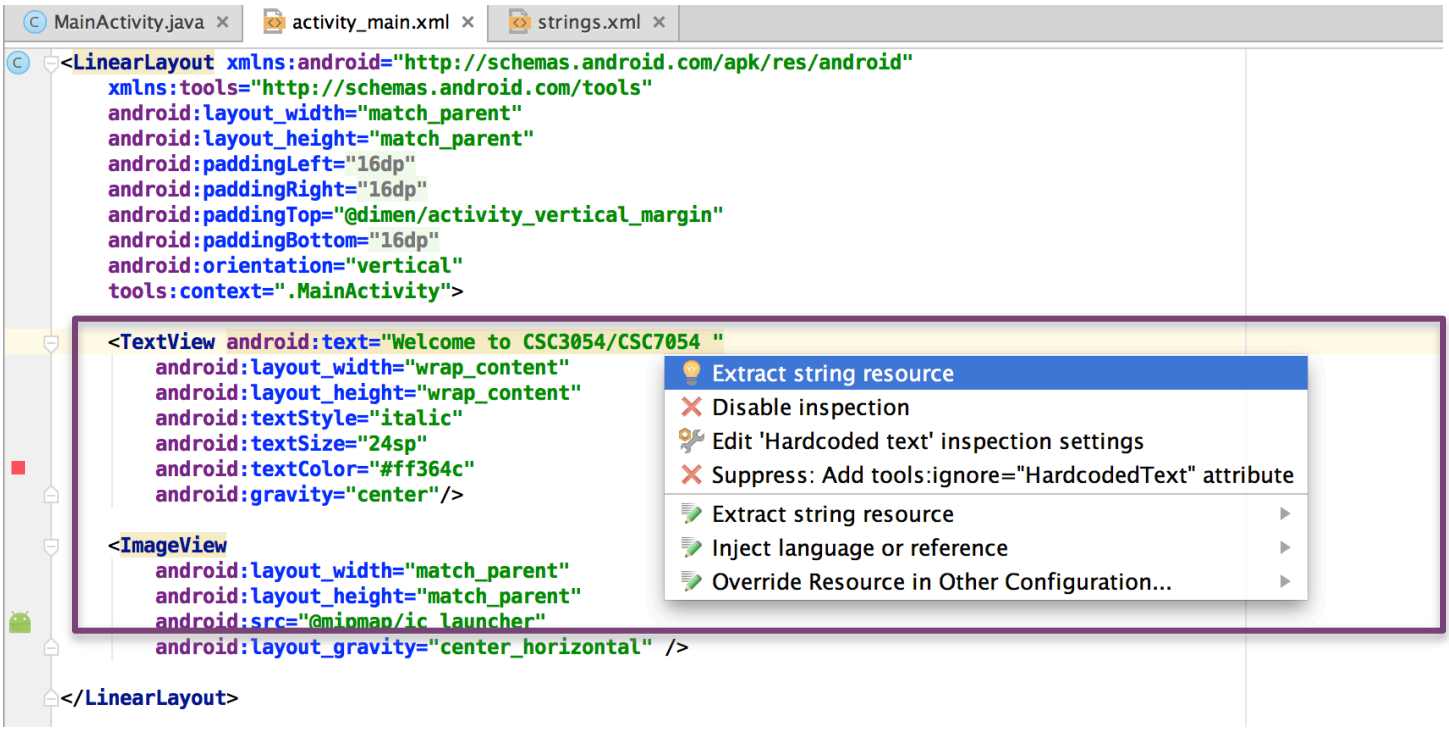

FIGURE 2 - EXTRACTING STRINGS

Give your resource a describing name and hit "Ok" as shown in Figure 3.

