11.1 執行緒

到目前為止介紹過的各種範例都是單執行緒程式，也就是啟動的程式從 main()程式進入點開始至結束只有一個流程，有時候需要設計程式可以擁有多個流程，也就是所謂的多執行緒（Multi-thread）程式。

11.1.1 簡介執行緒

如果要設計一個龜兔賽跑遊戲，賽程長度為 10 步，每經過一秒，烏龜會前進一步，兔子則可能前進兩步或睡覺，那該怎麼設計呢？如果用目前所學過的單執行緒程式來說，你可能會如下設計：

刪除: 兔子

Thread TortoiseHareRace.java

```
package cc.openhome;

public class TortoiseHareRace {
    public static void main(String[] args) throws InterruptedException {
        boolean[] flags = {true, false};
        int totalStep = 10;
        int tortoiseStep = 0;
        int hareStep = 0;
        System.out.println("龜兔賽跑開始...");
        while(tortoiseStep < totalStep && hareStep < totalStep) {
            Thread.sleep(1000);
            tortoiseStep++;
            System.out.printf("烏龜跑了 %d 步...%n", tortoiseStep);
            boolean isHareSleep = flags[((int) (Math.random() * 10)) % 2];
            if(isHareSleep) {
                System.out.println("兔子睡著了 zzzz");
            } else {
                hareStep += 2;
                System.out.printf("兔子跑了 %d 步...%n", hareStep);
            }
        }
    }
}
```

❶ 由 JVM 處理例外

❷ 暫停 1000 毫秒，也就是 1 秒

❸ 烏龜走一步

❹ 隨機睡覺

❺ 兔子走兩步

刪除: 兔子

刪除: 兔子

目前程式只有一個流程，就是從 main()開始至結束的流程。要讓目前流程暫停指定時間，可以使用 **java.lang.Thread** 的靜態 **sleep()**方法，指定的單位是毫秒❷，呼叫這個方法必須處理 **java.lang.InterruptedException**，在這邊直接於 main()宣告 throws InterruptedException，由 JVM 處理此例外❶。

每次暫停一秒後，tortoiseStep 遞增 1 表示烏龜走一步❸，兔子則可能隨機睡覺❹，如果不是睡覺就將 hareStep 遞增 2，表示兔子走兩步❺，只要烏龜或兔子其中一個走完 10 步就離開迴圈，表示比賽結束。

由於程式只有一個流程，所以只能將烏龜與兔子的行為混雜在這個流程中撰寫，而且為什麼每次都先遞增烏龜再遞增兔子步數呢？這樣對兔子很不公平啊！如果可以撰寫程式再啟動兩個流程，一個是烏龜流程，一個兔子流程，程式邏輯會比較清楚。

在 Java 中，如果想在 main()以外獨立設計流程，可以撰寫類別實作 **java.lang.Runnable** 介面，流程的進入點是實作在 **run()**方法中。例如可以如下設計烏龜的流程：

Thread Tortoise.java

```
package cc.openhome;

public class Tortoise implements Runnable {
    private int totalStep;
    private int step;

    public Tortoise(int totalStep) {
        this.totalStep = totalStep;
    }

    @Override
    public void run() {
        try {
            while(step < totalStep) {
                Thread.sleep(1000);
                step++;
                System.out.printf("烏龜跑了 %d 步...%n", step);
            }
        } catch(InterruptedException ex) {
            throw new RuntimeException(ex);
        }
    }
}
```

在 Tortoise 類別中，烏龜的流程會從 run()開始，烏龜只要專心負責每秒走一步就可以了，不會混雜兔子的流程。同樣地，可以如下設計兔子的流程：

刪除: 兔子
刪除: 兔子
刪除: 兔子
刪除: 兔子
刪除: 兔子
刪除: 兔子
刪除: 兔子
刪除: 兔子

Thread Hare.java

```
package cc.openhome;

public class Hare implements Runnable {
    private boolean[] flags = {true, false};
    private int totalStep;
    private int step;

    public Hare(int totalStep) {
        this.totalStep = totalStep;
    }

    @Override
    public void run() {
        try {
            while(step < totalStep) {
                Thread.sleep(1000);
                boolean isHareSleep = flags[((int) (Math.random() * 10)) % 2];
                if(isHareSleep) {
                    System.out.println("兔子睡著了 zzzz");
                } else {
                    step += 2;
                    System.out.printf("兔子跑了 %d 步...%n", step);
                }
            }
        } catch(InterruptedException ex) {
            throw new RuntimeException(ex);
        }
    }
}
```

刪除: 兔子

在 Hare 類別中，兔子的流程會從 run()開始，兔子只要專心負責每秒睡覺或走兩步就可以了，不會混雜烏龜的流程。

刪除: 兔子

刪除: 兔子

在 Java 中，從 main()開始的流程會由主執行緒（Main thread）執行，那麼方才設計的 Tortoise 與 Hare，run()方法定義的流程該由誰執行呢？可以建構 **Thread** 實例來執行 **Runnable** 實例定義的 **run()**方法。例如：

Thread TortoiseHare2.java

```
package cc.openhome;

public class TortoiseHareRace2 {
    public static void main(String[] args) {
        Tortoise tortoise = new Tortoise(10);
        Hare hare = new Hare(10);
        Thread tortoiseThread = new Thread(tortoise);
        Thread hareThread = new Thread(hare);
```

```
        tortoiseThread.start();
        hareThread.start();
    }
}
```

在這個程式中，主執行緒執行 main()定義的流程，main()定義的流程中建立了 tortoiseThread 與 hareThread 兩個執行緒，這兩個執行緒會分別執行 Tortoise 與 Hare 的 run()定義的流程，要啟動執行緒執行指定流程，必須呼叫 Thread 實例的 **start()**方法。一個執行的範如下所示：

```
烏龜跑了 1 步...
兔子睡著了 zzzz
烏龜跑了 2 步...
兔子跑了 2 步...
烏龜跑了 3 步...
兔子跑了 4 步...
烏龜跑了 4 步...
兔子睡著了 zzzz
烏龜跑了 5 步...
兔子睡著了 zzzz
烏龜跑了 6 步...
兔子睡著了 zzzz
烏龜跑了 7 步...
兔子跑了 6 步...
烏龜跑了 8 步...
兔子跑了 8 步...
烏龜跑了 9 步...
兔子睡著了 zzzz
烏龜跑了 10 步...
兔子跑了 10 步...
```

刪除: 兔子

刪除: 兔子

刪除: 兔子

刪除: 兔子

刪除: 兔子

11.1.2 Thread 與 Runnable

從抽象觀點與開發者的角度來看，JVM 是台虛擬電腦，只安裝一顆稱為主執行緒的 CPU，可執行 main()定義的執行流程，如果想要為 JVM 加裝 CPU，就是建構 Thread 實例，要啟動額外 CPU 就是呼叫 Thread 實例的 start()方法。額外 CPU 執行流程的進入點，可以定義在 Runnable 介面的 run()方法中。

提示 »» 實際上 JVM 啟動後，不只有一個主執行緒，還會有垃圾收集、記憶體管理等執行緒，不過這是底層機制，就撰寫程式的角度來看 JVM，確實只有一個主執行緒。