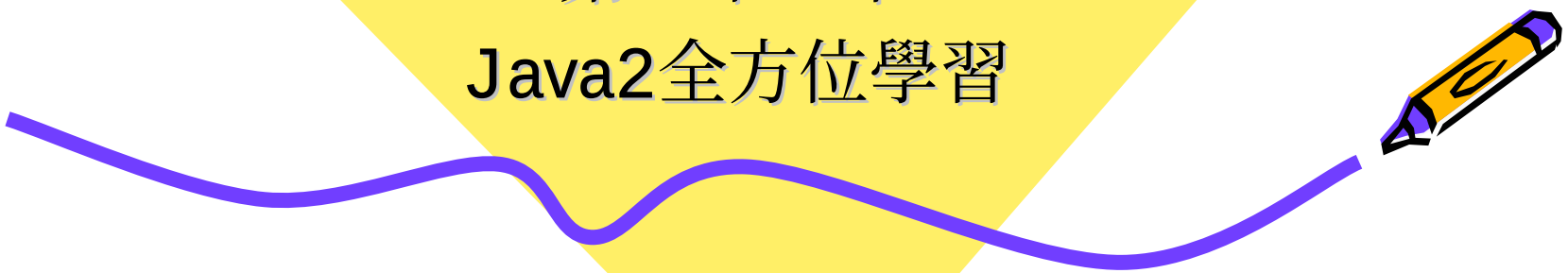
Threads

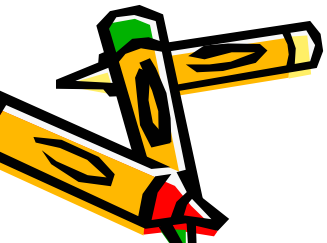
第二十二章

Java2全方位學習



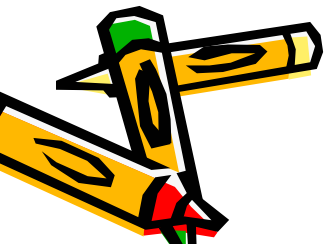
大綱

- 認識Threads
- 建立Threads
- Threads操作
- 資料同步處理
- 其它操作方法



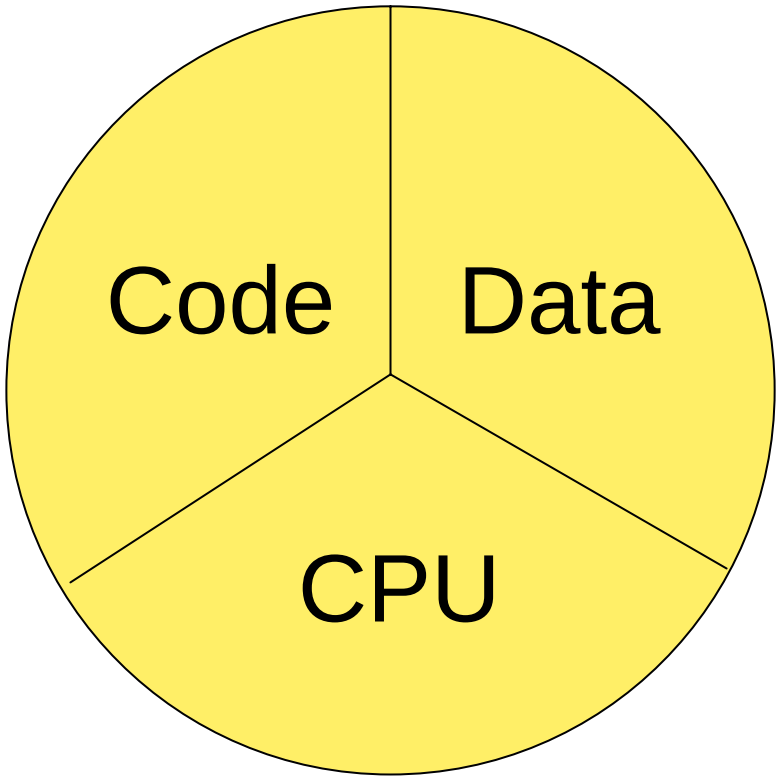
認識Threads

- Program、Process和Thread
- 多執行緒系統
  - 電腦在“同一時間內”執行好幾個程式
- 不同的Threads間可擁有自己的程式碼或資料，也可以跟別的threads共用



