

JAVA THREAD 3

030523313 - Network programming
Asst. Prof. Dr. Choopan Rattanapoka

ปัญหาที่ 1 : ในการใช้งาน Thread

- ในการทำงานจริงกับ Thread สำหรับ Application ทั่วไป และ Server Application การสร้าง Thread โดยไม่มีการจำกัดจำนวนอาจจะทำให้เครื่องคอมพิวเตอร์ที่เรียกใช้งาน โปรแกรมมีปัญหา

Thread แต่ละตัวจะใช้จองเนื้อ
ที่หน่วยความจำ 50 MB

จำลองให้ทำงาน 2 วินาที



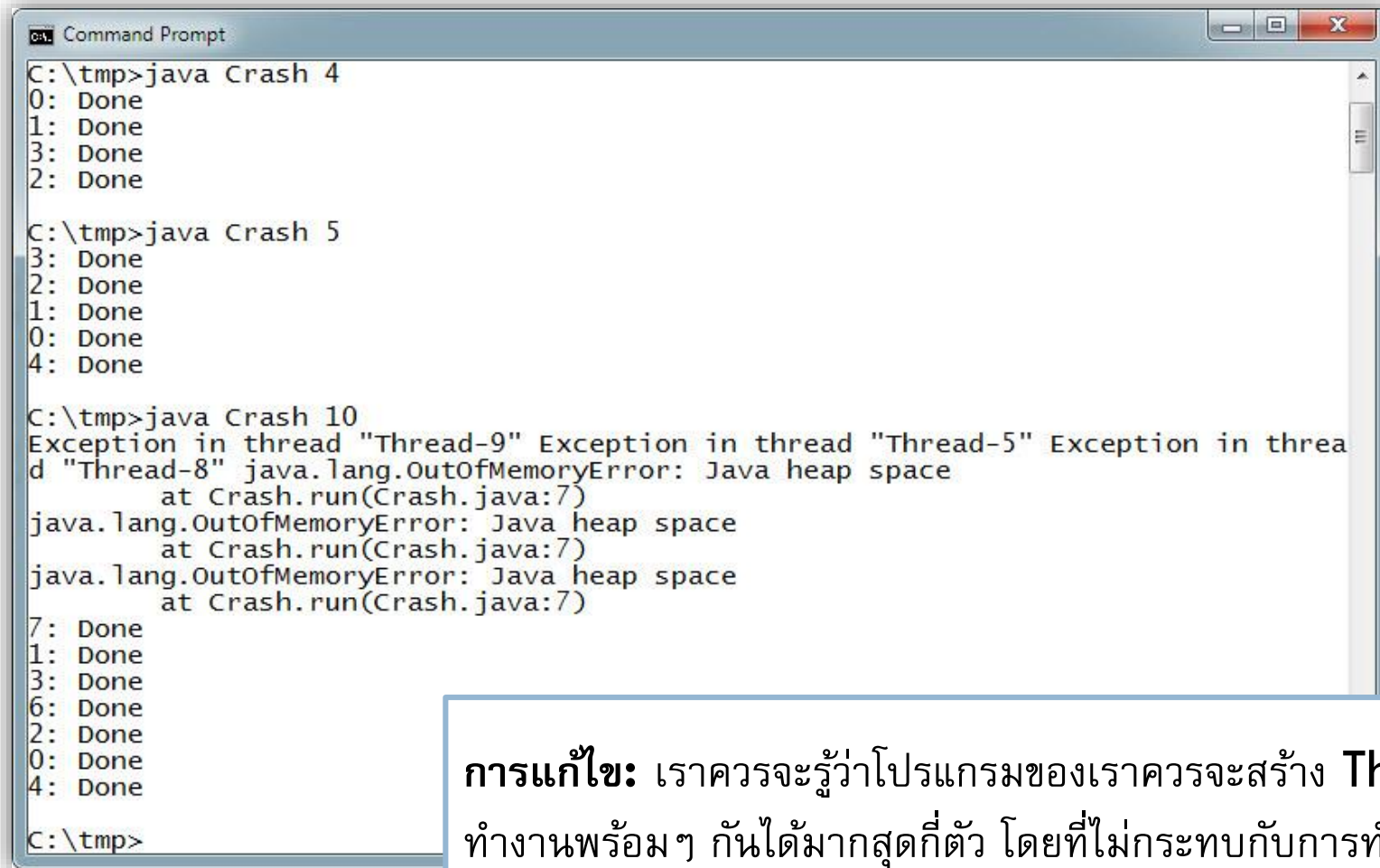
```
public class Crash implements Runnable {
    int id;

    public Crash(int id) { this.id = id; }

    public void run() {
        int buffer[] = new int[50000000];
        try {
            Thread.sleep(2000);
        } catch (Exception e) { }
        System.out.println(id + ": Done");
    }

    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for(int i = 0; i < n; i++) {
            Crash c = new Crash(i);
            Thread t = new Thread(c);
            t.start();
        }
    }
}
```

