**Lab7 Android的非同步執行**

1. 本節目的：

* 理解執行緒與非同步執行的觀念。
* 理解如何使用Thread類別與AsyncTask類別。

1. 觀念說明：

在前面章節中，我們知道應用程式的執行是在Activity之上，而Activity的運行好壞會直接影響到使用者的操作，假如說程式上設計不良，出現類似迴圈導致Activity執行時間過久呢？這時就會發生ANR(應用程式沒有回應)。



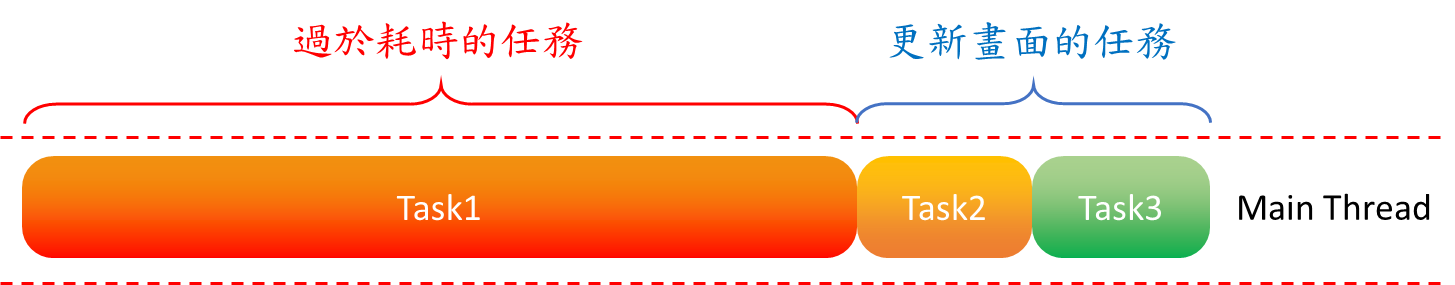
要解決此方法的關鍵就是使用非同步執行的方式來執行程式，以下會做說明。

1. **執行緒與非同步執行**

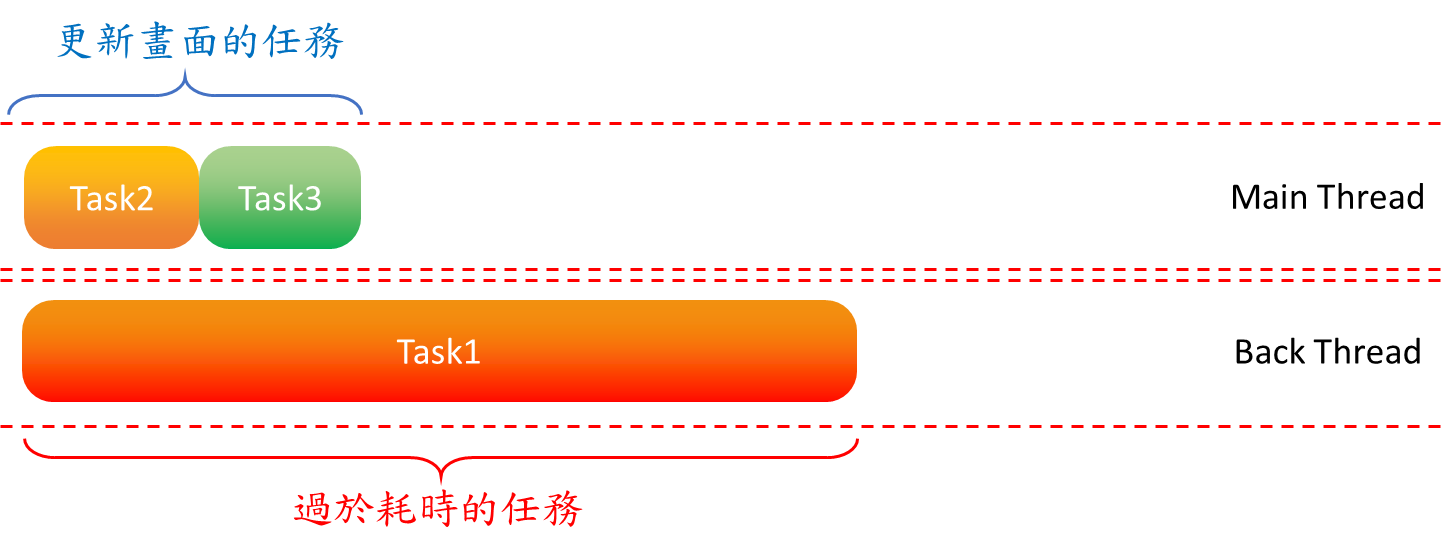
在沒有特別設計下，所有的任務(Task)都會在**Main Thread(或稱為UI Thread)**上執行，如下圖所示：



