第五章：執行緒(Thread)

本篇介紹有關執行緒的概念，以及使用java語言的特性：多執行緒來執行更多樣的工作。

<https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/75324/>

1. 何謂執行緒(Thread)

* 一個程式運行的路線就是一條執行緒，所以常用的main方法就是一條執行緒，並稱為主執行緒(main thread)

執行緒

程式結束

程式開始

* 單一執行緒就是應用程式執行事情的單一條行程(process)，所以多執行緒就是在程式中呈現多條行程的現象，它可以模擬出同步處理的效果，意指多工(multi-task)的效果
* 執行緒vs 多執行緒

→多執行緒用來與主執行緒分開，目的在於不影響程式進行的情形下，另外完成其它動作的技巧

* 為何要使用多執行緒？

如果只有一條執行緒：

主程序被拖累…

下一個動作進來

主程序執行中…

主程序執行中…

同時進行，程式運行變慢

主程序暫停，下一個動作先跑

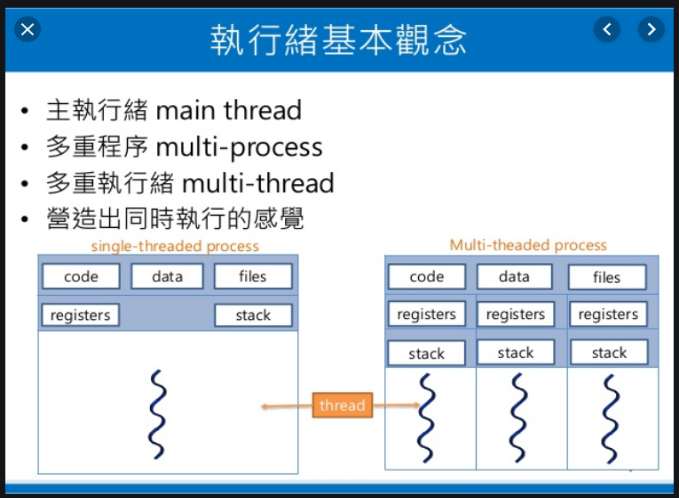
B計畫

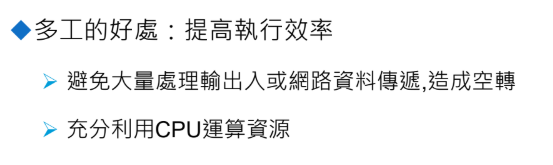
A計畫