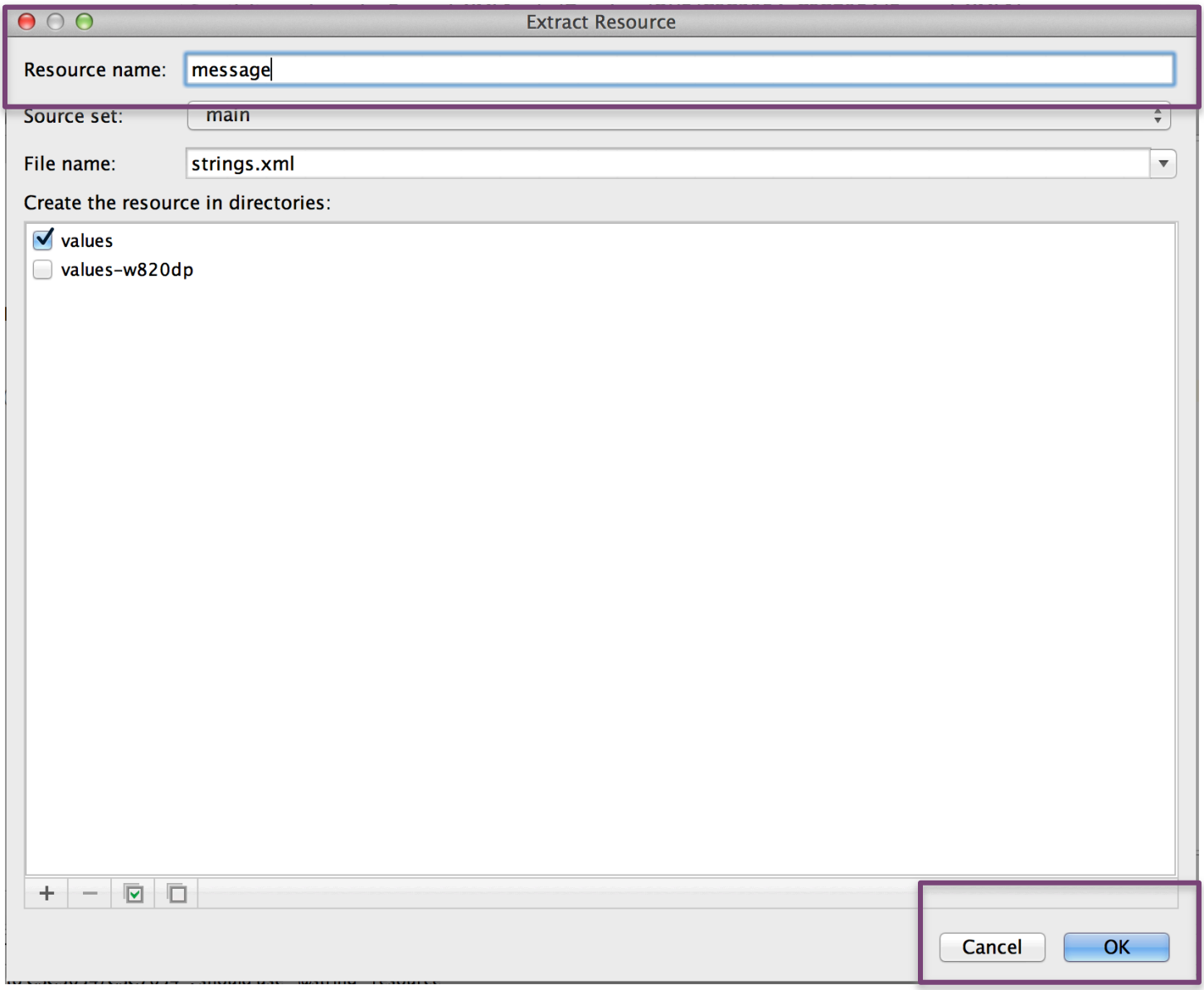

FIGURE 3 - CREATING A STRING VALUE

Your string has now been moved to the `strings.xml` file. Open this file by navigating through the project explorer as shown in figure 4.

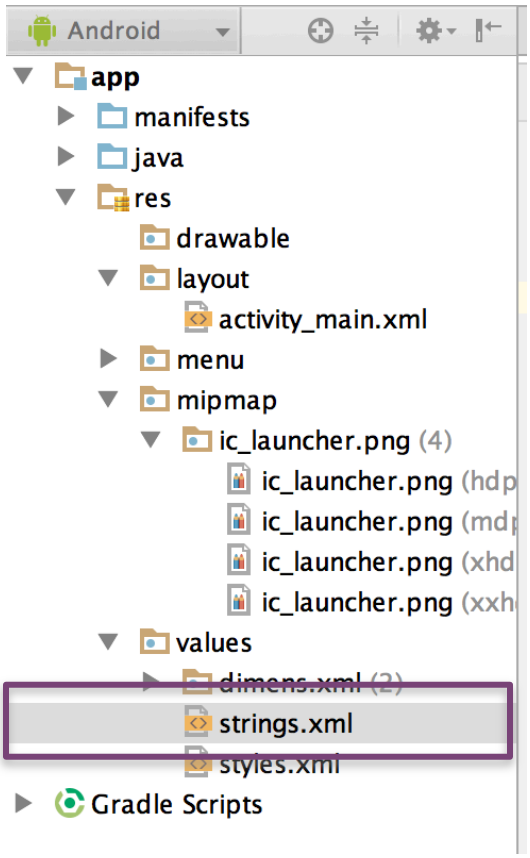

FIGURE 4 NAVIGATING TO THE STRINGS.XML FILE

Double click on the file and the file will open up as shown in figure 5.

