KCL Tech: Build X: Android – Hello Android

Welcome future Android programmers. In this lesson we will be creating a new Android Project as well as gaining a better understanding of two languages that we will be using, Java and XML.

**Challenge 1: Run your first Android app**

Prerequisites:

- You’ve installed Android Studio

- You’ve installed an Android Emulator or have an Android device connected via USB

Creating a new Android project

1. Start a new Android Studio project

2. New Project

- Application Name: Hello World

- Company Domain: kcl.tech

- Package Name: tech.kcl.helloworld

3. Target Android Devices

- Minimum Required SDK: the earliest version of Android that your app runs on

- API Level: used in development, the version of the Android API (how apps work)

To support as many devices as possible, you should set this to the lowest version available that allows your app to provide its core feature set.

- I always recommend ‘API 16: Android 4.1 (Jelly Bean)’ as it is the highest API level that reaches roughly 100% of devices being used

4. Activities

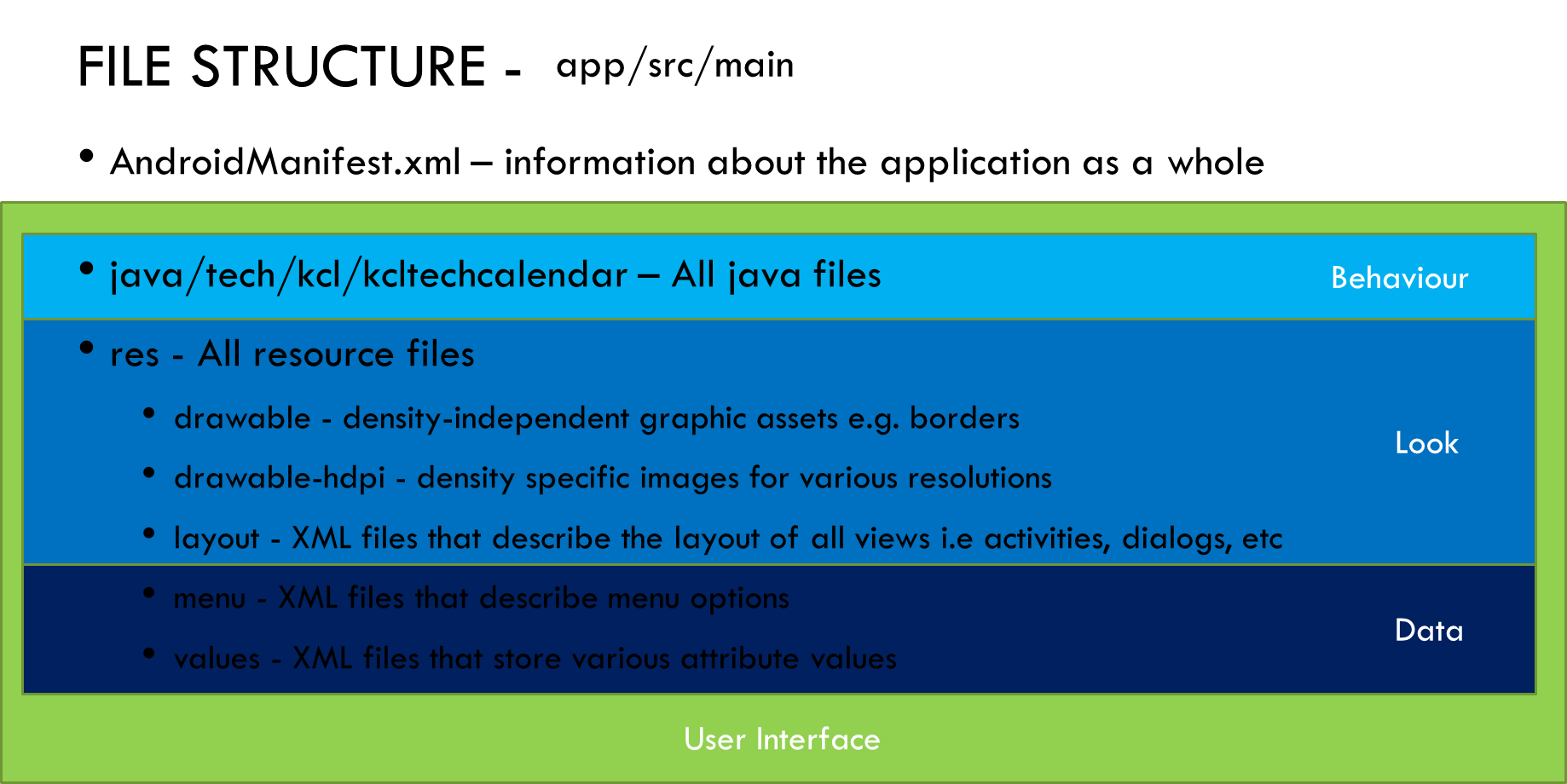
- Simply a page/screen. We will be starting with a ‘Empty Activity’ which is the minimum code for a screen.