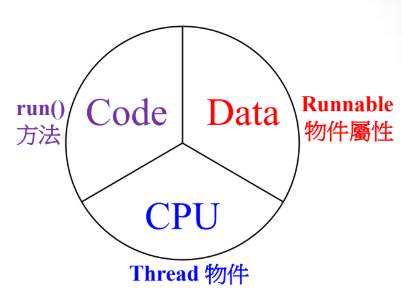
如果改多執行緒方式執行：

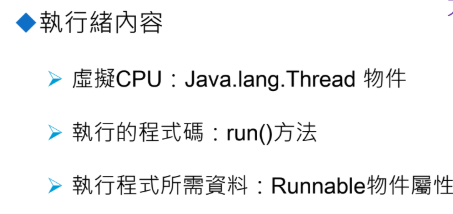
下一個動作進來

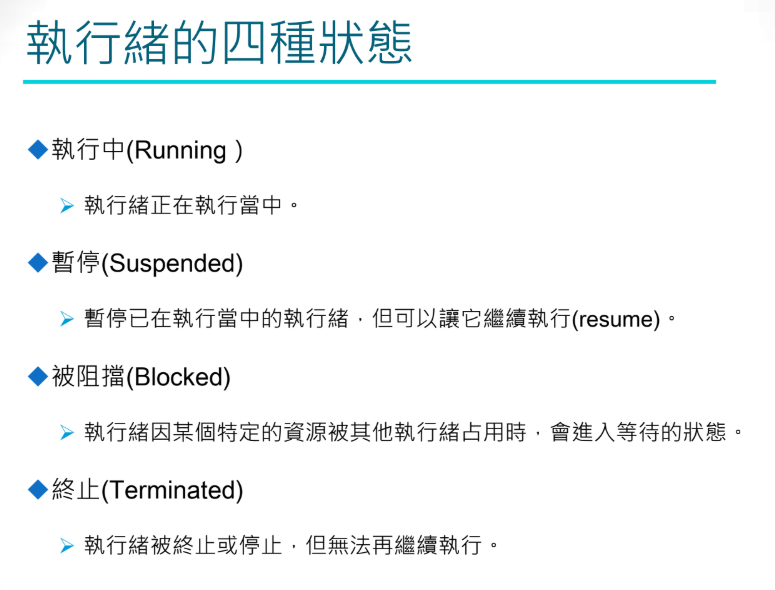
主程序執行中… 執行緒並行區塊… 主程序執行中









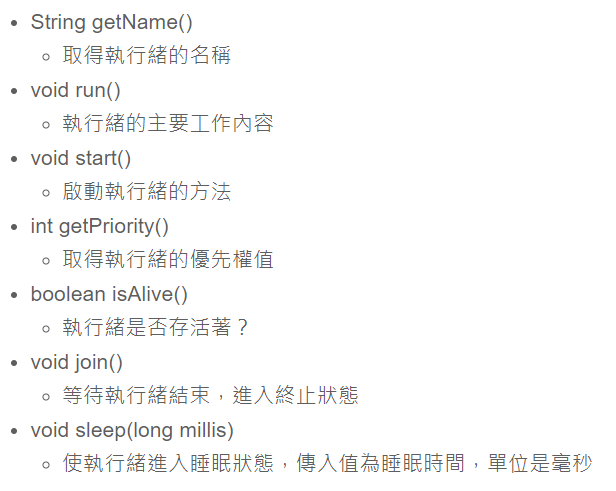


1. 建立thread執行緒

* 建立一個執行緒的三步驟

1. 新開一個類別繼承**Thread**類別(java.lang.Thread)
2. 覆寫thread的覆寫方法：run();，方法內寫要執行的項目
3. 建立此類別物件後呼叫start()方法

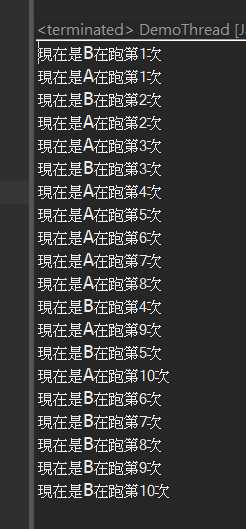
* 執行緒常用的方法



* 執行緒的簡單範例 - Thread

下面是一個簡單示範多線程的範例，這裡開了兩條線，做同樣的事情，並觀察執行結果





第二條結束了

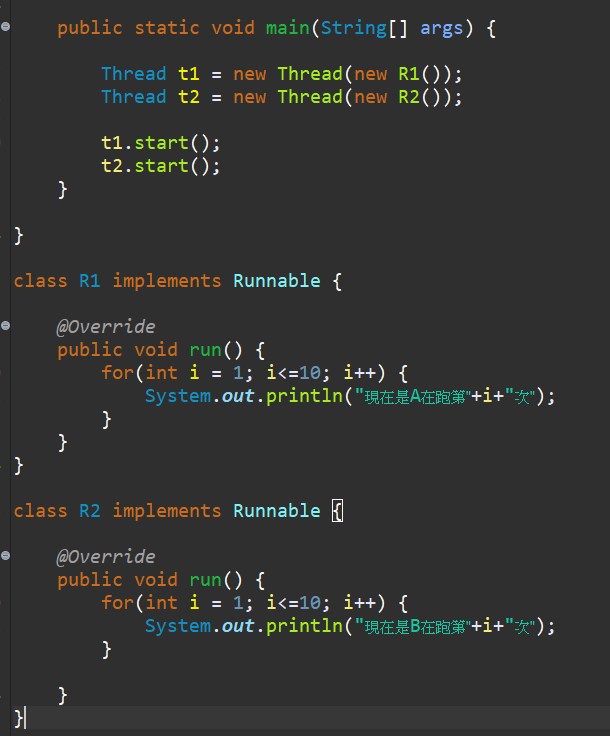
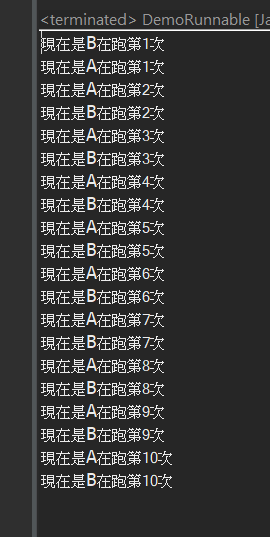
第一條結束

