

Thread 多執行緒

本章重點

- 何謂執行緒(Thread)
- Runnable介面與Thread類別
- Runnable範例
- 在應用程式中建立執行緒
- 執行緒常用的方法


何謂執行緒(Thread)

- 執行緒就像是CPU的分身，負責執行程式
- 要建立多執行緒前，必須先在特定類別的特定方法中準備好一段程式碼
- 一但開始執行某個執行緒，它就會有屬於自己的Stack記憶體，用來存放程式中的區域變數

執行緒的程式該寫在哪裡

- 設計一個類別實作Runnable介面
 - 覆寫run()方法: 執行緒需要的程式就從這裡開始
 - 這個設計較優，因為：
 - 是較佳的物件導向設計
 - 適合單一繼承的架構
 - 符合設計的一致性
- 設計一個類別繼承Thread類別
 - 覆寫run()方法: 執行緒需要的程式就從這裡開始
 - 這個設計只具有較簡單的程式碼，但不是好的物件導向設計

Runnable範例



```
class MyThreadRunner implements Runnable {
    int i, j; //member variable
    public void run() {
        while (true) {
            System.out.printf(
                "%s , Hello i=%d , j=%d%n",
                Thread.currentThread().getName()
                ,i ,j);

            i++; j++;
            if ( i == 50 ) {
                break;
            }
        }
    }
}
```

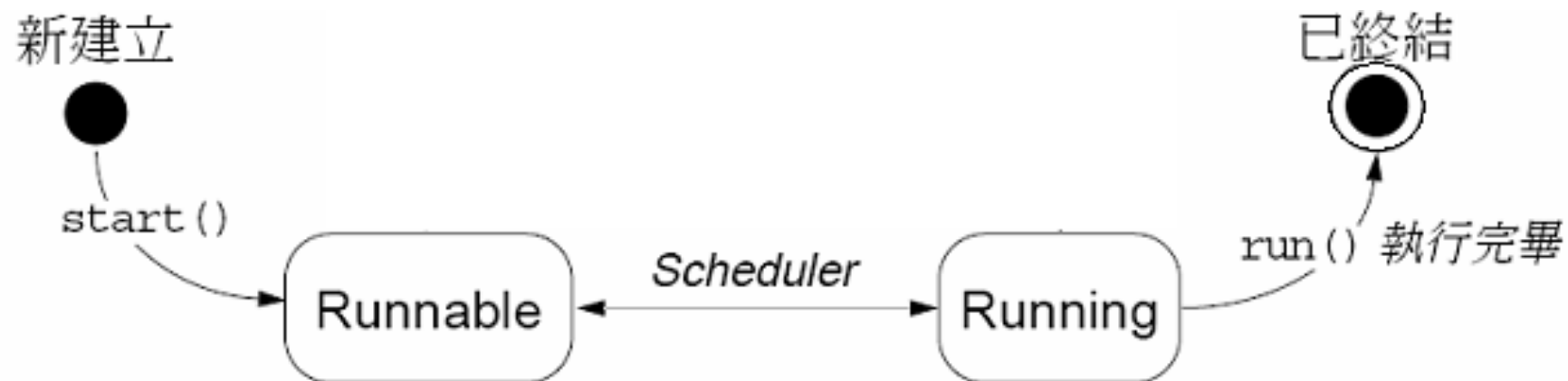
在應用程式中建立並啟動執行緒

建立Thread物件


- Thread t1 = new Thread(Runnable物件);
- Thread t2 = new MyThread();

呼叫Thread的start()來啟動執行緒

- t1.start();
- t2.start();



建立多執行緒程式範例

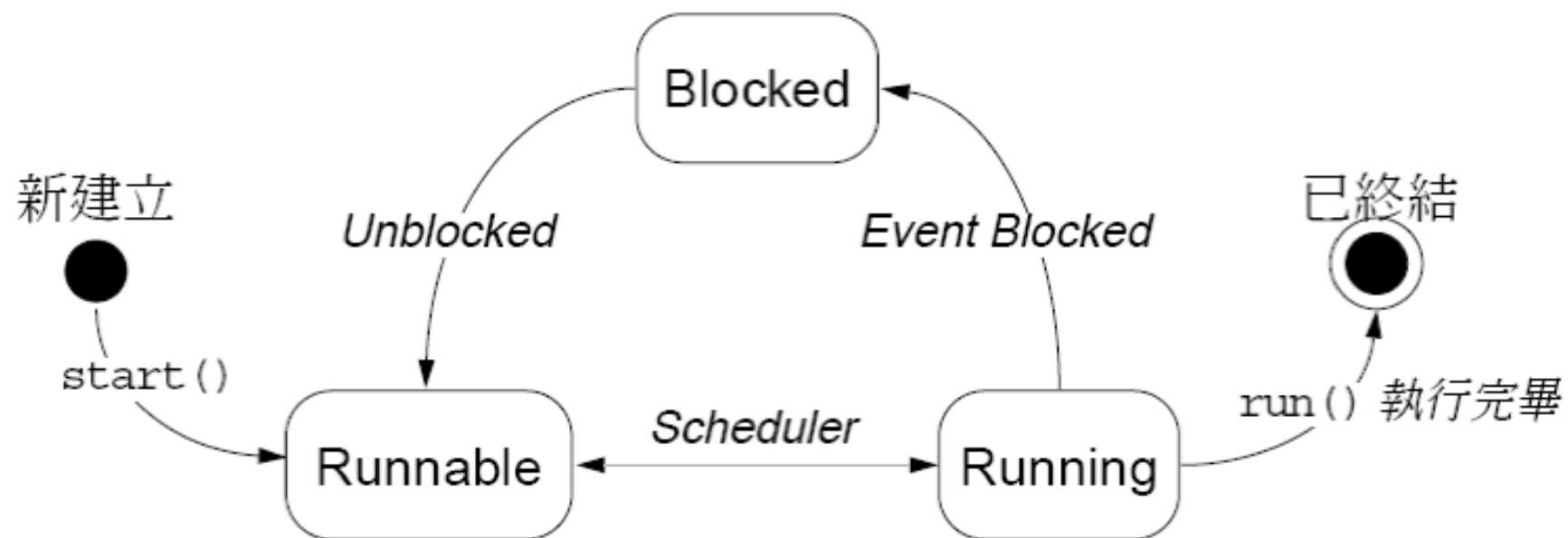


```
public class TestThread {  
    public static void main(String args[]) {  
  
        MyThreadRunner r = new MyThreadRunner();  
  
        Thread t = new Thread(r , "t");  
        t.start();  
  
        Thread t1 = new Thread(r, "t1");  
        t1.start();  
  
        System.out.println("The End");  
    }  
}
```

執行緒常用的方法

靜態方法，常用在run()方法中：

- Thread.currentThread()
- Thread.yield()
- Thread.sleep(time)



執行緒常用的方法(續)

非靜態方法，常常是寫在建立執行緒的程式中(如剛才示範的main方法中)：

- `getPriority()`
- `setPriority()`
- `isAlive()`
- `join()`