G54MDP Assessed Exercises
2: Countdown Clock

Introduction

In this exercise, you are to create a simple Android app that provides the functionality of a countdown timer. However, for an extra bit of fun, the display should resemble the VT countdown clock used within TV studios (see this video clip for details `http://www.youtube.com/watch?v=bqCb7-0f7fw`). This is an assessed exercise and will account for 15% of your final mark (together the two assessed exercises form 25% of your final mark for the module).

This exercise is due in at 23:59 on 3rd May, and your submission should be submitted as a `.zip` or `.tar.gz` file via the School's `cw submit` system. Details about how to use `cw submit` can be found at `http://support.cs.nott.ac.uk/coursework/cwstud/` or see one of us in the lab. Please note that your submission should be your own work and that plagiarism of other people's solutions (be they fellow students, or found on the web) is unacceptable.

Specification

This Countdown app starts with an `Activity` that allows the user to set the duration for the countdown timer in minutes and seconds. There should also be a 'Go' button which starts the timer. If the timer is already counting down, then the `Activity` should not allow a second timer to be started, but should let the user stop the timer and start a new one.

In addition, the user should bring up another `Activity` that shows the progress of the countdown as a clock that resembles a VT clock (see above clip) that countdowns the time left (when the timer has finished the clock should stop turning). In addition to the second hand shown in the YouTube clip above, you will also need to have a small minutes hand to show how many minutes are left in the countdown. Don't spend too much time trying to exactly match to the clock in the clip, but you should aim to match it as much as possible (e.g. the three-quarter circle outline, and numbers).

When the timer has finished you should fire a 'toast' on screen to tell the user that the timer has finished (using `Toast.makeToast`). This should happen even if the user has navigated away from your app, so you may want to use android's `Alarm` classes to cause the Toast to to be displayed when the timer finishes. You will want to make use of the an exact alarm here, since the device must wake up at that exact point. `Alarm`s where covered in the Google I/O videos we watched in lecture 16 and linked to for lecture 17.

To draw the clock, you will need to create your own custom `View` sub-class. This is something we briefly covered in the class, but it is relatively simple to do. In essence, all you need to do is create a sub-class of `View`, and implement the method `onDraw()`. This takes a `Canvas` object as a parameter and you can use it to draw lines, circles etc. making it very easy to create the design above. You can then reference the new view in your `.xml` file using the full package name, e.g. in my case:

```
<uk.ac.nott.cs.g54mdp.srb.alarmclock.ClockView android:layout_width="fill_parent"
                                               android:layout_height="wrap_content"
                                               android:id="@+id/clock" />
```

Note that the documentation suggest that you also need to implement `onMeasure()`, however, in my experience it isn't necessary for this `View`.

The following pages in the Google documentation will probably be of help:

- Documentation on `View` class
  `http://developer.android.com/reference/android/view/View.html`
- Documentation on `Canvas` class
  `http://developer.android.com/reference/android/graphics/Canvas.html`
- Documentation on 'Building Custom Components'
  `http://developer.android.com/guide/topics/ui/custom-components.html`

You may also want to look at `System.getCurrentTimeMillis()` and the `android.text.format.Time` class to enable you to get the current time and convert it into hours, minutes and seconds to draw the clock. To redraw the clock, you can call the `invalidate()` message on your `View` sub-class. This is something you'll need to do every second, so you may want to look at the `android.os.Handler` class to help pass the message from your update `Thread` to the UI thread, specifically the `sendEmptyMessage()` method. You should also ensure that this `Thread` is stopped when the `Activity` isn't visible on the screen.