Main Thread(或稱為UI Thread)負責處理畫面更新的作業做更新，如果Main Thread其中一個任務(Task)非常耗時間，或是完成時間不可預期，如網路相關的動作、資料庫的動作、檔案操作或複雜的計算，使Main Thread無法執行更新畫面相關的操作，就會造成畫面卡住，甚至出現ANR情況，如下圖所示：



所以，我們需要把非常耗時間，或是完成時間不可預期的任務(Task)，使用**非同步的方式**來處理，透過非同步的方式，放到**Background Thread**來執行，這樣就可以避免Main Thread出現任務(Task)卡住的問題，如下圖所示

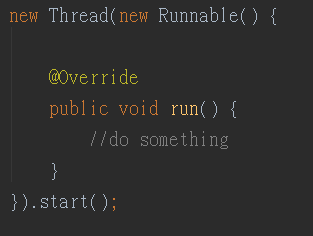


下面會介紹兩個非同步執行的方法：(1) Thread類別 (2) AsyncTask類別，Thread是Java本來就有的語法，而AsyncTask是Android提供的類別，使用這兩個類別方法來實現出非同步執行，可以把過於耗時的任務(Task)放到Background Thread，讓Main Thread的任務(Task)執行不會受到影響。

1. **非同步執行方法**

在Android中我們很常使用到非同步執行的方法，例如我們前面所教導的Toast就是很典型的非同步執行，他能在執行之後獨立運作，顯示期間Activity依然可以繼續的執行下去。

針對我們應用程式中的需求，我們可以產生出新的Thread去執行我們要實作的耗時作業，要在程式中實作一個新的Thread最簡單的方法可以用以下寫法：



產生一個Thread之後，我們需要用Runnable()介面，Runnable()會提供run()方法執行我們要跑的程式，因此要將要實作的程式碼寫在run()之內，然後使用Thread.start()方法將任務啟動。

由於Thread類別會產生出新的Background Thread去執行任務，但是Background Thread由於不是Main Thread，無法操作畫面的更新，因此當Background Thread中的任務需要操作畫面時，就必須要嘗試與Main Thread溝通，透過Main Thread來對畫面更新，這時就會需要使用到Handler類別。

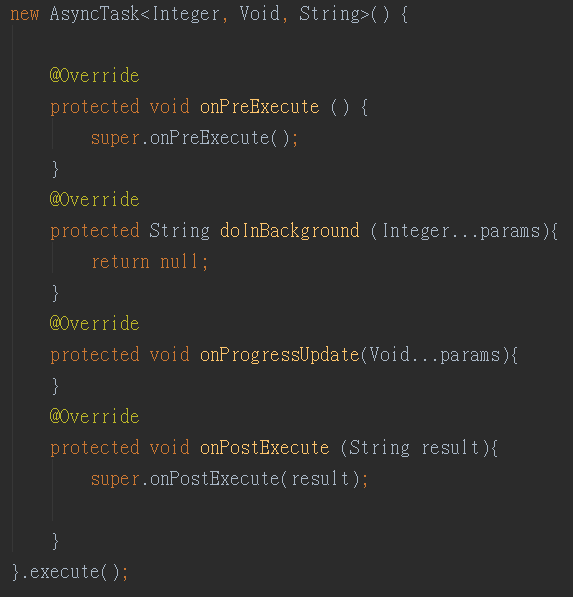
Handler是一種跨Thread的溝通機制，可以在一個Thread內把訊息丟到Message類別中，再讓另外一個Thread從Message中取得訊息。在Thread類別中，我們需要額外加入以下程式碼：

