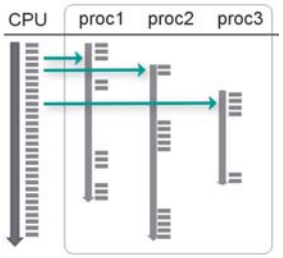
# 第十五章
# 多執行緒

本章學習目標
- 認識執行緒
- 學習如何建立執行緒
- 學習如何管理執行緒
- 認識執行緒的同步處理

## 認識執行緒(Thread)

- 傳統方法都是順著程式流程進行
- Java multi-thread則可以同時執行多個程序區塊，使的執行效率提高。
  - 例如：有些迴圈需要執行一段時間，如此即可啟動另一個執行緒來做其它處理。



## 15.1 認識執行緒

15-2

「多執行緒」的機制可以同時執行多個程式區塊。

app15_1是**單一執行緒的範例**：

```
01  // app15_1, 單一執行緒的範例
02  class CTest
03  {
04    private String id;
05    public CTest(String str)   // 建構元，設定資料成員id
06    {
07      id=str;
08    }
09    public void run()    // run() method
10    {
11      for(int i=0;i<4;i++)
12      {
13        for(int j=0;j<100000000;j++);  // 空迴圈，用來拖慢14行執行的速度
14          System.out.println(id+" is running..");
15      }
16    }
17  }
18
19  public class app15_1
20  {
21    public static void main(String args[])
22    {
23      CTest dog=new CTest("doggy");
24      CTest cat=new CTest("kitty");
25      dog.run();
26      cat.run();
27    }
28  }
```

```
/* app15_1 OUTPUT-------
doggy is running..
doggy is running..
doggy is running..
doggy is running..
kitty is running..
kitty is running..
kitty is running..
kitty is running..
-----------------------*/
```

第25行用dog物件呼叫run() method的執行結果

第26行用cat物件呼叫run() method的執行結果

15-4

# 啟動執行緒

要啟動執行緒，必須先準備好下列兩件事情：

1) 此類別必須是延伸自Thread類別，使自己成為它的子類別。
2) 執行緒的處理必須撰寫在run() method內。

15-5

## 要使一類別可啟動執行緒，必須用下列的語法來撰寫：

格式15.1.1執行緒之定義語法

```
class 類別名稱  extends Thread  // 從Thread類別延伸出子類別
{
      類別裡的資料成員;
      類別裡的method;
      修飾子 run() // 改寫Thread類別裡的run()method
       {
          以執行緒處理的程序;
       }
}
```

## 以上述的觀念來重新撰寫app15_1，使它可以同時啟動多個執行緒：

15-6

```
01  // app15_2, 啟動執行緒的範例
02  class CTest extends Thread  // 從Thread類別延伸出子類別CTest
03  {
04    private String id;
05    public CTest(String str)  // 建構元，設定成員id
06    {
07      id=str;
08    }
09    public void run()   // 改寫Thread類別裡的run() method
10    {
11      for(int i=0;i<4;i++)
12      {
13        for(int j=0;j<100000000;j++);// 空迴圈,用來拖慢14行執行的速度
14        System.out.println(id+" is running..");
15      }
16    }
17  }
19  public class app15_2
20  {
21      public static void main(String args[])
22    {
23      CTest dog=new CTest("doggy");
24      CTest cat=new CTest("kitty");
25      dog.start();  // 注意是呼叫start(),而不是run(),啟動執行緒
26      cat.start();  // 呼叫start(),而不是run(),啟動執行緒
27    }
28  }
```

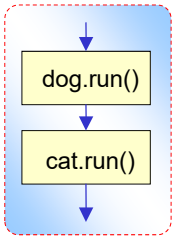
啟動執行緒是呼叫由Thread類別繼承而來的start(), 然後由start()在排程的scheduler排程器中登錄該執行緒。這個執行緒執行時，run()自然會被呼叫。

```
/* app15_2 OUTPUT----------
kitty is running..      第26行用cat物件呼叫start() method
doggy is running..      第25行用dog物件呼叫start() method
kitty is running..
kitty is running..      因為兩個執行緒一起執行，所以是交錯出現的
doggy is running..
kitty is running..
doggy is running..
doggy is running..
----------------------------------*/
```