ปัญหาที่ 1: เรียกใช้งานโปรแกรม



```
Command Prompt
C:\tmp>java Crash 4
0: Done
1: Done
3: Done
2: Done

C:\tmp>java Crash 5
3: Done
2: Done
1: Done
0: Done
4: Done

C:\tmp>java Crash 10
Exception in thread "Thread-9" Exception in thread "Thread-5" Exception in thread "Thread-8" java.lang.OutOfMemoryError: Java heap space
    at Crash.run(Crash.java:7)
java.lang.OutOfMemoryError: Java heap space
    at Crash.run(Crash.java:7)
java.lang.OutOfMemoryError: Java heap space
    at Crash.run(Crash.java:7)
7: Done
1: Done
3: Done
6: Done
2: Done
0: Done
4: Done

C:\tmp>
```

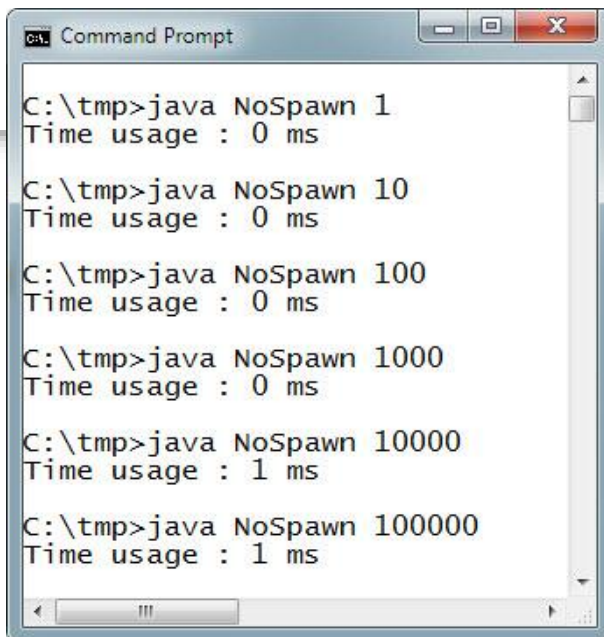
การแก้ไข: เราควรจะรู้ว่าโปรแกรมของเราควรจะสร้าง **Thread** ให้ทำงานพร้อมๆ กันได้มากที่สุดก็ตัว โดยที่ไม่กระทบกับการทำงานมากนัก

ปัญหาที่ 2 : ในการใช้งาน Thread

```
public class Spawn implements Runnable {  
    public void run() { }  
  
    public static void main(String[] args) {  
        int n = Integer.parseInt(args[0]);  
        long startTime = System.currentTimeMillis();  
        for(int i = 0; i < n; i++) {  
            Spawn s = new Spawn();  
            Thread t = new Thread(s);  
            t.start();  
            try {  
                t.join();  
            } catch(Exception e) { }  
        }  
        long stopTime = System.currentTimeMillis();  
        System.out.println("Time usage : " + (stopTime - startTime) + " ms");  
    }  
}
```