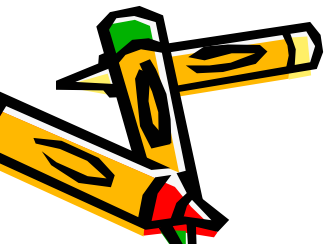
Thread的組成

- 一個虛擬的CPU
- CPU所要執行的程式碼(Code)
- 程式碼所要處理的資料(Data)



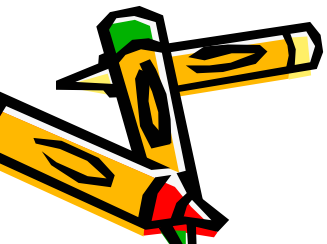
建立Thread物件

- java.lang.Thread類別
- 建構子
  - 傳入Runnable物件
- 例
  - Thread t1 = new Thread(r);



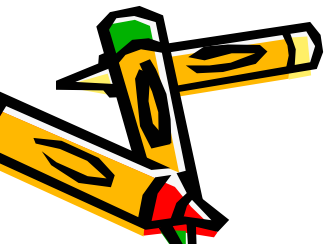
Runnable物件

- 方式
  - 實作Runnable介面
  - 繼承Thread類別
- 方法
  - run( )

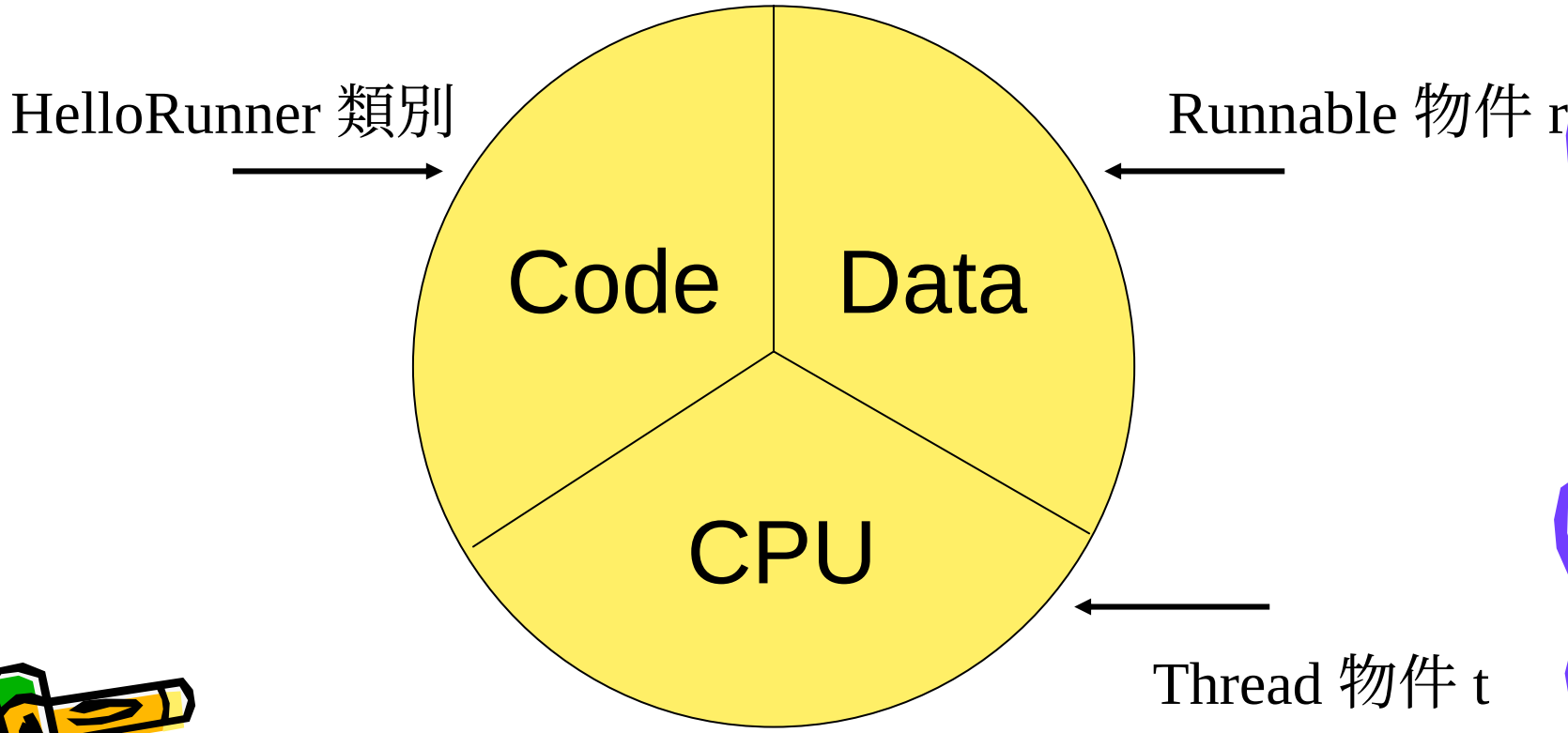


兩種方式比較

- 實作Runnable介面
  - 較符合物件導向的精神與設計架構
  - Java只能單一繼承，但能實作多個介面
  - 程式的一致性
- 繼承Thread類別
  - 程式碼較簡單