15-8

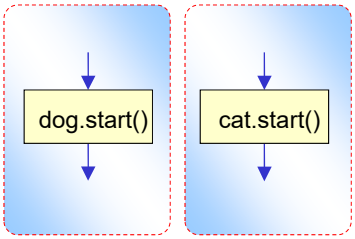
## 下圖為單一執行緒與兩個執行緒的執行流程比較：

單一執行緒




app15_1.java
執行完dog.run() 之後，
再執行cat.run()

兩個執行緒




app15_2.java
由start() 啟動執行緒，
因而dog和cat物件的run() 會同時被執行

# ex15_1_1.java

app15_2 可同時啟動兩個執行緒來執行。接續，試以下列的語法來建立物件，並利用pig 物件呼叫start() method，使其同時啟動3 個執行緒：

```
CTest pig=new CTest("piggy");
```

```
/* output----------
doggy is running..
kitty is running..
piggy is running..
piggy is running..
doggy is running..
kitty is running..
kitty is running..
piggy is running..
doggy is running..
piggy is running..
doggy is running..
kitty is running..
----------*/
```

# 15.1 回家作業
# 認識執行緒

hw15_1_1.java

(1)

a.建立一類別 MyThread，繼承Thread。其新建的類別內 容中為

```
private String x;
public MyThread(int x){
// turn to string
this.x = String.valueOf(x);
}
```

b.並在此類別中，加入一方法run()，可印出Hello I'm x，x為傳入的變數。

(2.)主程式中，new 5個Thread的物件，t1, t2, t3, t4, t5，依次傳入1, 2, 3, 4, 5，並依次呼叫 start()方法。

```
/* output----------
Hello I'm 1
Hello I'm 5
Hello I'm 3
Hello I'm 4
Hello I'm 2
----------*/
```

9

15-9

## 15.2 實作Runnable介面來建立執行緒

- ✓ 如果類別本身已經繼承了某個父類別(而Java無法多重繼承，因此無法再繼承執行緒)，這時可以利用實作Runnable介面的方式建立執行緒。
- ✓ Runnable介面裡宣告了抽象的run() method，因此把處理執行緒的程式碼放在run() 裡就可以建立執行緒。

下面的實例說明了Runnable介面的使用：

15-10

```
01  // app15_3,實作Runnable介面來建立執行緒
02  class CTest implements Runnable  // 由CTest類別實作Runnable介面
03  {
04    private String id;
05    public CTest(String str)  // 建構元，設定成員id
06    {
07      id=str;
08    }
09    public void run()  // 詳細定義runnable() 介面裡的 run() method
10    {
11      for(int i=0;i<4;i++)
12      {
13        for(int j=0;j<100000000;j++);// 空迴圈,用來拖慢14行執行的速度
14        System.out.println(id+" is running..");
15      }
16    }
17  }
19  public class app15_3
20  {
21   public static void main(String args[])
22    {
23      CTest dog=new CTest("doggy");
24      CTest cat=new CTest("kitty");
25      Thread t1=new Thread(dog);  // 產生Thread類別的物件t1
26      Thread t2=new Thread(cat);  // 產生Thread類別的物件t2
27      t1.start(); // 用t1啟動執行緒
28      t2.start(); // 用t2啟動執行緒
29    }
30  }
```

```
/* app15_3 OUTPUT-------
kitty is running..      第28行用t2物件呼叫run() method
doggy is running..      第27行用t1物件呼叫run() method
kitty is running..
kitty is running..
doggy is running..
kitty is running..
doggy is running..
doggy is running..
----------------------*/
```

# ex15_2_1.java

ex15_2_1.java

試設計**Add** 類別，其資料成員與建構元如下：

```
class Add{
private int n;
private int sum=0;
public Add(int a)
{  n=a;  }
}
```

```
/* output--------------
1+2+...+5=15
1+2+...+10=55
---------------------*/
```

