

應用軟體設計

Lab7-1&2

任課教師: 黃世嘉 老師

班級：電子三甲

學生：洪昌甫

學號 : 107360117

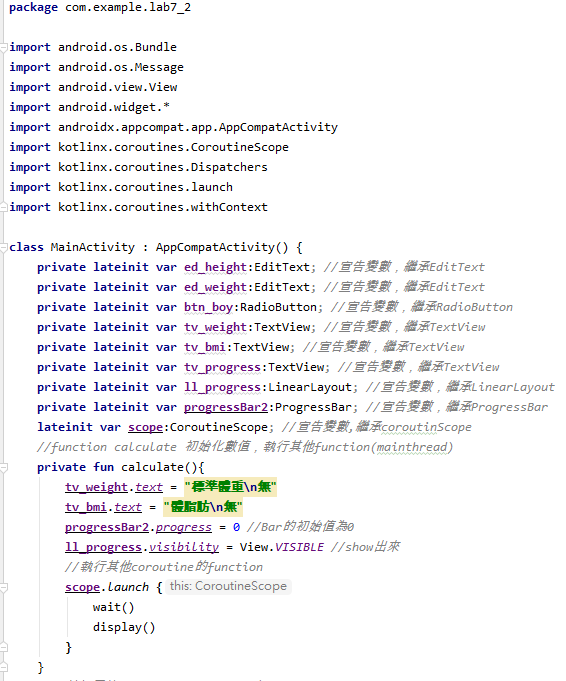
# 說明

Lab7-1





Lab7-2





# Lab7-1

**宣告區**

將所有元件宣告為延遲初始化的變數(latinit var)，並宣告兔子和烏龜的初始值為0。

**Main**

連接元件沒有變化，按鈕可否觸發設置要打”=false”，scope為coroutine的執行範圍（mainthread），Dispatchers.main為thread的調度員，負責調度thread的使用，執行兔子和烏龜的進度條function。

**兔子**

使用launch來開始執行coroutine，在calculate１中執行的是原本runthread的程式（負責產生進度條的數字變化），在calculate所使用的function前面要加上suspend代表是可以擱置的function，withcontext內的意思為此function將被調度於IOthread上執行，function內一樣使用message來傳達訊息給update讓它知道要執行了，update中withcontext內的意思為此function被調度於Mainthread上執行，update一樣是可擱置的，update內執行的為原本在更信進度條的工作也就是原本Handler在執行的事。

**烏龜**

烏龜進度條的更新重async改成使用coroutine來進行非同步的執行，其中和兔子一樣分為兩個thread在執行，原本在runasync中執行的現在都在可擱置的function　calculate2中執行，原本在onprogressupdate和onpostexecute中執行的都改道update2內執行，這邊的update一樣是接受來自calculte2的message來判斷是否進行更新進度條。

# Lab7-2

**宣告區**

將所有元件宣告為延遲初始化的變數(latinit var)。

**Ｍain**

連接元件的打法沒有變化，當按鈕觸發而且身高跟體重都又輸入後開始執行calculate，scope為coroutine的執行範圍（mainthread），Dispatchers.main為thread的調度員，負責調度thread的使用，calculate一開始為進度條的初始化，在launch兩個coroutine function，兩個一起執行。

**進度條亂數的產生**

與Ｌab 7-1的做法一樣，也是使用coroutine的方法來調度thread的使用，我將進度條數字產生放在wain()裡面，並開始了一個新的thread(IOthread可以開啟新的thread)， 因為在kotlin中好像沒有publish這個更新進度條的函式，所以我一樣使用了Message來讓update和wait溝通。

**進度條更新**

因為這邊沒有publish，所以一樣使用Ｍessage來讓update知道甚麼時候要執行（update執行時wait是被擱置的）

**計算**

我用IOthread開啟了新的thread來計算體脂肪和標準體重，裡面的程式沒有太多的變化。

# 心得

這次的Lab有點難度，因為原本的thread和async就已經很難了，所以沒有用過的coroutine似乎上手難度高了一點，而在查找許多資料後，發現coroutine最重要的就是scope(執行的範圍)、Dispatchers(調度員)還有context，context我還不太了解，它裡面包含了scope和job，應該是權限最高的，在編寫程式時我發現Message其實蠻好用的，像是在體脂肪計算中因為我不會使用publish，所以我就一樣使用了message來告知update要進行更新，而在thread調度上，我理解的是IOthread是可以強制開啟一個新的thread讓程式在背後執行，而會使用到Mainthread的都是在畫面上跑的進度條程式，以上的調度都是使用Dispatchers來調度，像是在行畫面更新時就畫讓亂數產生的程式暫停執行，而要想可以暫停的函數前面也要加上suspend，從字面上的意思就是它是可以被暫停的，所以我覺得這次的實驗難度很高，但確實coroutine應該會比原本那兩個thread、async的方式還好用，因為它不用下太多指令，而且我這次沒遇到null的問題，或許是因為我全部的變數都是延遲初始化。

Github:

<https://github.com/TommyHongtw/109-hw4-kotlin.git>