兩條同時進行！

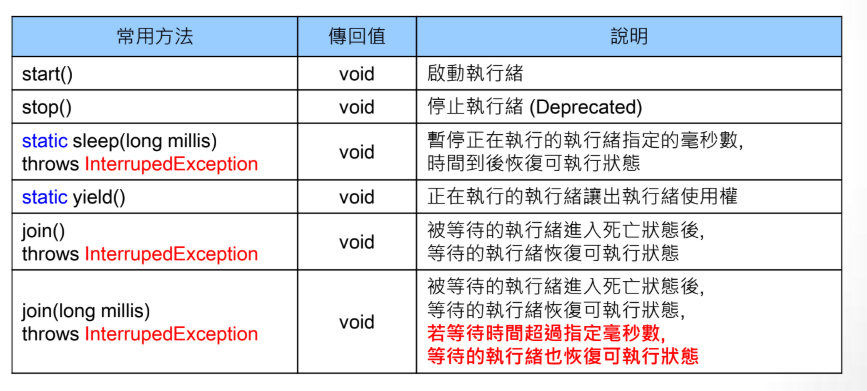
執行順序開始不一樣！

→兩條程式同時進行，速度不一定！所以每次結果不一樣

* 執行緒的簡單範例 - Runnable執行緒

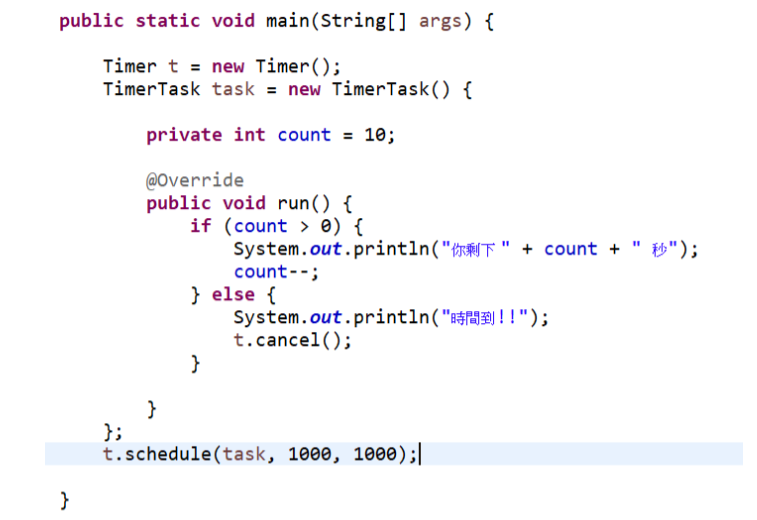
* Thread類別的常用方法



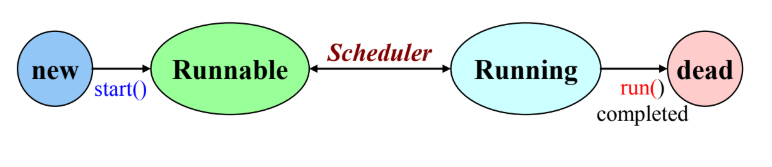
* mills指的是毫秒，1/1000; 1000毫秒=1分鐘
* 使用多執行緒來寫一個計時器



* 上次寫過的timertask



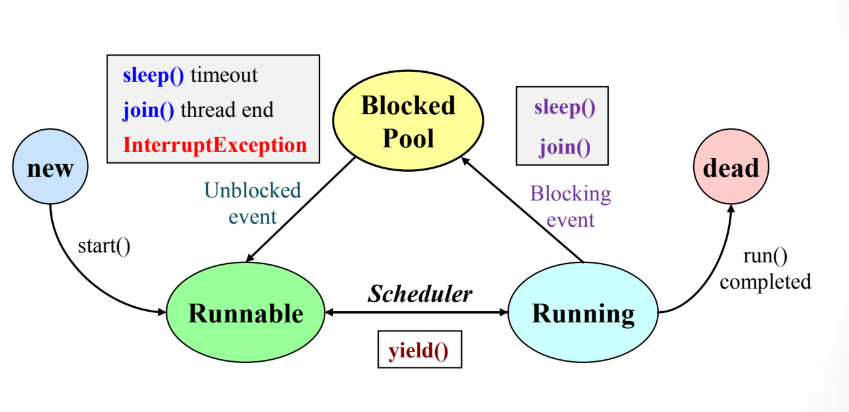
* 執行緒的周期



* 暫停thread的幾種方法

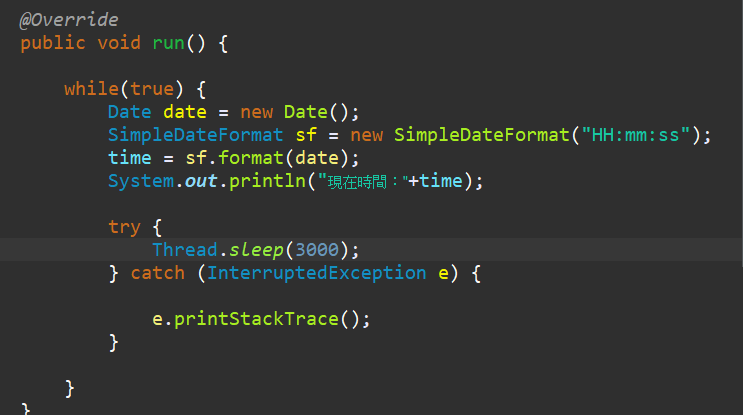


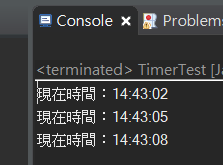
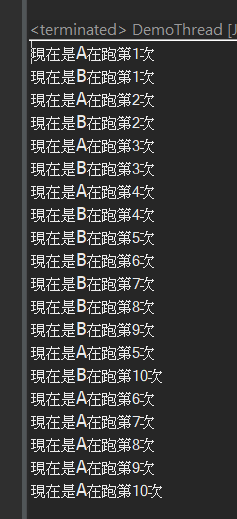
* 執行緒被迫暫停時，會停在一個blocked pool內，直到暫停狀況主動或被動結束後才會回到原執行緒繼續執行