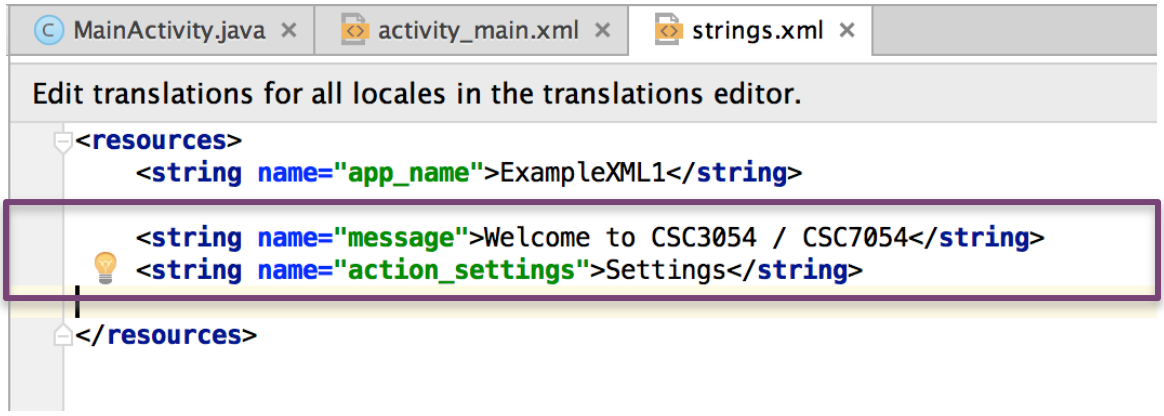

FIGURE 5 - STRINGS.XML FILE WITH NEW STRING ADDED

Step 3: Add an `imageView`

Add an `imageView` to your XML within the `LinearLayout`. Set the following attributes:

Attribute	Values
`android:layout_width`	`="match_parent"`
`android:layout_height`	`="match_parent"`
`android:src`	`="@mipmap/ic_launcher"`
`android:layout_gravity`	`="center_horizontal"`

Step 4: Switch to Design View

Now that you have added the XML code, switch to `Design View` to see how the views are rendered on the canvas as shown in figure 6.

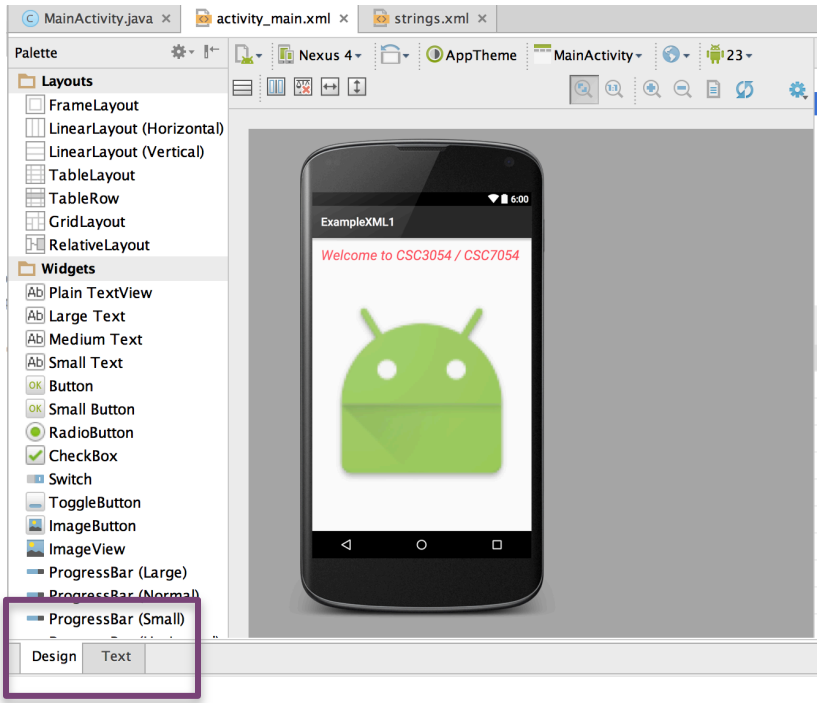

FIGURE 6 - RENDERED CANVAS

Step 5: Test the app on the emulator

Press the play button and run the app in the emulator as shown in figure 7.