請在**Add** 類別中加入可以計算**1+2+…+n** 的程式，並以多執行緒的方式，分別計算**1+2+…+5** 與**1+2+…+10** 的值。本程式請以實作**Runnable** 介面的方式建立執行緒。

12

15-14

## 新產生的執行緒

用new Thread() 建立物件時。

## 可執行的狀態

當start() method啟動執行緒時，執行緒便進入可執行的狀態。

## 被凍結的狀態

發生下列的事件時，凍結狀態的執行緒便產生：

1. 該執行緒呼叫物件的wait() method。
2. 該執行緒本身呼叫sleep() method。
3. 該執行緒和另一個執行緒join() 在一起。

## 銷毀的狀態

當run() method執行結束，或是由執行緒呼叫它的stop()method時。

15-15

### 15.3.2 讓執行緒小睡片刻

**下面是sleep() method的使用範例：**

```
01  // app15_4, sleep() method的示範
02  class CTest extends Thread  // 從Thread類別延伸出子類別
03  {
04    private String id;
05    public CTest(String str)  // 建構元，設定成員id
06    {
07      id=str;
08    }
09    public void run()   // 改寫Thread類別裡的run() method
10    {
11      for(int i=0;i<4;i++)
12      {
13        try
14        {
15          sleep((int)(1000*Math.random()));
16        }
17        catch(InterruptedException e){}
18        System.out.println(id+" is running..");
19      }
20    }
21  }
23  public class app15_4
24  {
25    public static void main(String args[])
26    {
27      CTest dog=new CTest("doggy");
28      CTest cat=new CTest("kitty");
29      dog.start();
30      cat.start();
31    }
32  }
```

sleep(1000) 睡一秒鐘
Math.random() 0~1間取亂數

sleep() method必須寫在try-catch區塊裡

```
/* app15_4 OUTPUT-------
kitty is running.. 第30行用cat物件呼叫start() method
doggy is running..第29行用dog物件呼叫start() method
kitty is running..
kitty is running..  每次執行結果不相同~
doggy is running..
kitty is running..
doggy is running..
doggy is running..
----------------------*/
```

# ex15_3_1.java

試利用Runnable 介面改寫app15_4。

```
/* output----------
doggy is running..
doggy is running..
kitty is running..
doggy is running..
kitty is running..
doggy is running..
kitty is running..
kitty is running..
----------*/
```

在兩個執行緒啟動之後，再加上一行列印字串的範例結果會怎樣呢？

15-17

```
// app15_5, 執行緒排程的設計(一)
class CTest extends Thread  // 從Thread類別延伸出子類別
{
  private String id;
  public CTest(String str) // 建構元，設定成員id
  {
    id=str;
  }
  public void run() // 改寫Thread類別裡的run() method
  {
    for(int i=0;i<4;i++)
    {
      try
      {
        sleep((int)(1000*Math.random()));
      }
      catch(InterruptedException e){}
      System.out.println(id+" is running..");
    }
  }
}

public class app15_5
{
  public static void main(String args[])
  {
    CTest dog=new CTest("doggy");
    CTest cat=new CTest("kitty");
    dog.start();        // 用dog物件來啟動執行緒
    cat.start();        // 用cat物件來啟動執行緒
    System.out.println("main() finished");
  }
}
```

main()也是一個執行緒，但因為main()不用經過執行緒的啟動程序，所以通常都先印出。(但原則上，上述三個執行緒，看誰先搶到資源則誰先印出)

**Thread(執行緒)：**

**一個 Process 裡面會有至少一個 Thread
**在 Java 中預設只有 main 一個
**可以建立很多 Thread 讓他們同時運作

```
/* app15_5 OUTPUT-------
main() method finished
doggy is running..
kitty is running..
doggy is running..
doggy is running..
doggy is running..
kitty is running..
kitty is running..
kitty is running..
----------*/
```