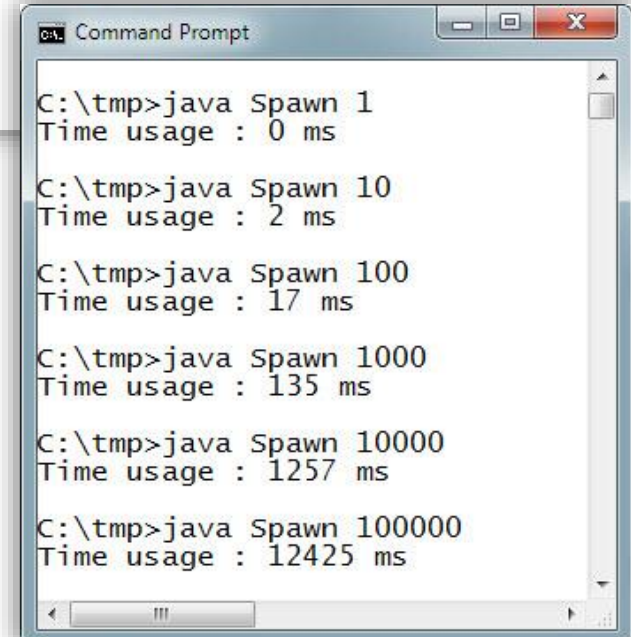
ปัญหาที่ 2: เรียกใช้งานโปรแกรม

```
public class NoSpawn {  
  
    public static void main(String[] args) {  
        int n = Integer.parseInt(args[0]);  
        long startTime = System.currentTimeMillis();  
        for(int i = 0; i < n; i++) { }  
        long stopTime = System.currentTimeMillis();  
        System.out.println("Time usage : " + (stopTime - startTime) + " ms");  
    }  
}
```



Command Prompt

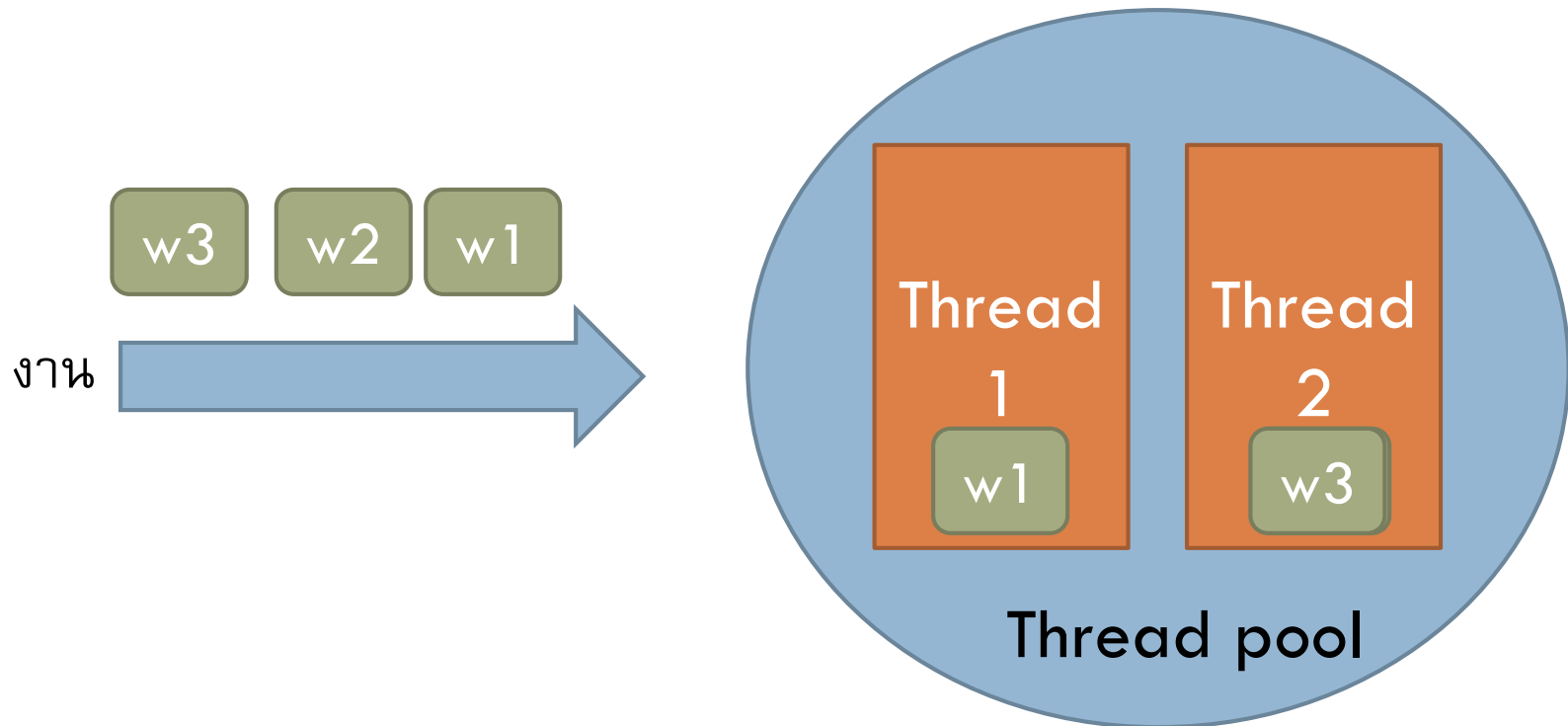
```
C:\tmp>java NoSpawn 1  
Time usage : 0 ms  
  
C:\tmp>java NoSpawn 10  
Time usage : 0 ms  
  
C:\tmp>java NoSpawn 100  
Time usage : 0 ms  
  
C:\tmp>java NoSpawn 1000  
Time usage : 0 ms  
  
C:\tmp>java NoSpawn 10000  
Time usage : 1 ms  
  
C:\tmp>java NoSpawn 100000  
Time usage : 1 ms
```