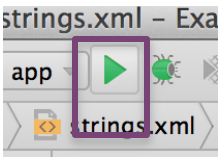

FIGURE 7 - RUNNING AN APP

When it runs in the emulator it should look like figure 8.

FIGURE 8 - COMPLETED APP

Complete XML Code

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView android:text="@string/message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="italic"
        android:textSize="24sp"
        android:textColor="#ff364c"
        android:gravity="center"/>

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@mipmap/ic_launcher"
        android:layout_gravity="center_horizontal" />

</LinearLayout>
```

Exercise 2 - Creating an App with four Views

In this app, you will create an app that contains four views (3 buttons and a TextView).

Before You Begin

Open `Android Studio` and create a new project called `"XMLExample2"`. Refer to the `'Creating your first project'` tutorial to help you create a project. Once created your project should look like figure 1. Switch from `Design` view to Text `view` .

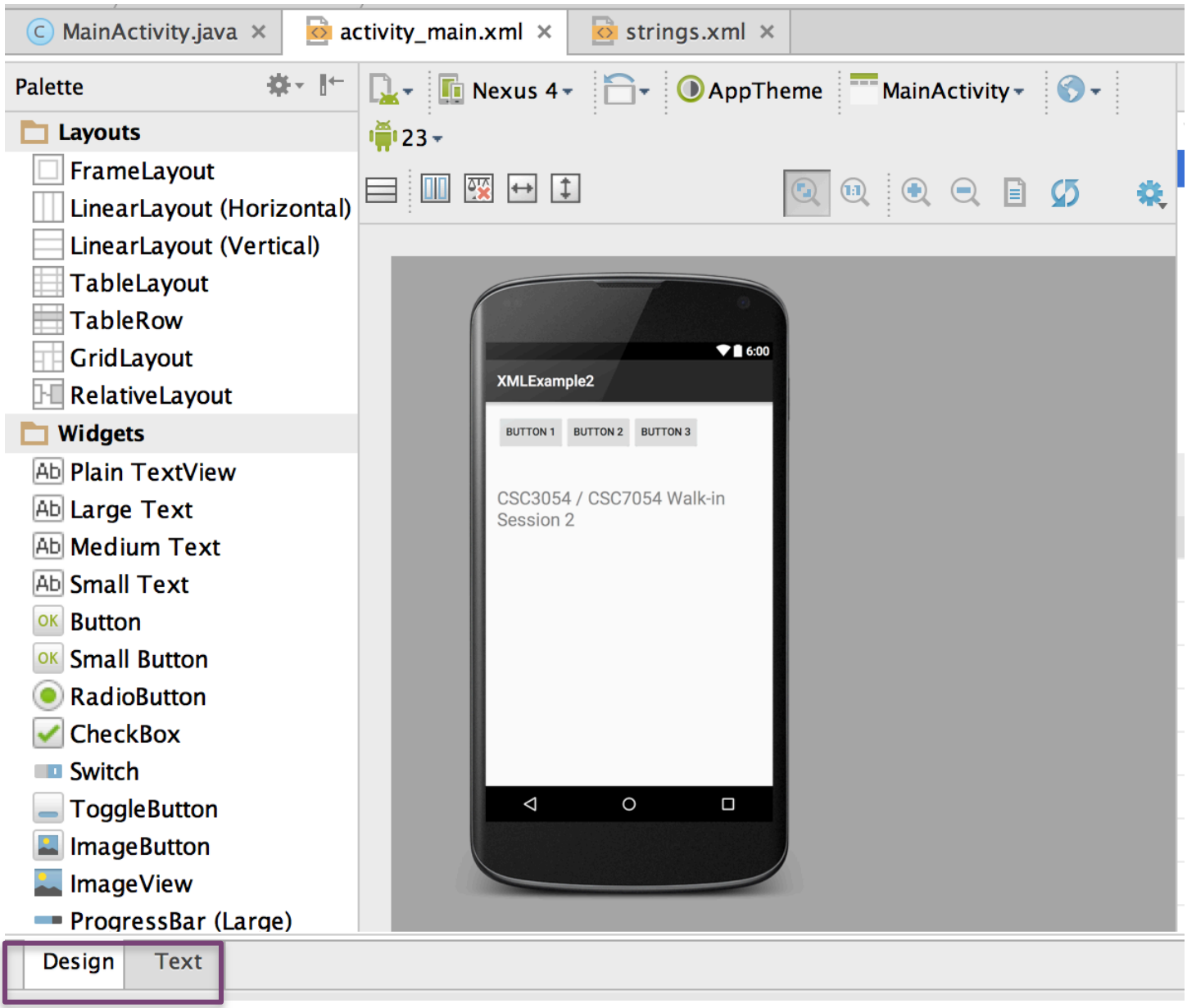

FIGURE 9 - OPEN PROJECT

Step 1 Change the Layout

First change the layout from `RelativeLayout` to `LinearLayout` and set the `orientation` attribute to `vertical`

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity">

</LinearLayout>
```