5. Give your Activity a name. Or leave it as the default and click Next.

6. Run

NOTE: The Application Name you set is what comes up under the app icon.

File Structure



**Challenge 2: Make TextView fill screen**

XML does not DO anything. It's not a programming language. XML is a software and hardware independent tool for storing (and transporting) data.

In Android XML is used to create the look of the screen and/or store resource data like text and numbers. We can modify it by using either the Design or Text Interface tabs. The Design tab allows you to drag-and-drop elements whereas Text tab allows you to edit the actual XML.

Elements in Android are called ‘Views’. An example of a view is a *TextView* which can be used to display text. Other examples can be *EditText*, to take text input; *Button*, which is a button

Controlling how large a ‘view’ is on the screen can be done using the *android:layout\_width=””* and *android:layout\_height=””* tags*.* We can set to either fill the screen (*match\_parent*) or be the just big enough to show its contents (*wrap\_content*). We can also give a view an id using the tag *android:id=”@+id/view\_id”*.

**Challenge 3: Change the text on the screen to Hello Android using XML**

Android separates resources from its views. Resources can be text, numbers, images etc. This reduces data duplication, which means changing the data once changes it everywhere, also allows your app to be available to different languages by storing all translations with the same string name but in a different resource file.

The text of a view is set using *android:text=””*. We can either hardcode the text like: “Bad idea” or set it in the string resource file and call it like this: “@string/better\_idea”. String resources is added to string.xml like this: *<string name=”better\_idea”>Yay</string>*

**Challenge 4: Change the text on the screen to Android World using Java**

Java is an Object-Oriented programming language. An object is something with different states and behaviours. When programming, we use classes to describe objects. Classes have variables and functions. In Java, we can extend parent classes to inherit all their variables and functions, as well as having your own. All activities in Android are sub-classes of the class Activity. All views in Android, such as TextView and Button, are sub-classes of the class View.

To create Views in Java we must:

- Declare with a data type and name: TextView view

- Instantiate: TextView view = new

- Initialise with a value: TextView view = new TextView()**;**

Remember: All lines of code must end with a semi-colon **;**

However if the View already exists in the XML then we can find it using the method: *findViewById()* and passing the id for the view like so: *R.id.view\_id*. However *findViewById(R.id.view\_id)* always returns a view object. Therefore when dealing with specific views we must cast them. An example for Button is: *Button button = (Button) findViewById(R.id.button\_id);*

Comments can be added to our code like this *// Comment* or */\* Comment \*/.*

**Challenge 5: Understand the Android Activity Lifecycle**

When moving from one state to another, different methods are called to prepare it for the new stage. This can be useful to properly initialise your app on launch and properly save the changes when the app is closed.

The final part of this lesson will be to use the *append()* method on the TextView to log whenever an activity changes state. You can do this by generating all the state methods using code completion: Alt + Insert or Right Click + Generate

If you require any help, please raise your hand or give us a shout 