Command Prompt

```
C:\tmp>java Spawn 1  
Time usage : 0 ms  
  
C:\tmp>java Spawn 10  
Time usage : 2 ms  
  
C:\tmp>java Spawn 100  
Time usage : 17 ms  
  
C:\tmp>java Spawn 1000  
Time usage : 135 ms  
  
C:\tmp>java Spawn 10000  
Time usage : 1257 ms  
  
C:\tmp>java Spawn 100000  
Time usage : 12425 ms
```

การแก้ไขปัญห ด้วย Thread Pool



Java ใจดีมี Class มาให้ใช้งาน

- มี 2 class ที่ทำงานร่วมกันเพื่อทำงานในรูปแบบของ Thread Pool
 - ▣ **java.util.concurrent.Executors**
 - เป็น class ที่มี static method สำหรับสร้าง **ExecutorService**
 - เมธอด **newFixedThreadPool(int nThread)** สำหรับสร้าง **ThreadPool** ที่มี thread ภายในรอการทำงานจำนวน **nThread** ตัว
 - เมธอด **newSingleThreadExecutor()** สำหรับสร้าง **Threadpool** ที่มี **Thread** เดียว
 - ▣ **java.util.concurrent.ExecutorService**
 - เป็น class ที่ทำหน้าที่เป็น **ThreadPool**
 - เมธอด **execute(Runnable r)** ใช้สำหรับเอางาน **r** เข้าไปเข้าคิวรอการทำงาน
 - เมธอด **shutdown()** เมื่อต้องการหยุดการให้บริการ
 - เมธอด **isTerminated()** ตรวจสอบว่า **ExecutorService** ได้ปิดการทำงานเสร็จแล้วหรือไม่

