+ Queens University Belfast

Android Studio

CSC3054 / CSC7054

Changing the colour of a `TextView` when a `Button` or `RadioButton` is pressed.

Week 3 - Book 2

Queens University Belfast

Introduction

The goal of this app is to change the colour of a `TextView` when a `Button` or `RadioButton` is pressed. Different colors will be displayed depending on which `Button` / `RadioButton` is pressed by the user.

This app will consist of two Java classes `MainActivity.java` and `ColourSetter.java`. The `ColourSetter.java` class will implement the `View.OnClickListener` interface and import `android.view.View.OnClickListener`.

By creating two separate classes you are able to pass arguments to change the behavior of views. Separate classes generally promote loose coupling therefore, if an event handler is applied to different controls, it can be changed independently from rest of app. However in most real situations, behavior is tightly coupled to app.

If you want to call any code in `MainActivity.java` you need a reference. Even then, the code in `MainActivity.java` must be `public`.

Exercise 1 – Creating the XML

Before You Begin

Open `Android Studio` and create a new project called `"ButtonExample"`. 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` .

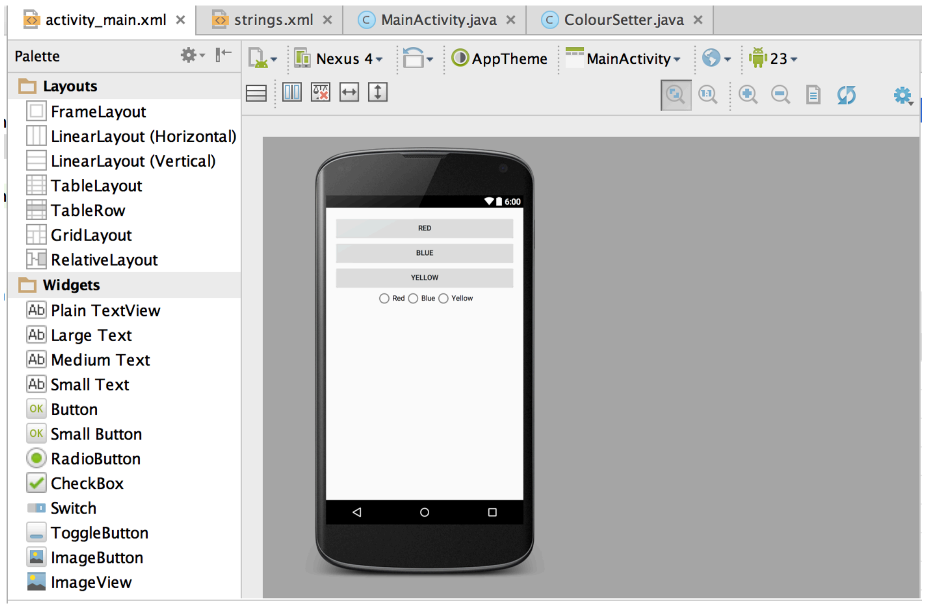

FIGURE 1 - OPEN PROJECT

This app contains the following XML views:

View	Notes	Details
`LinearLayout`	Overall the layout of this app is a `vertical` stack of graphical items	Orientation = `vertical`

`Button`	This part defines the 3 buttons.
	
	Each button is given an `id` so that it can be found in Java via `findViewById`, and then assigned an event handler via `setOnClickListener`.
	
	The `text` (`Button` label) is taken from `strings.xml` instead of being entered directly as the same value will also be used for the radio buttons.
		ID = `button1`
		Width = `match_parent`
		Height = `wrap_content`
		Text = `@string/RedPrompt`

`Button`		ID = `button2`
		Width = `match_parent`
		Height = `wrap_content`
		Text = `@string/blue_prompt`

`Button`		ID = `button3`
		Width = `match_parent`
		Height = `wrap_content`
		Text = `@string/yellow_prompt`

`Radio Group`	This group contains three radio buttons. A horizontal `RadioGroup` gives the same layout as horizontal `LinearLayout`. A `RadioGroup` also means that only one of the `RadioButtons` inside can be selected at any given time.	Width = `match_parent`
		Height = `wrap_content`
		Gravity = `center_hotizontal`
		Orientation = `horizontal`

`RadioButton`		Width = `wrap_content`
		Height= `wrap_content`
		Id= `radio_button1`
		Text= `@string/RedPrompt`

`RadioButton`		Width = `wrap_content`
		Height= `wrap_content`
		Id= `radio_button2`
		Text= `@string/RedPrompt`

