**應用軟體設計實習**

Hw5

任課教師：黃士嘉 老師

班級：電子三甲

學生：謝佳丞 106360122

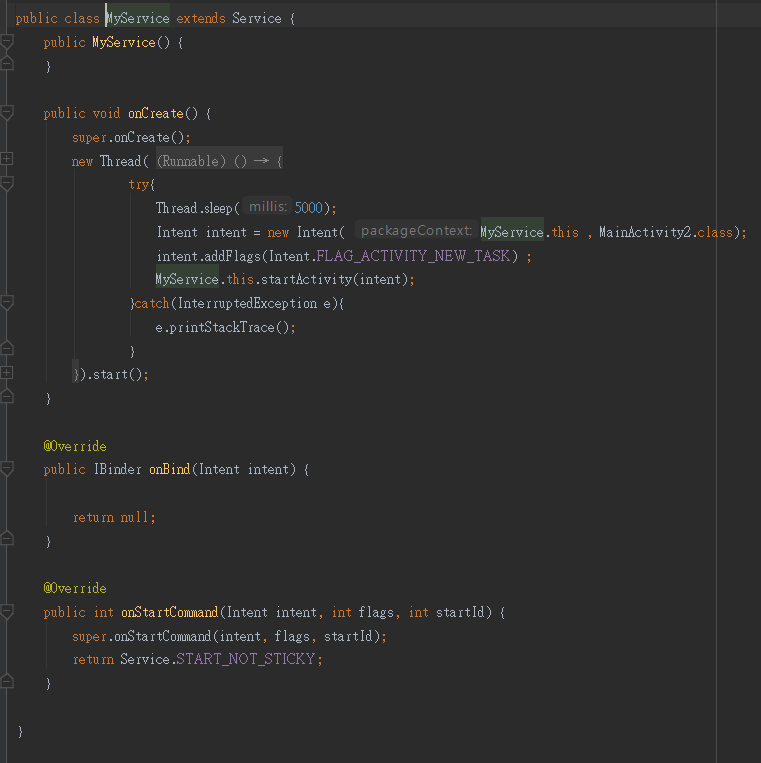
Service

在這個lab，Java與Kotlin的相似度很高，實際看起來沒甚麼差異，也只是宣告變數之類的不同而已，在這裡我就提個Service的概念就好。

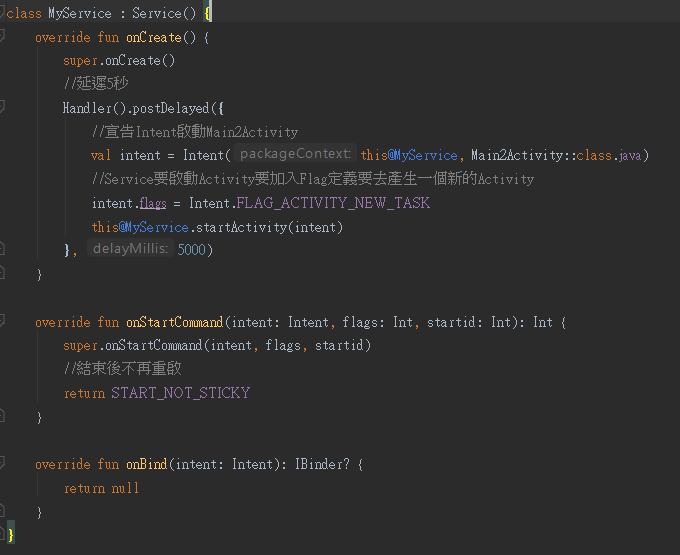
Activity 在離開畫面後會進入停止狀態，在這狀態下我們是無法控制 Activity， 除非透過調用 Thread 的方法，但是當 APP 完全結束關閉後，這些工作也會停止， 如果希望能在背景中繼續執行工作，我們就會需要使用到 Service 來執行背景作業，像是在背景等待網路連線、背景作業等工作。 Service 最大的特色是他的執行任務與使用者的操作無關，Service 會獨立運 行於背景，因此能夠 APP 完全結束關閉後能保持啟動，甚至能在使用者操作別 的 APP 時繼續執行，如等待網路訊息通知、下載資料等作業，一旦任務完成 Service 就可以被結束，也可能就此常駐於裝置上。