1. 首先，我們需要在建立一個Handler類別，在Handler中實現一個handleMessage()方法，利用handleMessage這事件來監聽Thread是否有送Message出來，並在Main Thread執行對應的操作。
2. Thread類別中需要建立一個對應的Message物件，Message會用於傳遞至handleMessage()，Message的what屬性可以加入一組代號，handleMessage()可根據代號來判別要做什麼。
3. 當Thread類別要發出Message時使用Handler.sendMessage()方法來觸發handleMessage()，如此就能把畫面操作透過Handler來執行處理。
4. **AsyncTask類別**

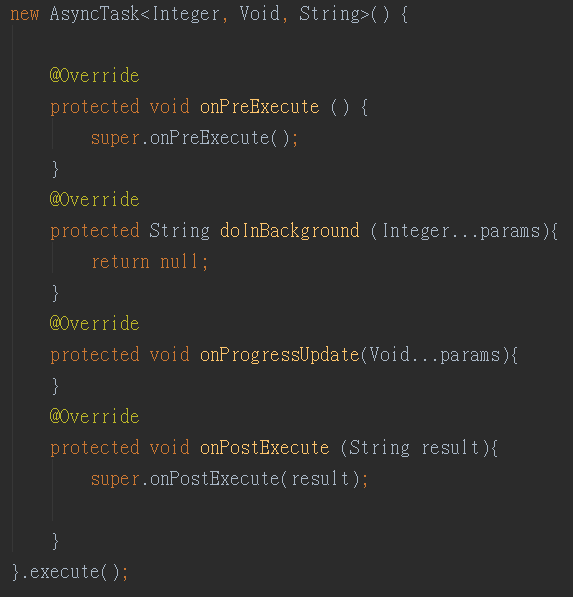
AsyncTask是用於實現非同步操作的一個類別，是Android平台自己的非同步工具，融入了Android平台的特性，讓非同步操作更加的安全，方便和實用。

AsyncTask可以方便的執行非同步操作(doInBackground)，又能方便的與Main Thread進行聯繫。AsyncTask 出現的目的就是在提供簡單易用的方式達成Handler、Thread 與 Message的功能， AsyncTask 只要定義幾個方法就可以達到他們的效果。

一個完整的AsyncTask的架構如下：



首先AsyncTask必須明確定義整個流程中的輸出入資料型態。因此一開始AsyncTask就須要帶三個標籤－**<Integer, Void, String>**，這三個標籤必須放入一個變數型態，如「Integer」、「Void」、「String」。這三個參數別用於定義**輸入的資料型態**、**進度更新的資料型態**、**輸出的資料型態**，程式中會對應到doInBackground、onProgressUpdate與onPostExecute的輸入值。