`RadioButton`		Width = `wrap_content`
		Height= `wrap_content`
		Id= `radio_button3`
		Text= `@string/RedPrompt`

`TextView`	No text but controls will change the background colour of this region	Width= `match_parent`
		Height= `match_parent`
		Id= `colour_region`

Exercise 2 – Adding resources to `strings.xml`

The `strings.xml` file should have the following three resources added:

```
<string name="RedPrompt">Red</string>
<string name="blue_prompt">Blue</string>
<string name="yellow_prompt">Yellow</string>
```

`activity_main.xml` refers to these names with `@string/RedPrompt`, `@string/blue_prompt`, `@string/yellow_prompt`. Each string is used as a label for one `Button` and one `RadioButton`.

Exercise 3 - Creating the class `ColourSetter.java`

This is class that will handle the event that occurs when a `Button` or `RadioButton` is clicked on the app by the user. This class must store a reference to `MainActivity` so that it can call back to it. Another option is to pass the `TextView` to this class, but passing the `MainActiivty` is a more general solution.

```
import android.view.View;
/**
 * Created by nallen on 15/02/2016.
 */
public class ColourSetter implements View.OnClickListener {
        private int regionColour;
        private MainActivity mainActivity;

        public ColourSetter(int regionColour, MainActivity mainActivity) {
            this.regionColour = regionColour;
            this.mainActivity = mainActivity;
        }

        @Override
        public void onClick(View v) {
            mainActivity.setRegionColour(regionColour);
        }
}
```

Exercise 4 Creating the class `MainActivity.java`

Step 1 Import statements

Ensure you have the following `import` statements at the beginning of `MainActiivty.java`, just under the `package` declaration.

```
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
```

Step 2 Inheriting from class `Activity`

Ensure `MainActivity` extends `Activity` and create a `View` object called `colourRegion`.

```
public class MainActivity extends Activity {

    private View colourRegion;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }
}
```

Step 3 Assign `Views` to `Variables` in `onCreate`

The following code just looks up the `views` that were defined in `activity_main.xml` and assigns them to the variables.

```
colourRegion = findViewById(R.id.colour_region);

Button b1 = (Button)findViewById(R.id.button1);
Button b2 = (Button)findViewById(R.id.button2);
Button b3 = (Button)findViewById(R.id.button3);

RadioButton r1 = (RadioButton)findViewById(R.id.radio_button1);
RadioButton r2 = (RadioButton)findViewById(R.id.radio_button2);
RadioButton r3 = (RadioButton)findViewById(R.id.radio_button3);
```

Step 4 Assigning listeners to each Button and RadioButton in onCreate

The following code will assign the `ColourSetter` class as the event handler for each `Button` and `RadioButton`. You can pass arguments to this class (the `colours`) so that the same `ColourSetter` class can have different behaviours for different controls. You must also pass a reference the `MainActiivty.java (this)` so that the `ColourSetter` class can call back to the code in `MainActivity.java`

```
b1.setOnClickListener(new ColourSetter (Color.RED, this));
b2.setOnClickListener(new ColourSetter (Color.BLUE, this));
b3.setOnClickListener(new ColourSetter (Color.YELLOW, this));
r1.setOnClickListener(new ColourSetter (Color.RED, this));
r2.setOnClickListener(new ColourSetter (Color.BLUE, this));
r3.setOnClickListener(new ColourSetter (Color.YELLOW, this));
```

Step 5 - Create setRegionColour method

Since this method will be called by s method in the `ColourSetter` class, it must be `public`.

```
public void setRegionColour(int colour){
        colourRegion.setBackgroundColor(colour);
}
```

Exercise 5 Test the app in the Emulator

Press the play button and run the app in the emulator as shown in figure 2.

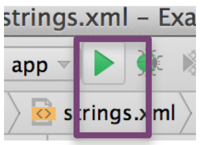

FIGURE 2 - RUNNING AN APP

When it runs in the emulator it should look like figure 3. Test that all functionality works.

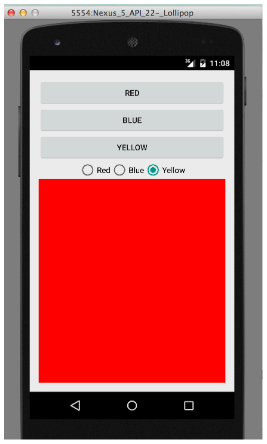

FIGURE 3 - COMPLETED APP