MyService 程式

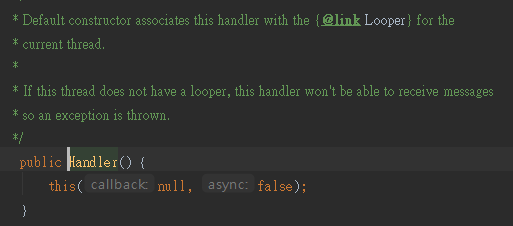
Java:



Kotlin:



Java在這裡用try…catch，Kotlin不需要。非常奇妙的，我查看了一下Kotlin所使用的Handler，這個Handler的功能，理所當然地跟Java的try…catch有著異曲同工之妙。Handler也會丟出Exception。



Broadcast\_Service

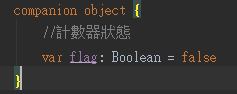
MyService:

**計數器狀態:**

Java:



Kotlin:



在設定計數器狀態就有不同的寫法出現，那麼這個**companion object**

**一定有存在的特殊意義；使用companion object是因為Kotlin沒有靜態成員或成員函數。 Kotlin建議僅使用包級功能。這會牽扯到程式當中的基本概念:**

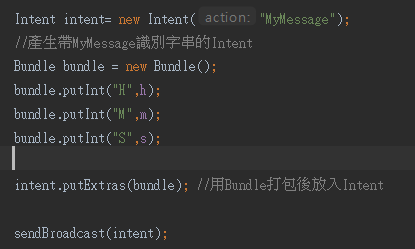
1.靜態的成員 不需要實體(Instance) 就能進行訪問

2.非靜態的成員 必須 new 一個 實體(Instance) 才能進行訪問

**如果您需要編寫一個可以在沒有”類”實例的情況下調用但需要訪問”類”內部的函數，則可以將該函數編寫為該”類”內部companion object聲明的成員。**

**發送廣播:**

**Java:**



**Kotlin:**

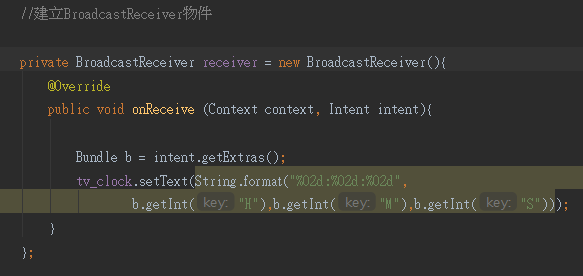


兩邊程式碼在使用sendBroadcast()前有不同之處，在Java會先用Bundle打包後放入Intent，而Kotlin宣告bundle的時候，就已經認定把之後產生的資料都打包好。

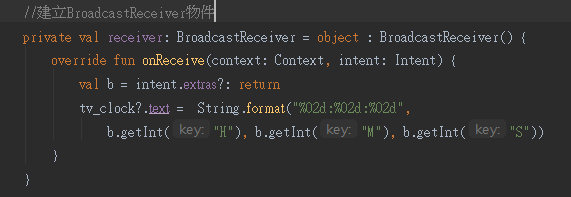
**MainActivity:**

**建立BroadcastReceiver物件:**

Java:



Kotlin:



Java是把b宣告成Bundle物件，之後可以打包再回傳，在Kotlin比較彈性的把b當作變數而已，而且是只有有資料改變就會回傳改變後的資料。

這就是為甚麼在MyService裡的發送廣播前，Java要先打包材放入intent，而Kotlin不需要(因為已經return了)。

心得:

這次的Lab雖然不多，但是我發現在傳送資料的方法上，兩種語言還是有一定的差異，Java利用intent切換Activity時，可以用Bundle攜帶資料憶起切換，但換到Kotlin就不用那麼麻煩了呢?但我認為在Activity之間的資料傳遞上不太需要去鑽牛角尖，因為那只是程式的特性而已。

我在寫Kotlin的時候，總是在想為甚麼Java這麼做，而Kotlin為甚麼可以那麼做等等……，我認為搞清楚差異很好，但是在語法上確實不用那麼糾結，不然學也學不完，我得學會在相同的概念前，適應可以不同的方式表相同的概念這件事。