* Sleep方法

用剛剛的計時器來測試sleep方法：



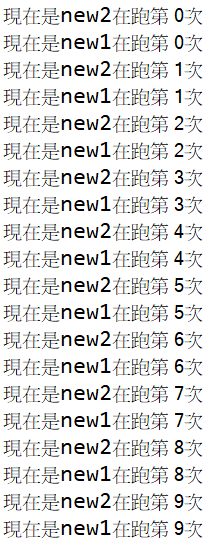


* Yield方法

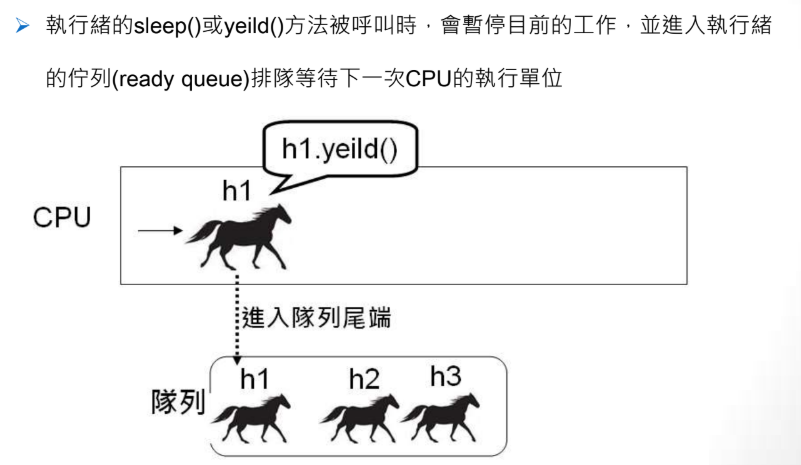


→初期會讓出，交替進行，但後面就失控了

* Yield搭配sleep方法

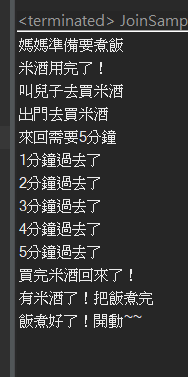


→執行多次測試後得到規律的交互進行現象比較多次！

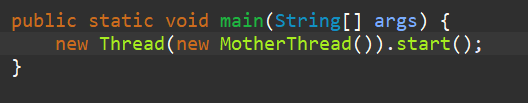


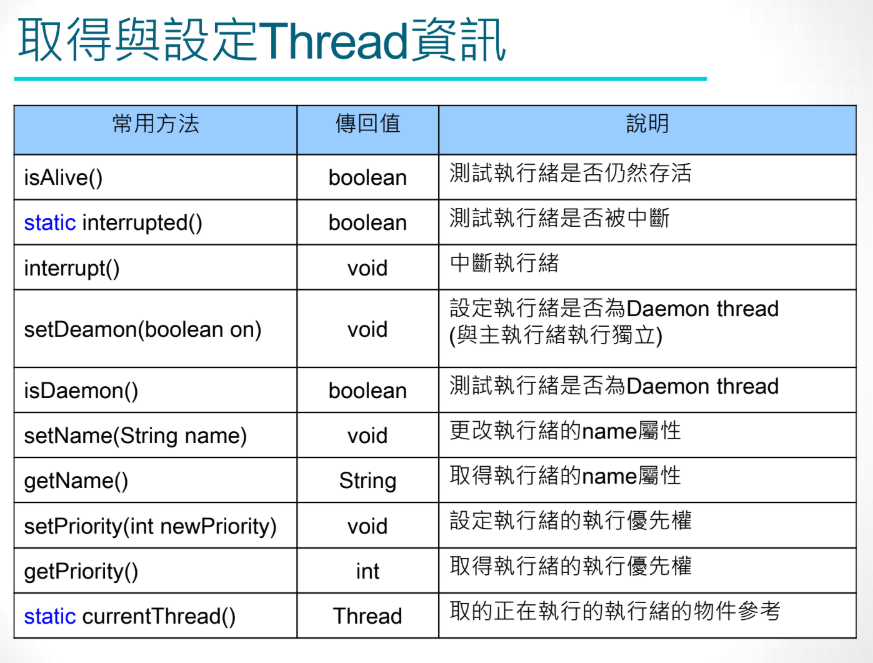
* Join()方法