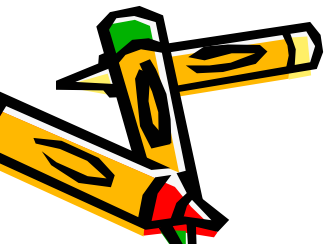


Thread物件

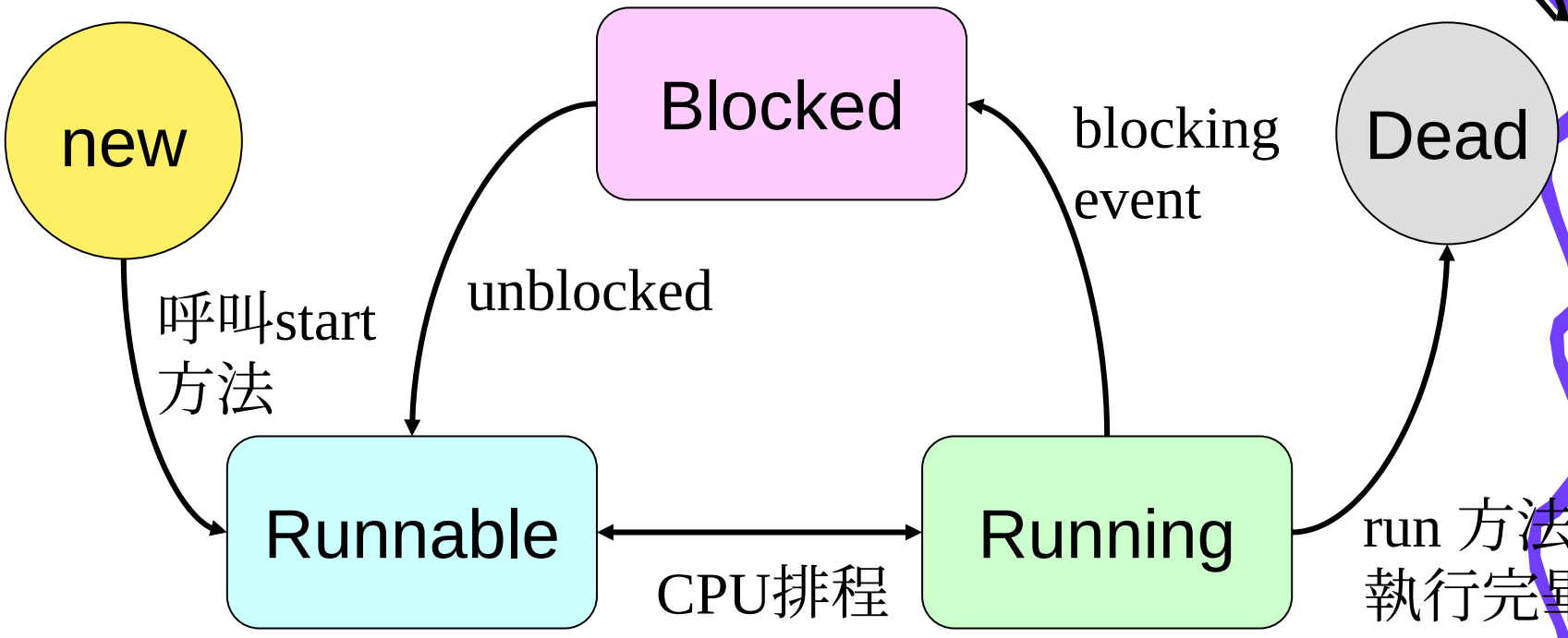


啟動Thread

- Thread類別的start方法
- Thread物件進入runnable狀態
  - 並沒有立刻被執行
- 主程式還是在執行
  - 主程式本身也是個thread



Thread生命週期狀態圖

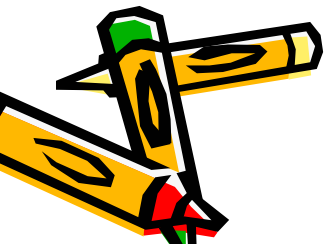


Thread狀態轉移

- CPU排程
  - 不同作業系統有不同的排程方式
  - Windows系統使用time-sliced
- Blocked狀態
  - 當thread在等待I/O處理時
  - Thread進入睡眠狀態
- Dead狀態
  - 當run方法執行完後
  - 一旦進入dead狀態，就無法再執行

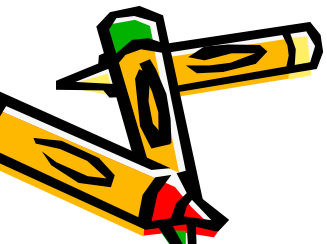
暫停Thread執行

- sleep方法
- join方法
- yield方法



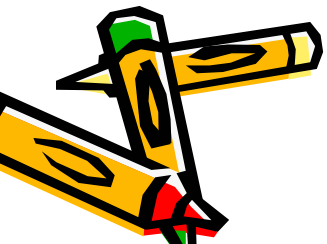
sleep方法

- Thread類別的sleep方法
- 以千分之一秒為單位
- 進入blocked狀態
  - 有可能會提前“睡醒”(interrupt)
  - Thread類別的interrupt方法
- “睡醒”後會進入runnable狀態