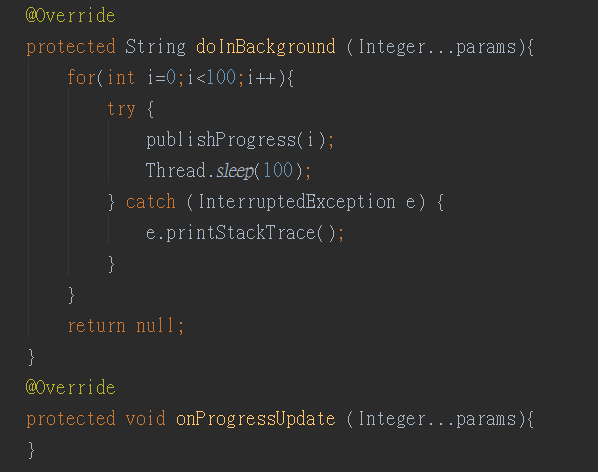


下面我們就AsyncTask提供的四種方法分別做介紹：

**onPreExecute**：這個方法會在AsyncTask開始非同步執行時被執行，這方法中可以讓我們初始化一些資訊或是保存一些變數在AsyncTask的區域變數之中，如果沒有必要資料的話此步驟可以被省略。

**doInBackground**：doInBackground是AsyncTask中唯一執行於新的Thread的方法，也是AsyncTask中唯一必須被實作的方法。執行於doInBackground中的方法會獨立被執行，直到完成後回傳結果。

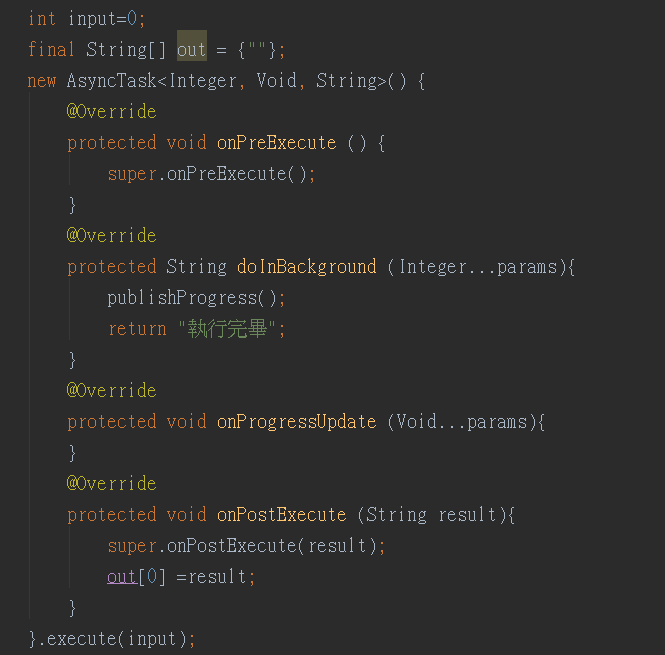
**onProgressUpdate**：onProgressUpdate是AsyncTask中用於監聽doInBackground進度的方法，我們可以使用他搭配ProgressDialog來實現進度條效果。我們可以在doInBackground中加入publishProgress()方法來觸發onProgressUpdate()，如下圖所示：



範例中，我們使用Thread.sleep()這個方法讓這個Thread延遲一段時間來呈現出等待效果，在迴圈中我們加入了publishProgress(i)將進度時間值發送出去，而onProgressUpdate就能接收到發出來的訊息。

**onPostExecute**：onPostExecute的主要工作在於處理任務結束後的回傳結果，當doInBackground完成了工作並回傳之後onPostExecute會接收到doInBackground傳來的結果，可以在這方法中對結果實作後續處理。

總結以上方法後，AsyncTask透過execute()方法啟用，execute()也可以傳入參數，該參數會被傳入到doInBackground之中，整理AsyncTask一次輸出入的流程如下：



1. 設計重點(龜兔賽跑專案):

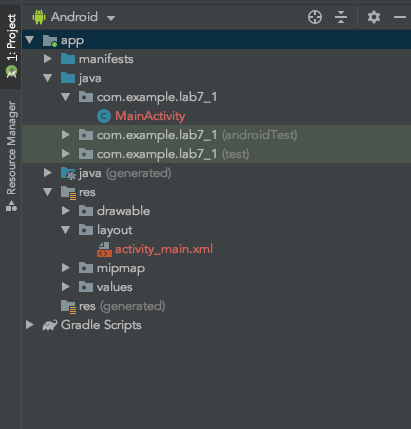
* 畫面中會使用兩個進度條(Seekbar)來模擬烏龜與的兔子跑步路線。
* 由於烏龜與的兔子的移動要同時進行，且都為耗時的工作，因此烏龜與的兔子的移動分別由AsyncTask與Thread執行。
* 按下開始後，同時啟動AsyncTask與Thread，並每0.1秒隨機讓各自的進度條增加0~2%的值，使兩個進度條同時的增加長度。
* 先抵達終點(進度條達到100%)，則會用Toast顯示出贏家