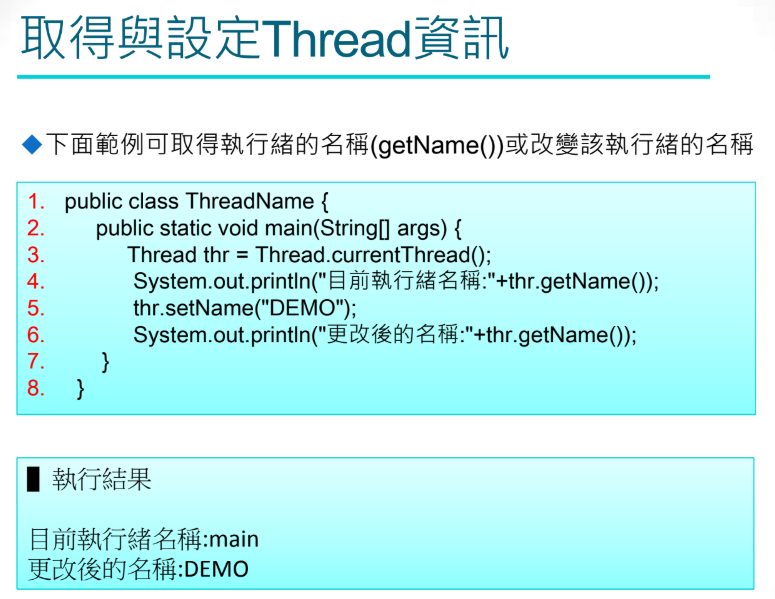












* 管理執行緒的優先權

→有鑑於先前測試發現執行緒運行順序不一定，需要用sleep或join等方法控制，但我們也可以設定執行緒的優先層級來控制執行的順序



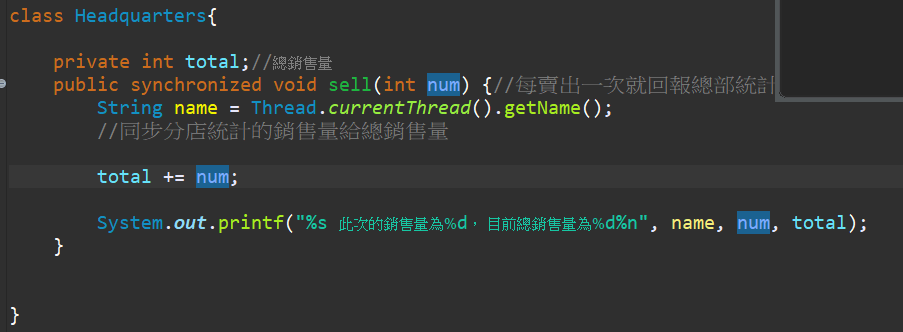
1. 資料同步機制(Synchronization)

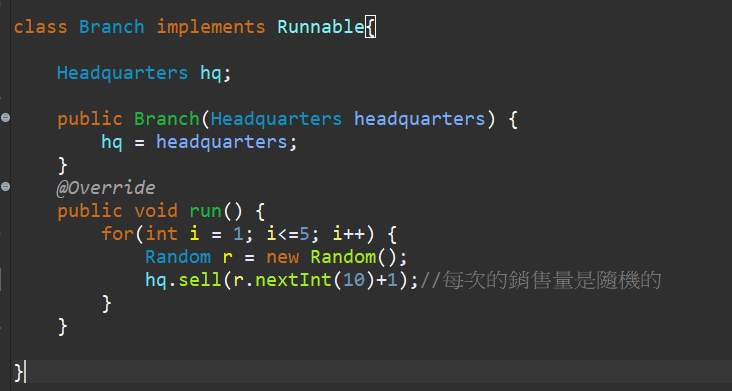
* 資料同步機制的出現是為了確保資料在進行傳輸時不會被其它執行緒干擾，而產生不符合情況的事情發生，例如要計算公司總銷售量時，需要在各分店回報完後才能加總，而不能邊回報邊加，這樣出來的資訊可能會是不正確的
* Synchronized關鍵字