Step 2 Add another `LinearLayout`

Next, add another `LinearLayout` to the XML file but this time, set the `orientation` to `horizontal`

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"

    </LinearLayout>
</LinearLayout>
```

Step 3 Add 3 buttons to the `LinearLayout`

Add three buttons using the `<Button />` tag to this `LinearLayout` with the following attributes :

Button	Attribute	Value	Notes
Button1	android:layout_width	="wrap_content"	Remember to add the text string to the `Strings.xml` file
	android:layout_height	="wrap_content"	
	android:text	="Button 1"	
	android:id	="@+id/button1"	
Button 2	android:layout_width	="wrap_content"	
	android:layout_height	="wrap_content"	
	android:text	="Button 2"	
	android:id	="@+id/button2"	
Button 3	android:layout_width	="wrap_content"	
	android:layout_height	="wrap_content"	
	android:text	="Button 3"	
	android:id	="@+id/button3"	

Step 4 Add a Space View

Close the `LinearLayout`. Next, add a `Space` View. `Space` is a lightweight `View` subclass that may be used to create gaps between components in general purpose layouts.

```
    <Space
        android:layout_width="wrap_content"
        android:layout_height="50sp" />
```

Step 5 Add a `textView`

Add a `textView` using the `<TextView />` tag with the following attributes:

Attribute	Value	Notes
`android:text`	`="@string/message"`	The value of the attribute, `@string/message`, isn't what's displaying. What is specified in your layout file is not the actual `string` to be displayed but rather a `string` resource ID identifying the actual text. That way, all your app's strings can be in one place – `res/values/strings.xml`.
`android:layout_width`	`="match_parent"`	
`android:layout_height`	`="match_parent"`	
`android:textSize`	`="24sp"`	
`android:textAlignment`	`="center"`	

Step 6 Switch to Design View

Now that you have added the XML code, switch to `Design View` to see how the views are rendered on the canvas.

Step 7 Run the App

Press the play button and run the app in the emulator as shown in figure 10.

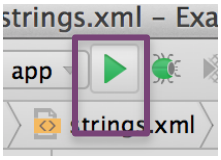

FIGURE 10 - RUNNING AN APP

When it runs in the emulator it should look like figure 11.

FIGURE 11 - COMPLETED APP

Full XML Code

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 1"
            android:id="@+id/button1" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 2"
            android:id="@+id/button2" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button 3"
            android:id="@+id/button3" />
    </LinearLayout>

    <Space
        android:layout_width="wrap_content"
        android:layout_height="50sp" />

    <TextView android:text="@string/message"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:textSize="24sp"
        android:textAlignment="center"/>

</LinearLayout>
```

Exercise 3 – Creating an App with multiple Views

Before You Begin

Open Android Studio and create a new project called "XMLExample3". Refer to the 'Creating your first project' tutorial to help you create a project. Once created your project should look like figure 12. Switch from Design view to Text view .