1. 設計步驟(龜兔賽跑專案):

Step1

新增專案，以及對應的java檔和xml檔。

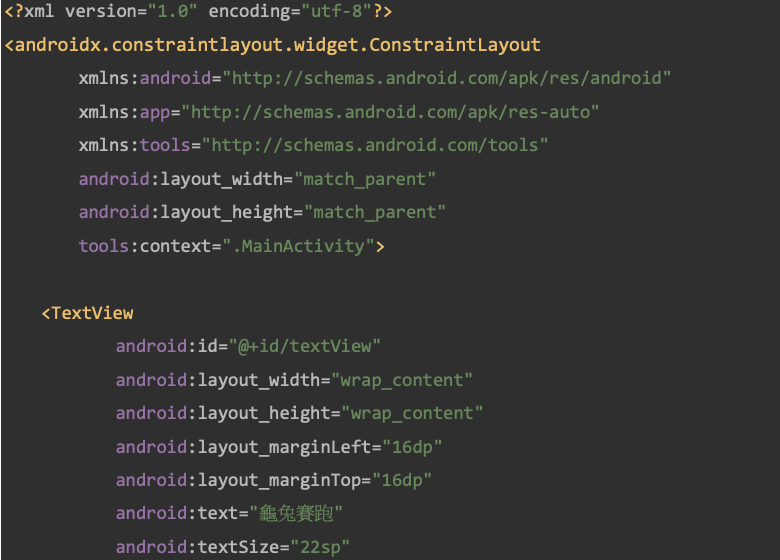


Step2

繪製activity\_main.xml檔



對應的xml如下：

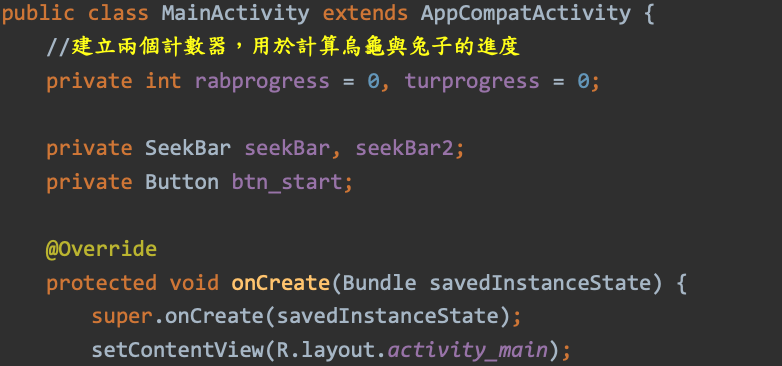


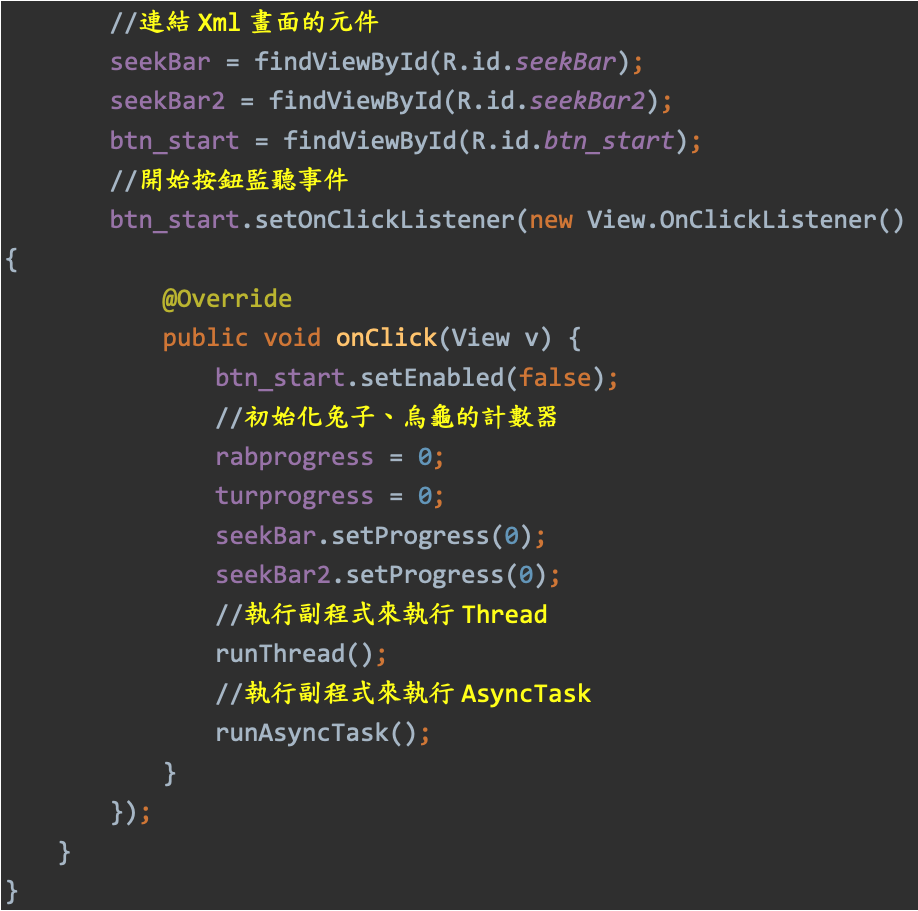