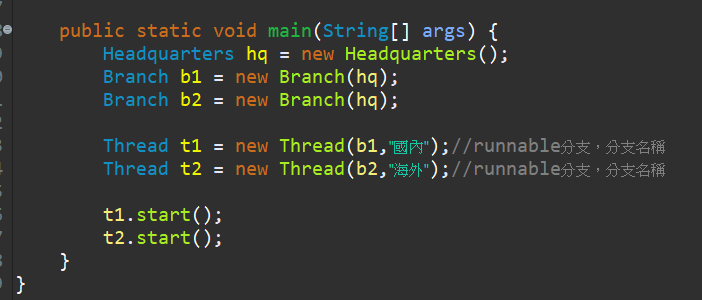
→用來保護某執行緒執行時，直到取到結果前都不會斷掉

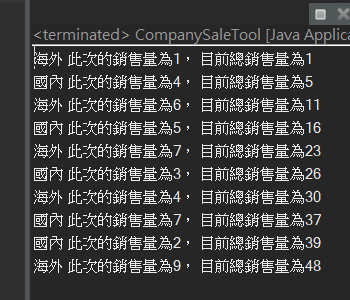


\*範例

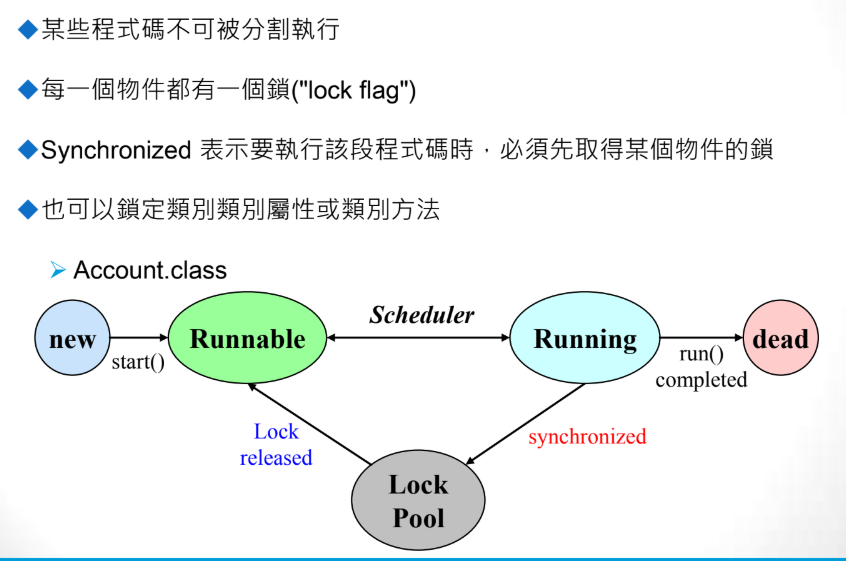




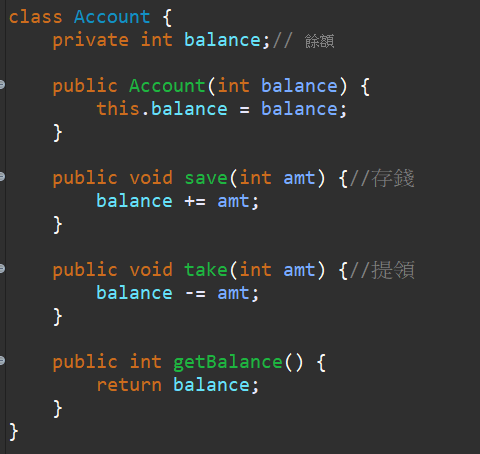


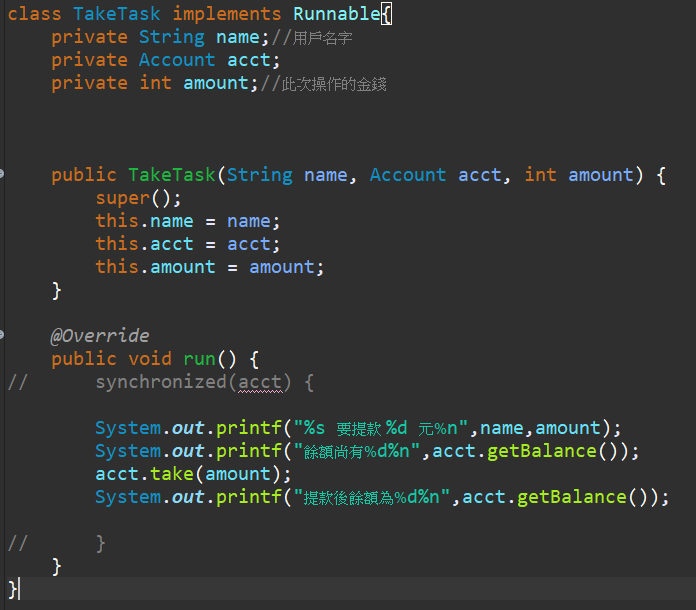


* 加入sychronized修飾字，可確保一地在回報完的時候，另一地才回報，加總出正確的銷售量

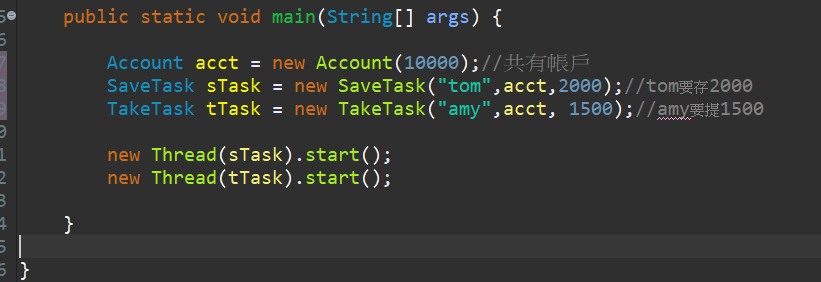


* 使用synchronized的重要性





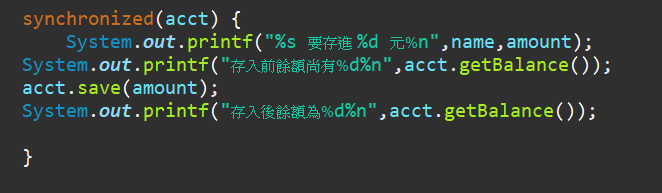


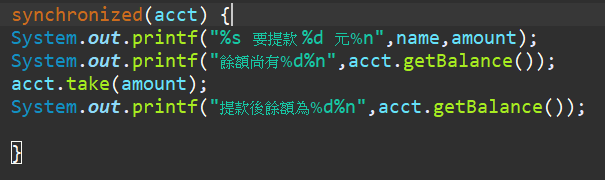


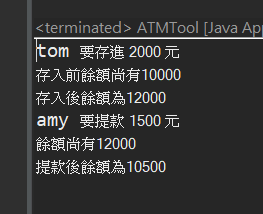