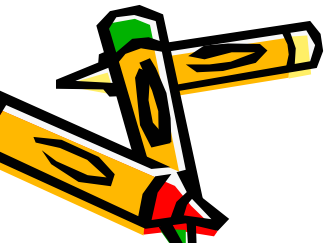
join方法

- 等待某thread執行完
- 可輸入等待時間
  - 千分之一秒為單位
- 進入blocked狀態



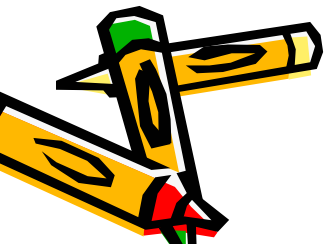
yield方法

- 放棄CPU執行權，讓別的thread執行
- 進去runnable狀態



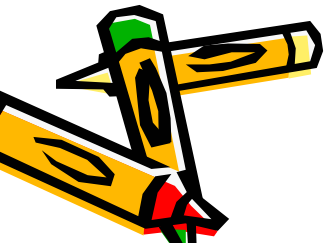
終止thread

- 不能使用stop方法
  - 已deprecated
- 結束run方法
  - 利用while迴圈加上條件控制



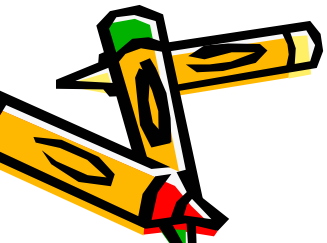
Thread基本操作方法

- currentThread( )
  - 取得現在執行的Thread物件
- isAlive( )
  - 檢查thread是否還“活”著