Step3

編寫MainActivity，按下按鈕後，分別執行runThread()與runAsyncTask()兩個副程式

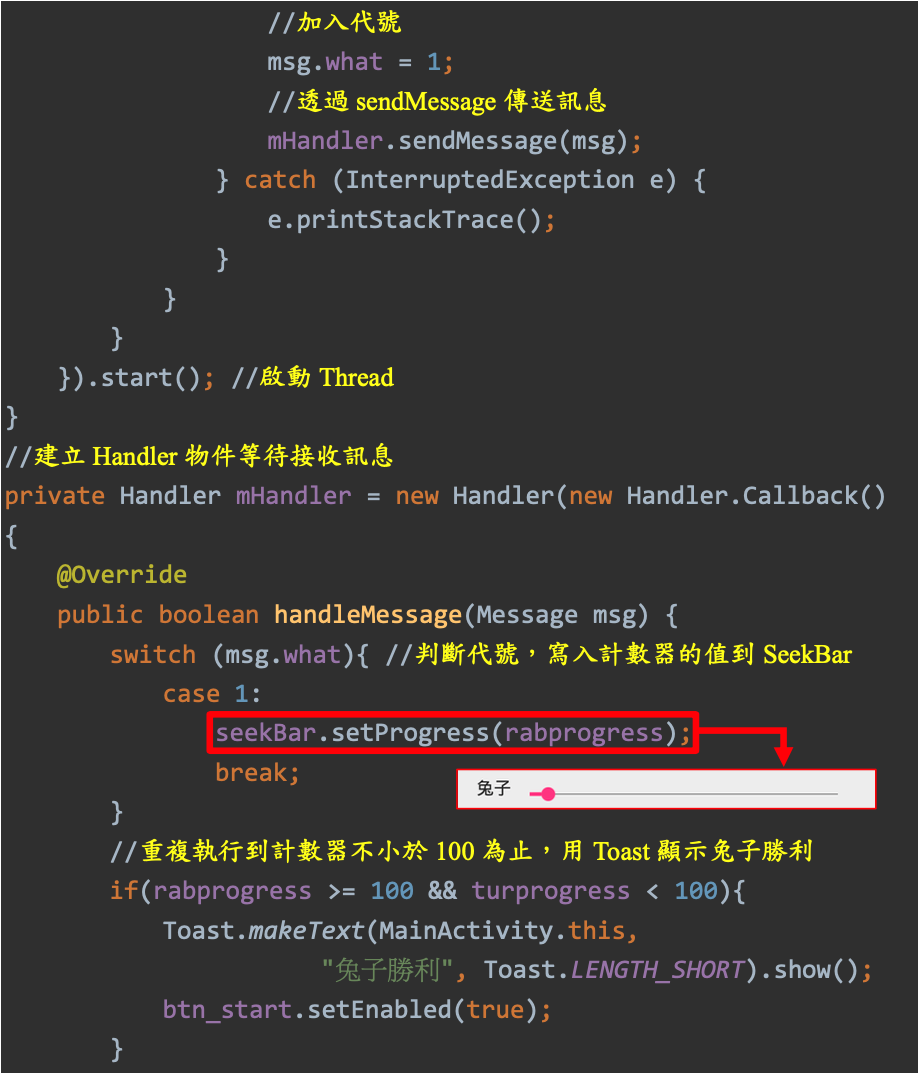




Step4

runThread ()中編寫執行一個Thread來模擬兔子的移動，每0.1秒隨機增加計數器0~2的值，透過Handler來顯示到Seekbar之上。