下面是執行緒排程設計的範例：

15-18

```
// app15_6, 執行緒排程的設計(一)
class CTest extends Thread   // 從Thread類別延伸出子類別
{
  private String id;
  public CTest(String str)    // 建構元，設定成員id
  {
    id=str;
  }
  public void run()       // 改寫Thread類別裡的run() method
  {
    for(int i=0;i<4;i++)
    {
      try
      {
        sleep((int)(1000*Math.random()));
      }
      catch(InterruptedException e){}
      System.out.println(id+" is running..");
    }
  }
}

public class app15_6
{
  public static void main(String args[])
  {
    CTest dog=new CTest("doggy");
    CTest cat=new CTest("kitty");

    dog.start();  // 啟動dog執行緒
    try
    {
      dog.join(); // 限制dog執行緒結束後才能往下執行
      cat.start();  // 啟動cat執行緒
      cat.join(); // 限制cat執行緒結束後才能往下執行
    }
    catch(InterruptedException e){}
    System.out.println("main() finished");
  }
}
```

sleep() 必須寫在try-catch區塊裡

join() 必須寫在try-catch區塊裡

```
/* app15_6 OUTPUT-------
doggy is running..
doggy is running..
doggy is running..
doggy is running..
kitty is running..
kitty is running..
kitty is running..
kitty is running..
main() method finished
----------------------*/
```

先執行dog執行緒

再執行 cat執行緒

最後再執行這一行敘述

# ex15_3_2.java

試利用Runnable 介面改寫app15_6。

```
/* output----------
doggy is running..
doggy is running..
doggy is running..
doggy is running..
kitty is running..
kitty is running..
kitty is running..
kitty is running..
main() method finished
----------------------*/
```

# 15.3回家作業 執行緒的管理

hw15_3_1.java

試撰寫一程式，由實作Runnable 介面的方式建立t1 與t2 兩個執行緒。t1 執行緒每隔1 秒便印出 "Thread 1 is running" 的字串，t2 執行緒每隔2.5 秒便印出 "Thread 2 is running" 字串，直到每個執行緒執行run() method 5 次為止。

```
/* output----------
Thread 1 is running..
Thread 1 is running..
Thread 2 is running..
Thread 1 is running..
Thread 1 is running..
Thread 2 is running..
Thread 1 is running..
Thread 2 is running..
Thread 2 is running..
Thread 2 is running..
----------*/
```

19

# 15.3 回家作業 執行緒的管理

hw15_3_2.java

試撰寫一程式，建立t1、t2 與t3 三個執行緒。t1 執行緒每隔1 秒便印出 2 個 "Thread1 is running" 字串，t2 執行緒每隔2.5 秒便印出5個 "Thread 2 is running" 字串，當t1 執行緒跑完後，t3 執行緒便會接著啟動，並且每隔3.5 秒印出3 個 "Thread 3 is running" 字串。

```
/* output----------
Thread 1 is running..
Thread 1 is running..
Thread 2 is running..
Thread 2 is running..
Thread 3 is running..
Thread 2 is running..
Thread 3 is running..
Thread 2 is running..
Thread 3 is running..
Thread 2 is running..
----------*/
```