執行動作混亂，容易造成訊息不同步，也可能造成balance出錯等問題

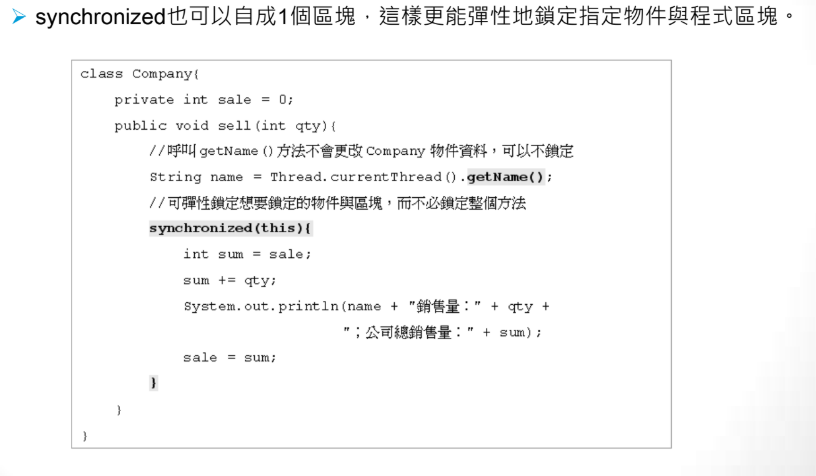
* 加入同步區塊，鎖定acct



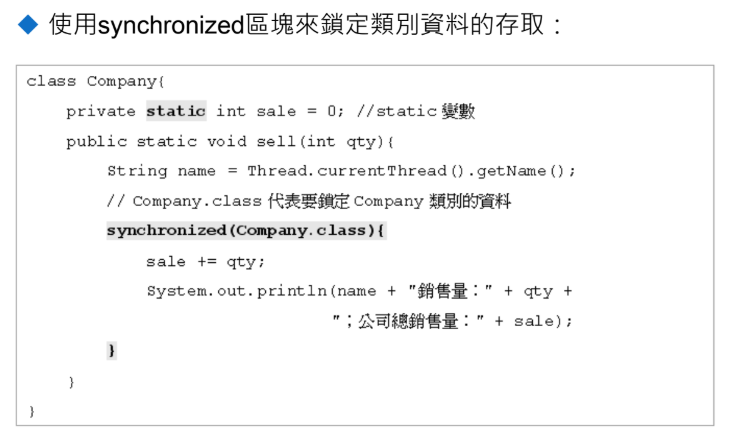




訊息整齊，共有資料不容易出錯！！

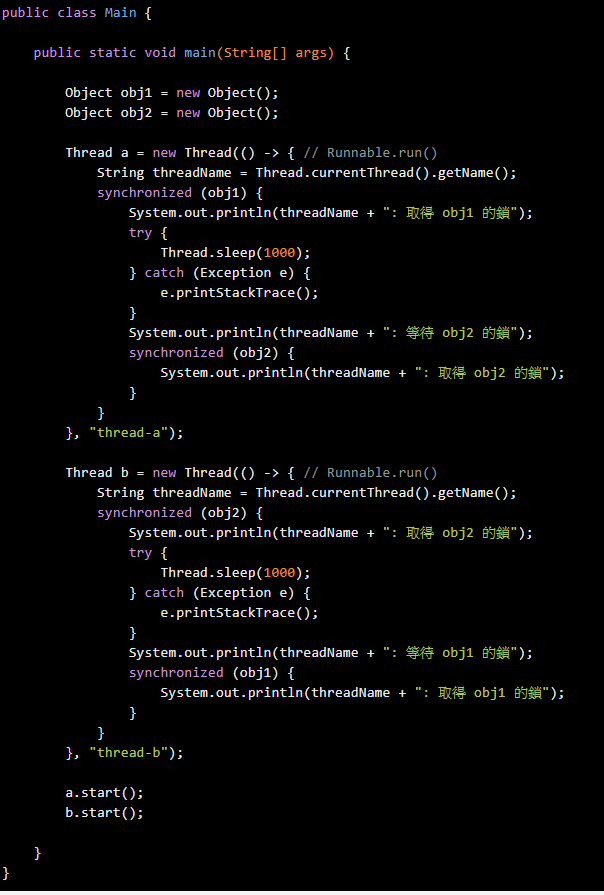


* synchronized可以鎖定物件資料，也可以鎖定static變數



* DeadLock(死結)現象

→兩條執行緒互相在等待對方釋放執行權後才執行，而後產生互相等待，互相都不執行的現象



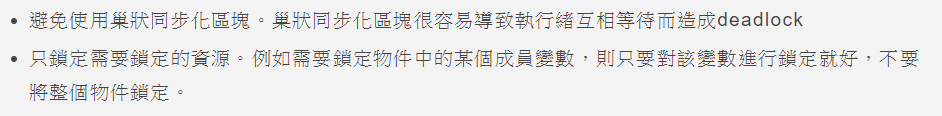
結果它在等obj2執行完

結果它在等obj1執行完

執行前會先檢查obj1是否執行完了

執行前會先檢查obj2是否執行完了

<https://matthung0807.blogspot.com/2017/10/java-deadlock.html>