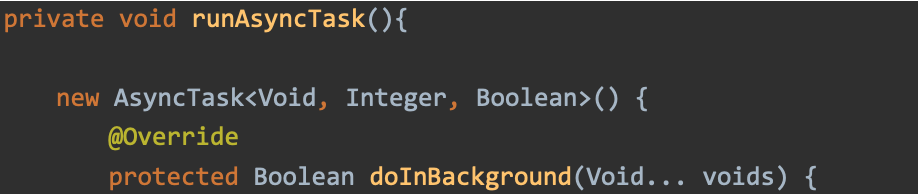






Step5

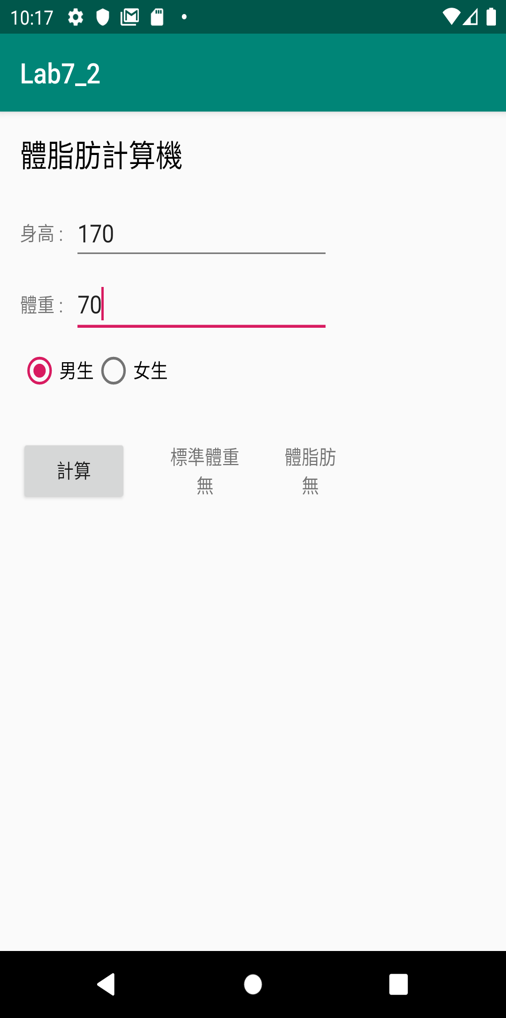
runAsyncTask ()中編寫執行一個AsyncTask來模擬烏龜的移動，每0.1秒隨機增加計數器0~2的值，並顯示到Seekbar之上。



****

1. 設計重點(體脂肪計算機):

