Queens University Belfast

Android Studio

CSC3054 / CSC7054

Randomly change the colour of a `TextView` based on a button click

Week 3 Book 3

Queens University Belfast

Introduction

This app will randomly change color of `TextView` when a `Button` is pressed. It uses the Random class to randomly select a colour from an array of colours.

Exercise 1 – Create the XML

Before You Begin

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

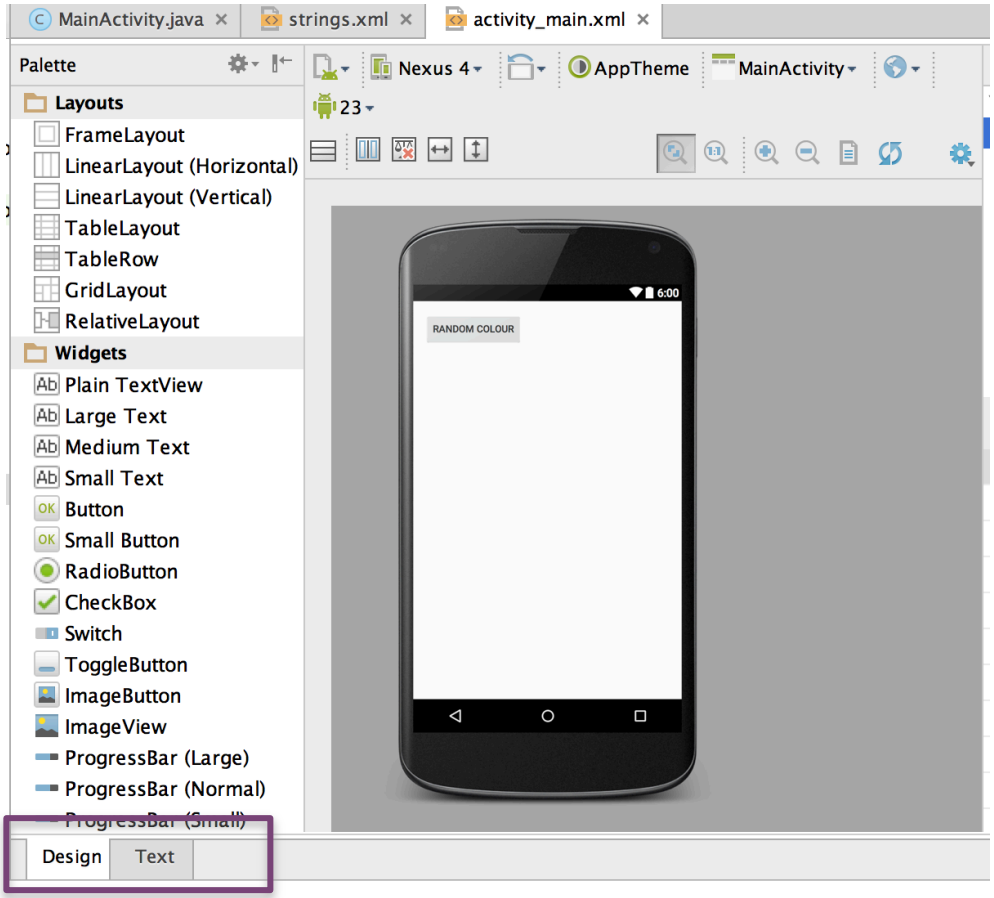

FIGURE 1 - OPEN PROJECT

XML Code

Component	Details
LinearLayout	orientation: vertical
Button	id: colour_button width: wrap_content height: wrap_content text: @string/ButtonPrompt
TextView	id: colour_region width: wrap_content height: wrap_content text: @string/ButtomPrompt

Exercise 2 – Update the Strings.xml

Strings.xml

```
<resources>
    <string name="app_name">EventHandleRandomExample</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="ButtonPrompt">Random Colour</string>
</resources>
```

Exercise 3 – Creating the Functionality

Step 1 – Import classes

In this app, we will make use of the class `java.util.Random`. This class provides methods that return pseudo-random numbers of different types, such as `int`, `long`, `double`, and `float`.

Normally (in other language/system), we will generate random number with seek of current time, to try to make the pseudo-random un-predictable. But in case of Android, it's not necessary and not recommended - Because "***It is dangerous to seed Random with the current time because that value is more predictable to an attacker than the default seed.***" - refer to Android document: http://developer.android.com/reference/java/util/Random.html

```
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import java.util.Random;
```

Step 2 - Ensure that `MainActivity` inherits from the `Activity` class

In android, the user interface of an application is displayed on a device through an `Activity`, typically with one `Activity` created for each unique screen. In Java, it is achieved by extending the original `Activity` class. Internally there is a stack of Activities, when moving from one screen to another `onCreate()`, `onResume()`, `onPause()`, `onStop()`, `onDestroy()`, `onRestart()` as show in Figure 2.