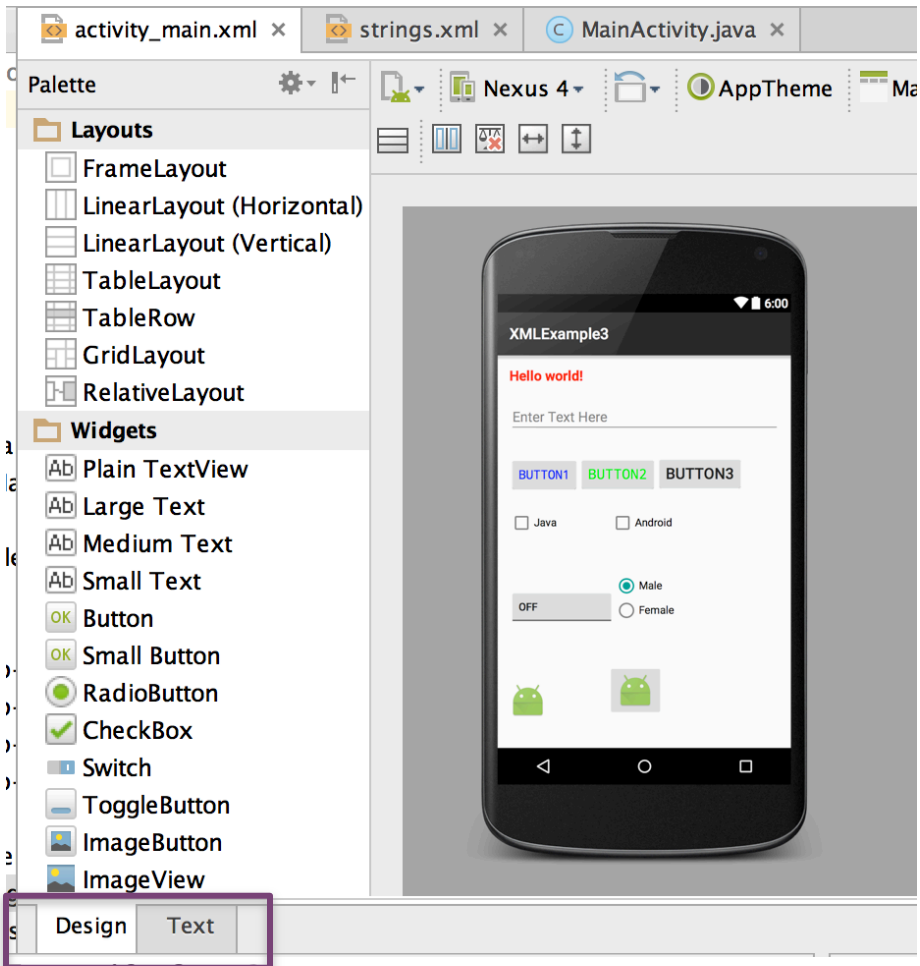

FIGURE 12 - OPEN PROJECT

Your task

Create the above app with the following view components:

View	Attribute
LinearLayout	Orientation: vertical
Text View	android:id = "@+id/textView1" android:text="@string/hello_world" android:layout_width="wrap_content" android:layout_height="wrap_content" android:textSize="18sp" android:textStyle="bold" android:textColor="#ff0000"
EditText	android:layout_width="match_parent" android:layout_height="wrap_content" android:id="@+id/editText" android:layout_alignLeft = "@+id/textView1" android:layout_below = "@+id/textView1" android:layout_marginTop="22dp" android:hint="Enter Text Here"
LinearLayout (this contains 3 buttons)	android:layout_width="wrap_content" android:layout_height="wrap_content" android: orientation: horizontal
Button	android:layout_width="wrap_content" android:layout_height="wrap_content" android:id = "@+id/button1" android:text="button1" android:typeface="monospace" android:textColor="#0000ff" android:textSize="16sp" android:layout_alignLeft = "@+id/editView1" android:layout_below = "@+id/editView1" android:layout_marginTop="28dp"/>
Button	android:layout_width="wrap_content" android:layout_height="wrap_content" android:id = "@+id/button2" android:text="button2" android:typeface="monospace" android:textColor="#00ff00" android:textSize="18sp" android:layout_alignLeft = "@+id/editView1" android:layout_below = "@+id/editView1" android:layout_marginTop="28dp"/>
Button	android:layout_width="wrap_content" android:layout_height="wrap_content" android:id = "@+id/button3" android:text="button3" android:textSize="20sp" android:layout_alignLeft = "@+id/editView1" android:layout_below = "@+id/editView1" android:layout_marginTop="28dp"