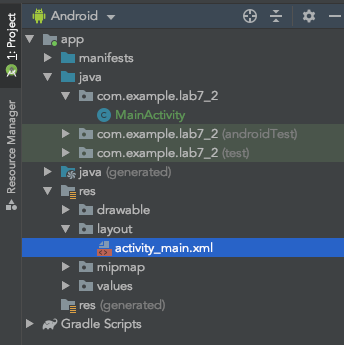
* 輸入身高(整數)和體重(整數)以及選擇性別後按下計算，會執行5秒左右，然後將體脂肪的結果做顯示。
* 為了呼應使用AsyncTask類別的情境，計算公式中我們加入Thread.Sleep()方法暫停執行5秒，模擬這個計算需要長時間執行，計算後得到標準體重和體脂肪。

1. 設計步驟(體脂肪計算機):

Step1

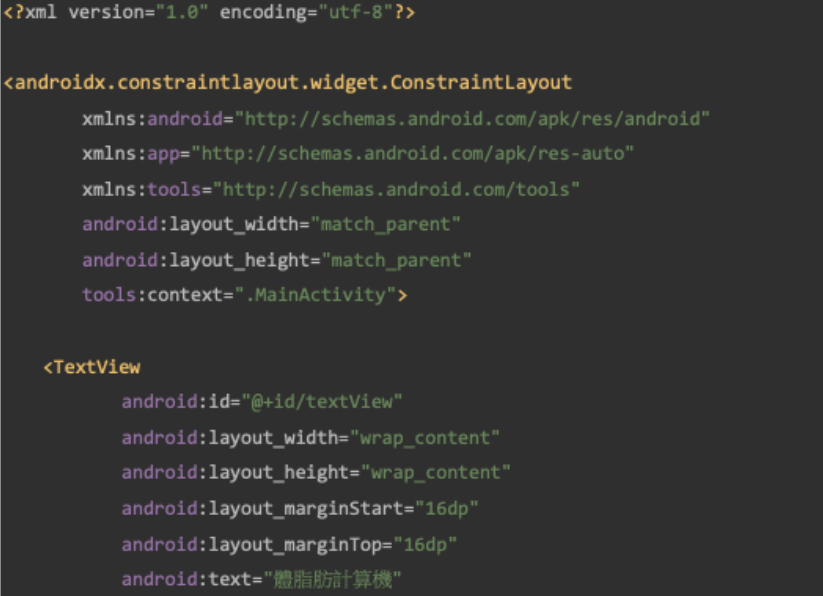
新增專案，以及對應的java檔和xml檔。

****Step2

繪製activity\_main.xml檔



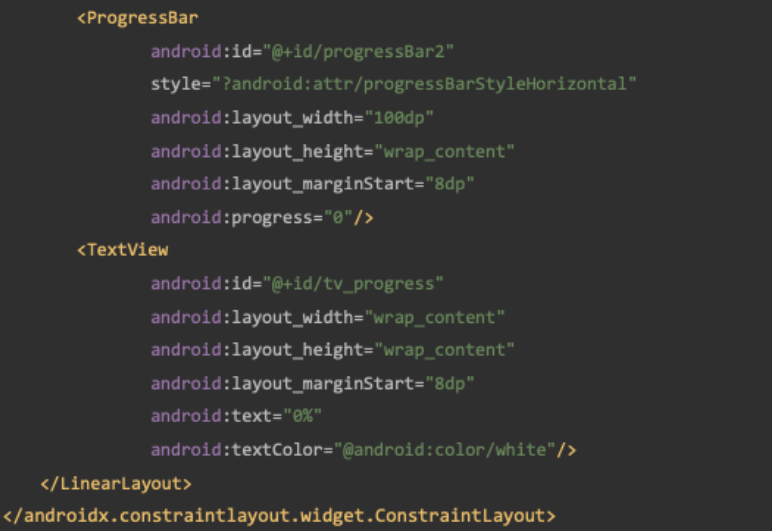
對應的xml如下：











Step3 在方法 onCreate 中建立 Button 監聽按下事件執行 AsyncTask。





Step4 runAsyncTask()中編寫AsyncTask。