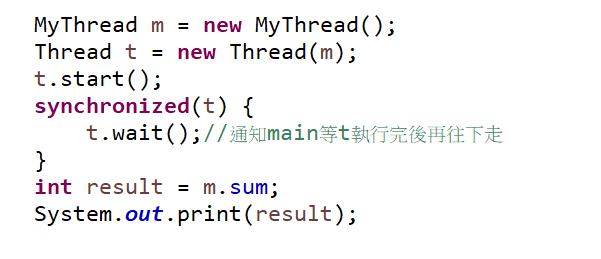


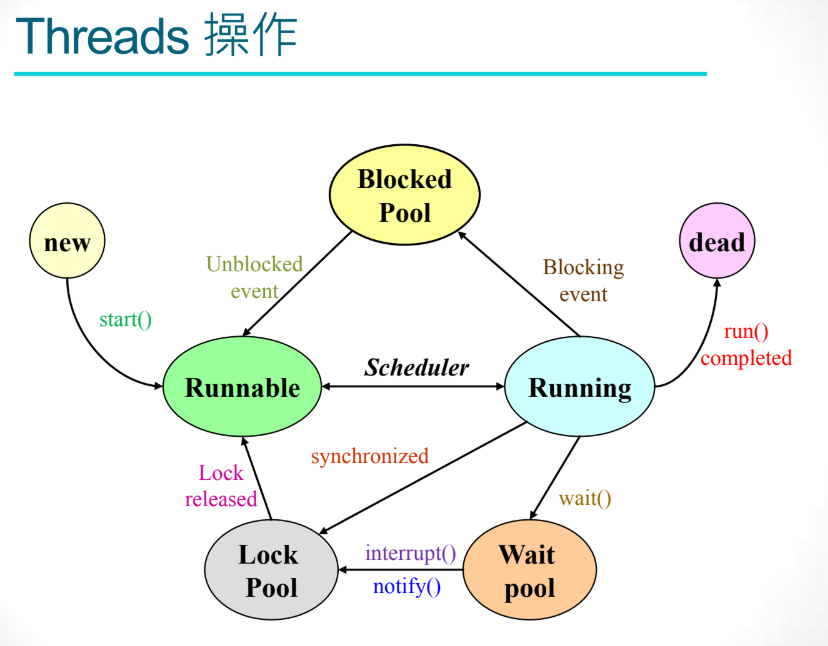
* Wait 與notify方法

→參考來源：<https://ithelp.ithome.com.tw/articles/10202999>

範例：建立另一條執行緒算出1加到100的總和







執行緒介紹

<http://dns2.asia.edu.tw/~wzyang/slides/Java_net/CH07.pdf>

結論

執行緒在程式中是一個不可分割的必要的觀念，會出現多執行緒有一個很大的重點就是為了不要影響主要任務線，因為這會影響用戶體驗(想像開個頁面讀圖片就卡頓幾十秒的感覺是很差的！)。所以許多耗時的操作就會需要多執行緒的協助，而某些操作也會乾脆直接不讓開發者寫在主執行緒上。

多執行緒的特性大概就像脫韁的野馬，除非跑完或是強制中斷，否則是不會停的，何時執行與誰先執行也沒有特定順序，但可以使用一些方法稍微控制一下，但基本這種野性是不會不見的，所以使用時要將這特性考慮進去，才能寫出符合需求的操作！