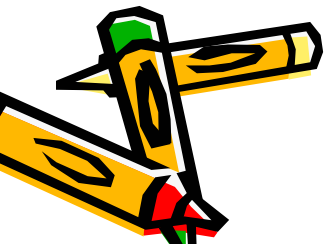
資料共享問題

- 多個thread存取同一段程式碼
  - 程式碼中存取某個共用的變數
- 此段程式碼是不可分割執行的



Synchronized修飾子

- 每個物件都有一個lock flag
- synchronized修飾子可使用於方法或程式區塊
- 要執行synchronized的方法或區塊時, 必需先取得該物件的lock flag
- 未取得lock flag則不能執行該段程式碼

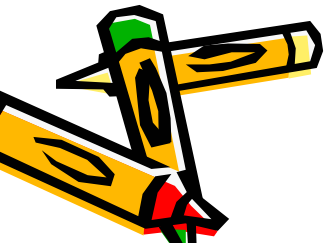


Synchronized使用方式

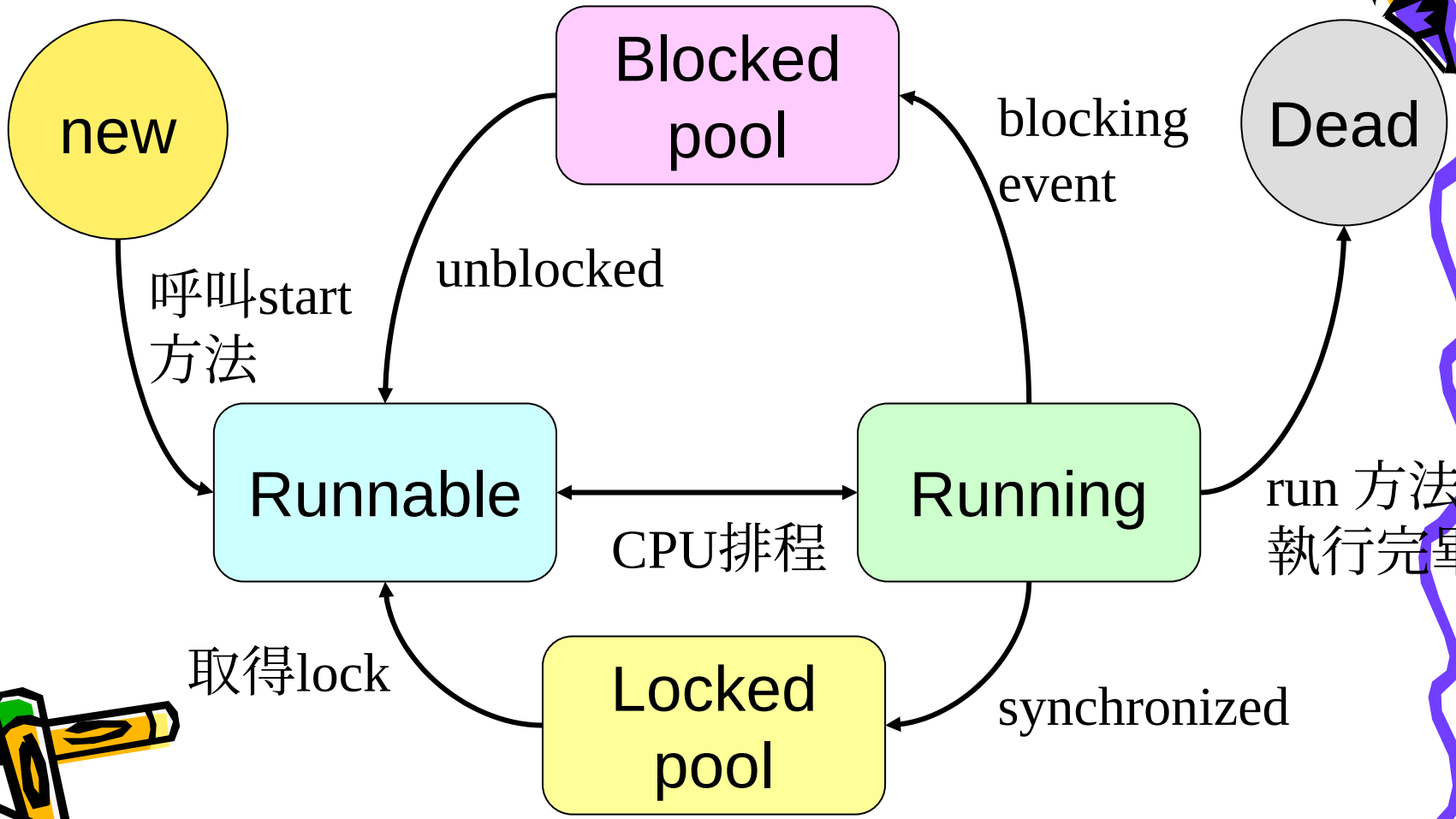
- 方法宣告
  - 整個方法中的程式碼都被鎖定
  - 可用於javadoc
  - 一些不需要保護的程式碼也被鎖定
    - 多餘的
- 程式區塊
  - syschronized (物件) { <要鎖定的程式碼> }
- 變數需宣告為private