LinearLayout (this contains two checkboxes)	android:layout_width="wrap_content" android:layout_height="wrap_content" android: orientation: horizontal android:paddingBottom="50dp"
CheckBox	android:layout_width="wrap_content" android:layout_height="wrap_content" android:id = "@+id/checkBox1" android:text="Java" android:layout_alignLeft= "@+id/button1" android:layout_below = "@+id/button1" android:layout_marginTop="24dp" android:paddingRight="70dp"
CheckBox	android:layout_width="wrap_content" android:layout_height="wrap_content" android:id = "@+id/checkBox2" android:text="Android" android:layout_alignRight = "@+id/checkBox1" android:layout_alignBottom = "@+id/checkBox1" android:layout_alignBaseline = "@+id/checkBox1" android:layout_marginTop="24dp"
LinearLayout (this contains a Toggle Button, A Radio Group and two Radio Buttons)	android:layout_width="wrap_content" android:layout_height="wrap_content" android:orientation=”horizontal” android:paddingBottom="50dp"
ToggleButton	android:layout_width="wrap_content" android:layout_height="wrap_content" android:id = "@+id/toggleButton1" android:layout_alignLeft = "@+id/checkBox1" android:layout_below = "@+id/checkBox1" android:layout_marginTop="20dp" android:text = "ToggleButton" android:paddingRight="100dp"
RadioGroup	android:layout_width="wrap_content" android:layout_height="wrap_content" android:id = "@+id/radioGroup1" android:text="@string/hello_world" android:layout_alignLeft = "@+id/checkBox2" android:layout_alignTop = "@+id/toggleButton1" android:baselineAligned="false"
RadioButton	android:layout_width="wrap_content" android:layout_height="wrap_content" android:id = "@+id/radio0" android:checked = "true" android:text="Male"
RadioButton	android:layout_width="wrap_content" android:layout_height="wrap_content" android:id = "@+id/radio1" android:text="Female"

LinearLayout (this contains an ImageView and an Image Button)	android:layout_width="wrap_content" android:layout_height="wrap_content" android:orientation = "horizontal"
ImageView	android:layout_width="wrap_content" android:layout_height="wrap_content" android:id = "@+id/imageView1" android:layout_alignLeft = "@+id/radioGroup1" android:layout_below = "@+id/radioGroup1" android:layout_marginTop="23dp" android:src="@mipmap/ic_launcher" android:contentDescription = "No Image found" android:paddingRight="80dp"
ImageButton	android:layout_width="fill_parent" android:layout_height="wrap_content" android:id = "@+id/imageButton1" android:layout_below = "@+id/imageView1" android:layout_alignRight = "@+id/imageView1" android:src = "@mipmap/ic_launcher"

Once complete, test and run your app on the emulator. It should look like figure 13.

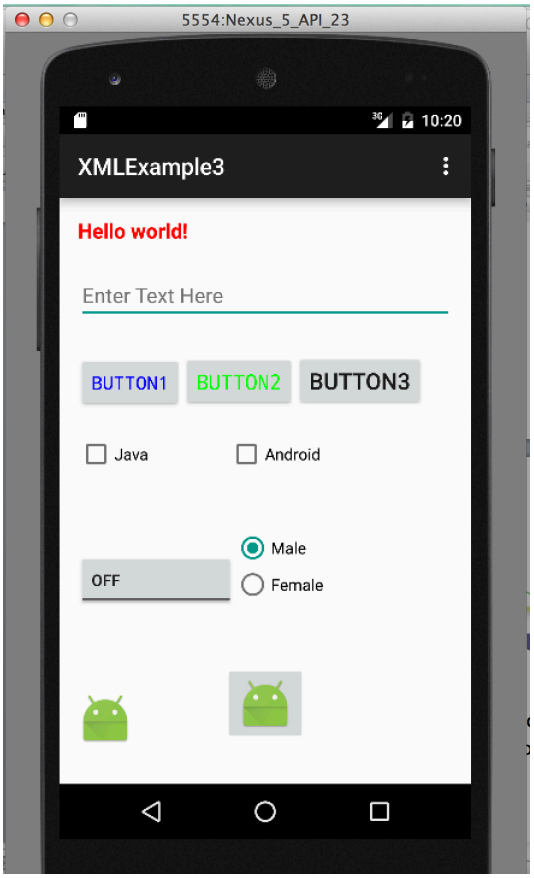

FIGURE 13 - COMPLETED APP