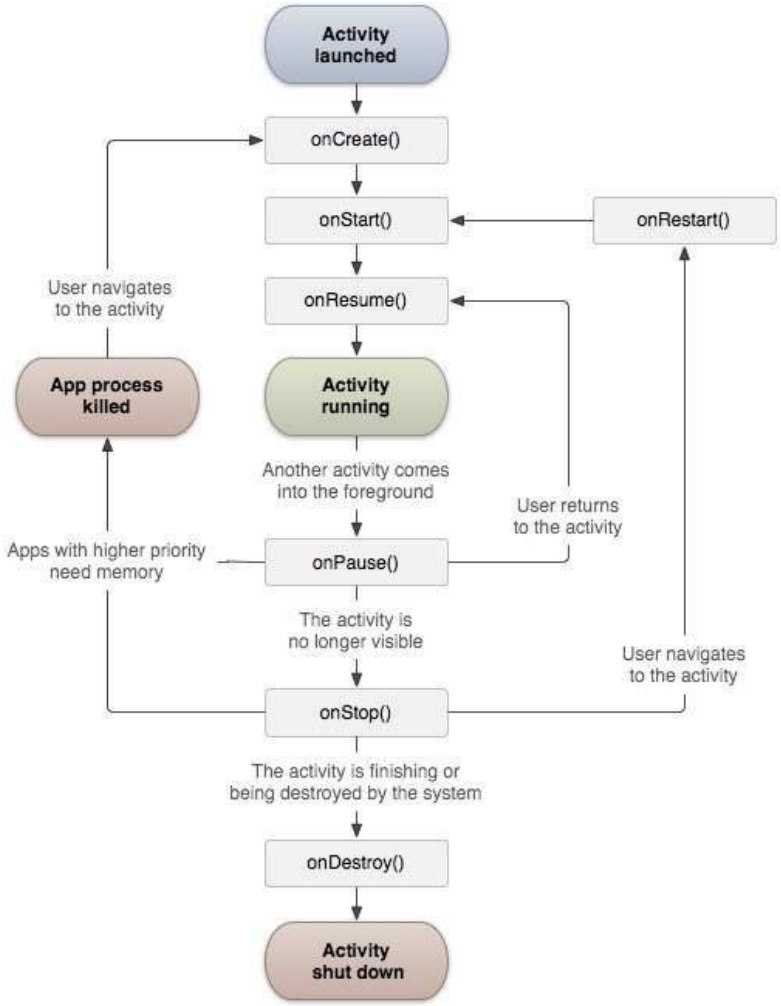

FIGURE 2 ACTIVITY LIFE CYCLE DIAGRAM: (*IMAGE COURTESY : ANDROID.COM*)

For further information on the `Activity` life cycle, please refer to the additional notes and exercises.

```
public class MainActivity extends Activity {
```

Step 3 - Declare Variables

Declare two variables a `View` called `colourRegion` and an `int` array called `colourChoices` that will hold the available colours for this app. In Android, colours are represented by the `android.graphics.Color` class. This class has pre-defined colours that make creating colour objects easy. These predefined colours include: `Color.BLACK`, `Color.MAGENTA`, `Color.YELLOW`, `Color.CYAN` and others. The `MainActivity` class defines an array of ints that contains some of the pre-defined colours in `android.graphics.Color`

```
private View colourRegion;
private int[] colourChoices = {Color.BLACK, Color.BLUE, Color.YELLOW, Color.CYAN};
```

Step 4 – Update method `onCreate`

Class `Random` supports two public constructors:

- `Random()` constructs a random generator with the current time of day in milliseconds as the initial state. This initial state is unlikely to be duplicated by a subsequent instantiation.
- `Random(long seed)` constructs a random generator with the given seed as the initial state.

In the case of this tutorial, we are creating a `Random` object using the first constructor. This object will be called '`generator`' and will be used to randomly generate a number based on the `length` of the array that was created in the previous step. This `int` will then be used as the `index` for the `colourChoices` array, which is passed in as a parameter into the `setRegionColour` method, which we will create in the next step.

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        colourRegion = findViewById(R.id.colour_region);
        Button colourButton = (Button)findViewById(R.id.colour_button);

        colourButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Random generator =  new Random();
                int index = generator.nextInt(colourChoices.length);
                setRegionColour(colourChoices[index]);
            }
        });
    }
```

Step 5 - Create additional method

This method takes an `int` parameter, which it will pass the `setBackgroundColour` method of the `View` object `colourRegion`.

```
    private void setRegionColour(int color){
        colourRegion.setBackgroundColor(color);
    }
```

Exercise 4 Test the app in the Emulator

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

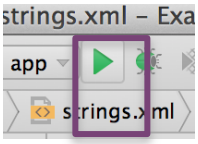

FIGURE 3 - RUNNING AN APP

When it runs in the emulator it should look like figure 4. Test that all functionality works.

FIGURE 4 - COMPLETED APP