Lock flag釋放

- 程式碼執行完畢
- break
- return
- exception



Synchronized狀態圖



死結(Deadlock)

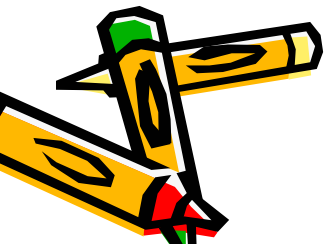
- 兩個以上的threads在互相等待對方所持有的lock flag
- 預防方法
  - 設定取得lock flag的順序
  - 嚴格遵守所設定的順序
  - 如果某個優先權較高的鎖沒辦法取得時，必需釋放掉自己手中所有的鎖。
  - 用相反的順序釋放lock flag

wait方法

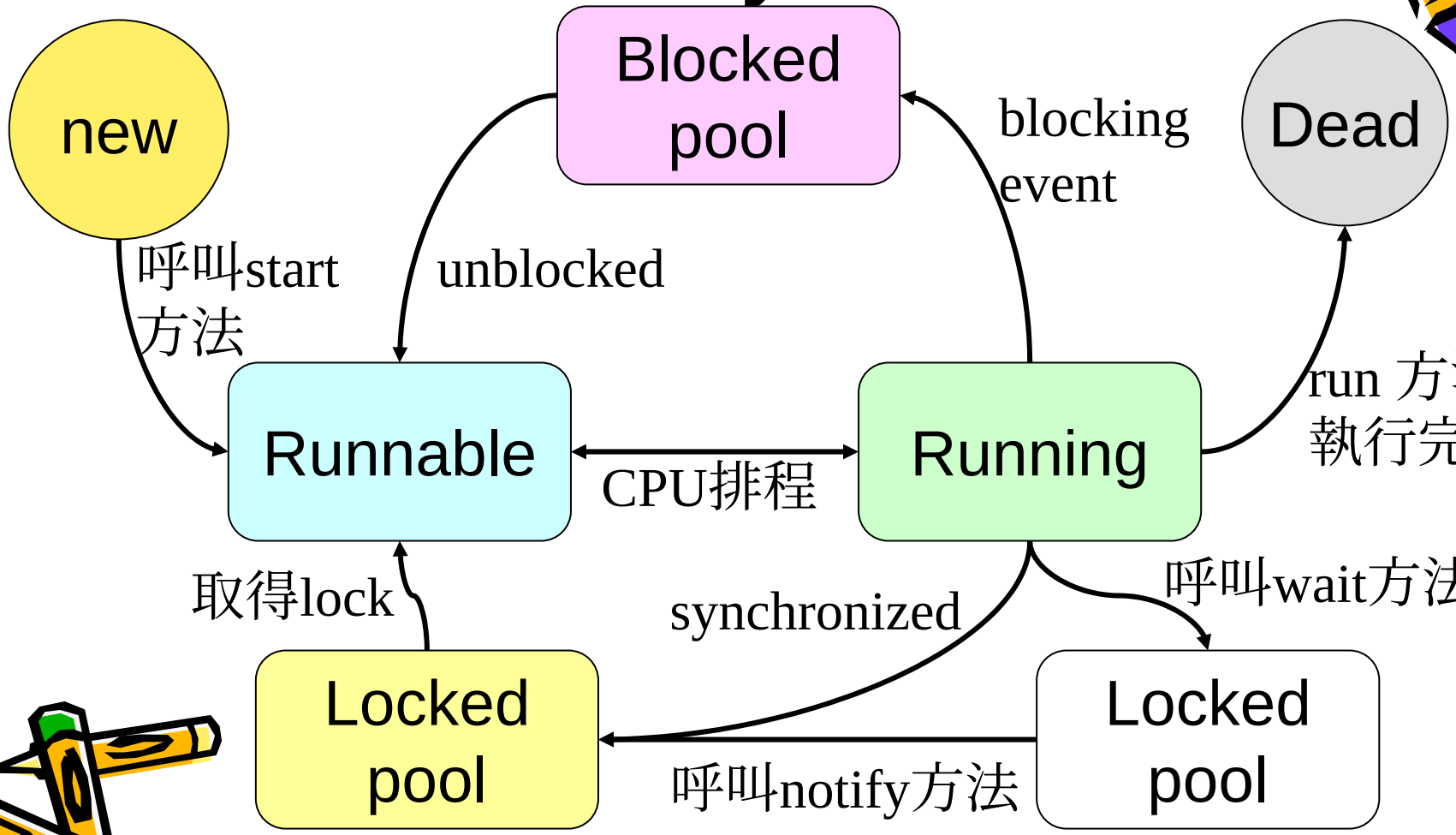
- java.lang.Object類別所提供的方法
- 當取得不到想要的lock時，可呼叫wait方法，讓自己進入wait pool
  - 必需在sychronized的程式區塊中才能呼叫
  - 進入wait pool時會釋放本身所有的lock
- 可設定wait的時間
  - 以千分之一秒為單位

notify方法

- java.lang.Object類別所提供的方法
- 從wait pool中任選一個thread進入lock pool
- 不能保証那個thread會被選到
- notifyAll方法可將wait pool中所有的thread都移到lock pool

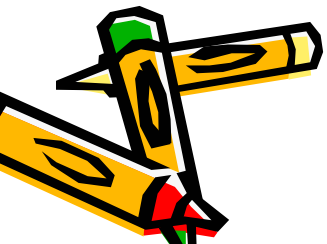


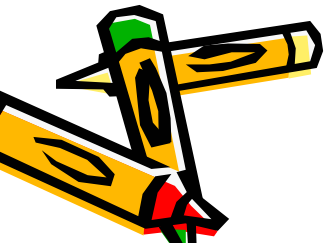
wait & notify狀態圖



Monitor Model

- 保證共用的資料會在一致的狀態
- 保證系統不會進入死結狀態





Q&A