20

15-20

# 15.4 同步處理

如果兩個執行緒共用一個變數，且其中一個執行緒在run() method還沒結束前，另一個執行緒已開始啟動，可能會造成錯誤，如下面的範例：

例如：有一家銀行，可以處理顧客匯款，每次匯款，可計算出匯款總額。現在有兩個顧客，每人都分3次匯款，每次匯款100元，最後銀行應該收到600元。

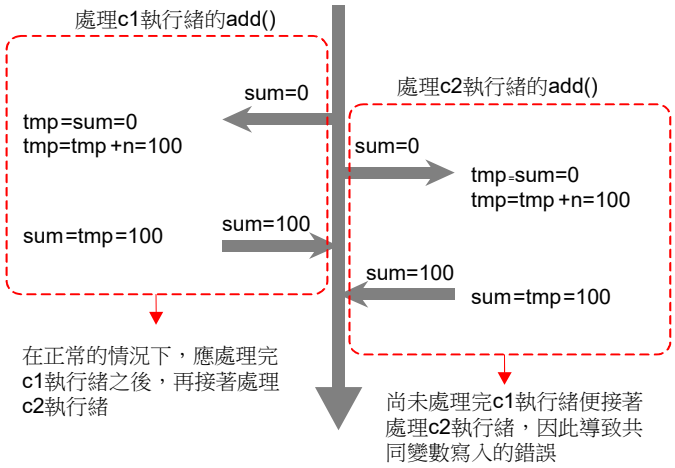
為了避免資料傳送塞車，增加睡眠裝置，讓每筆交易處理小睡0-1秒鐘，程式碼如下：

```
01 // app15_8, 沒有同步處理的執行緒
02 class CBank{
04   private static int sum=0;
05   public static void add(int n)
06   {
07     int tmp=sum;
08     tmp=tmp+n;  // 累加匯款總額
09     try
10     {
11       Thread.sleep((int)(1000*Math.random()));  // 小睡0~1秒鐘 0.56789*1000=567.89 567毫秒 0.567秒
12     }
13     catch(InterruptedException e){}
14     sum=tmp;
15     System.out.println("sum= "+sum);
16   } }
18 class CCustomer extends Thread // CCustomer,繼承Thread類別
19 {
20   public void run()     // run() method
21   {
22     for(int i=1;i<=3;i++)
23       CBank.add(100);    // 將300元分三次匯入
24   }
25 }
26 public class app15_7
27 {
28   public static void main(String args[])
29   {
30     CCustomer c1=new CCustomer();
31     CCustomer c2=new CCustomer();
32     c1.start();
33     c2.start();   } }
```

```
/* app15_7 OUTPUT----(沒有加synchronized的執行結果)
sum= 100
sum= 100
sum= 200
sum= 300
sum= 200
sum= 300
----------*/
```

15-22

## 下圖顯示了造成錯誤的原因：




15-23

**利用synchronized(同步化處理)關鍵字即可修正前敘的錯誤：**

```
05  public synchronized static void add(int n)
06  {
        ... ...
17  }
```

在add() method之前加上synchronized關鍵字

程式執行的結果應如下所示：

```
/* app15_7 OUTPUT----(加上synchronized的執行結果)
sum= 100
sum= 200
sum= 300
sum= 400
sum= 500
sum= 600
----------------------*/
```

**所以，若是有多個執行緒，又共同存取到同一個變數，則需要注意同步處理的重要性。**

# ex15_4_1.java

試利用Runnable 介面改寫app15_8，並加入Synchronized 關鍵字，使二執行緒同步。

```
/* output----------
sum= 100
sum= 200
sum= 300
sum= 400
sum= 500
sum= 600
----------*/
```

## 15.4 回家作業 同步處理

hw15_4_1.java

一個需要計算累加金額的變數 sum，但有兩個執行緒 p1 與 p2 同時共用。慈善捐款接受捐款，而每次會計算出總捐款額(sum)。

今日有兩位善心人士(p1,p2)，兩人每天(1次)都會捐150元，連續三天過後總捐款額若無他人捐款，那麼總額會是600元。

因應網路塞車等因素：可以加上睡眠，當每接受一次的捐款時，小睡0~1秒。

```
/* output----------
捐款總額= 150
捐款總額= 300
捐款總額= 450
捐款總額= 600
捐款總額= 750
捐款總額= 900
----------*/
```

26

# 數字累積20億次的電腦運算時間

```
import java.io.*;
public class Counter_before{
public static int count=0;
public static void main(String[] args) throws Exception{

long startTime=System.currentTimeMillis();//毫秒, 1秒=1000毫秒(ms)

for(int i=0; i<2000000000; i++)
 {
    count++;
 }
long endTime=System.currentTimeMillis();
System.out.println(endTime-startTime);
 }
}
```

1納秒=0.000001 毫秒
1納秒=0.00000 0001秒
System.currentTimeMillis()
System.nanoTime()

68

# 課堂練習

- ex15_1.java
- 撰寫一個程式，可以把數字累積20億次，接著，比較單工處理以及多工處理，其執行時間的差異。

- 作法：使用執行緒，讓工作同時分散進行，準備一個實做Runnable介面的類別，並產生物件實體，利用實做Runnable介面的物件，建立執行緒。

# 課堂練習

- ex15_2.java
- 請把數字20億次分派給4個執行緒工作，請比較單工處理，以及多執行緒的執行時間差異？