แก้ไขปัญหานี้ 1

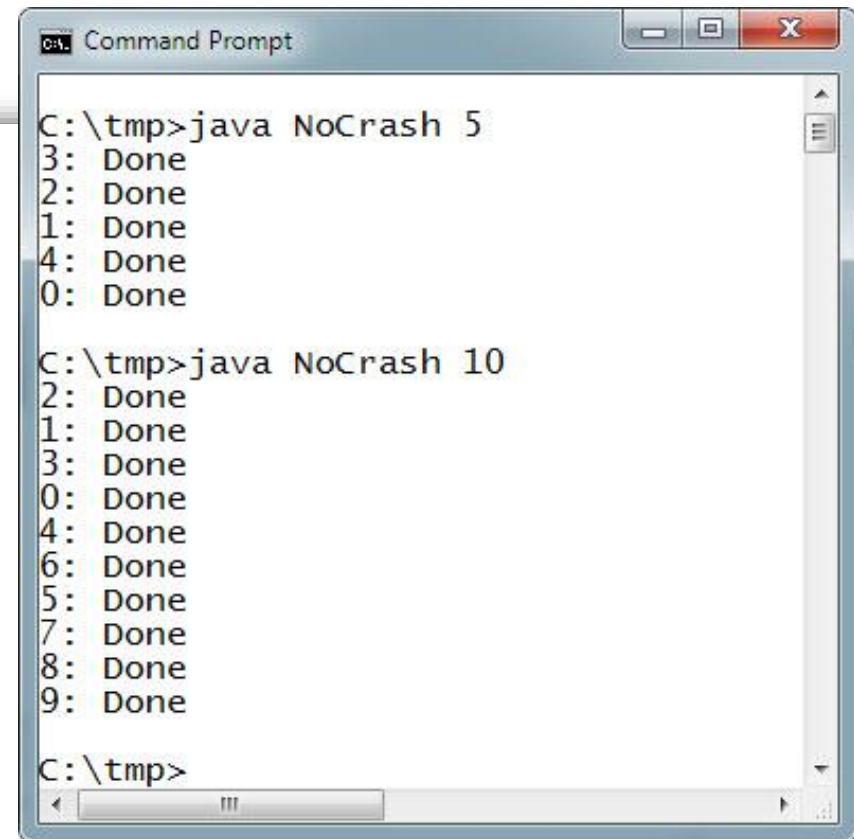
```
import java.util.concurrent.*;

public class NoCrash implements Runnable {
    int id;

    public NoCrash(int id) { this.id = id; }

    public void run() {
        int buffer[] = new int[50000000];
        try {
            Thread.sleep(2000);
        } catch (Exception e) { }
        System.out.println(id + ": Done");
    }

    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        ExecutorService es = Executors.newFixedThreadPool(5);
        for(int i = 0; i < n; i++) {
            NoCrash c = new NoCrash(i);
            es.execute(c);
        }
        es.shutdown();
    }
}
```



```
Command Prompt

C:\tmp>java NoCrash 5
3: Done
2: Done
1: Done
4: Done
0: Done

C:\tmp>java NoCrash 10
2: Done
1: Done
3: Done
0: Done
4: Done
6: Done
5: Done
7: Done
8: Done
9: Done

C:\tmp>
```


แก้ไขปัญหาคำที่ 2

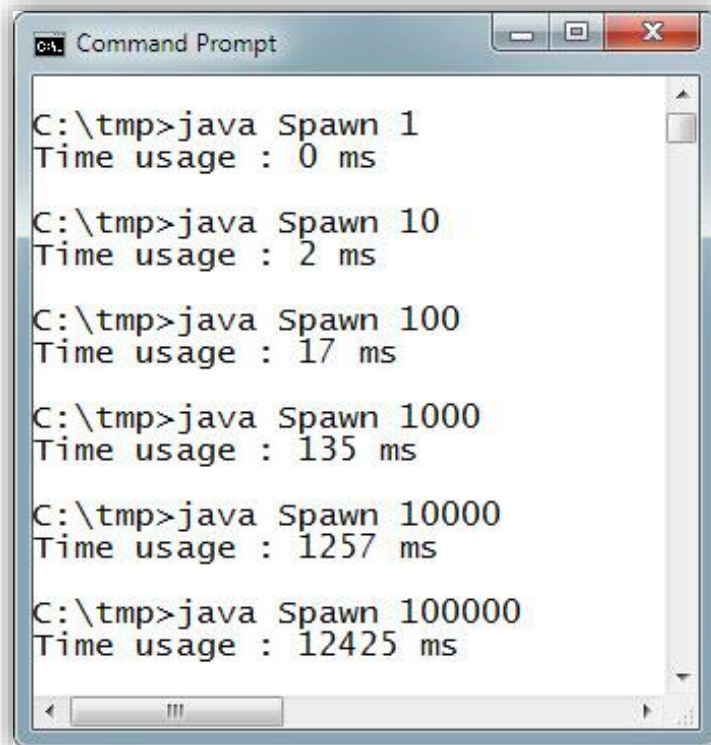
```
import java.util.concurrent.*;

public class SpawnThreadPool implements Runnable {
    public void run() { }

    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        long startTime = System.currentTimeMillis();

        ExecutorService executor = Executors.newSingleThreadExecutor();
        for(int i = 0; i < n; i++) {
            SpawnThreadPool s = new SpawnThreadPool();
            executor.execute(s);
        }
        executor.shutdown();
        while(!executor.isTerminated()) {}
        long stopTime = System.currentTimeMillis();
        System.out.println("Time usage : " + (stopTime - startTime) + " ms");
    }
}
```

ผลการรันเปรียบเทียบ



```
C:\tmp>java Spawn 1
Time usage : 0 ms

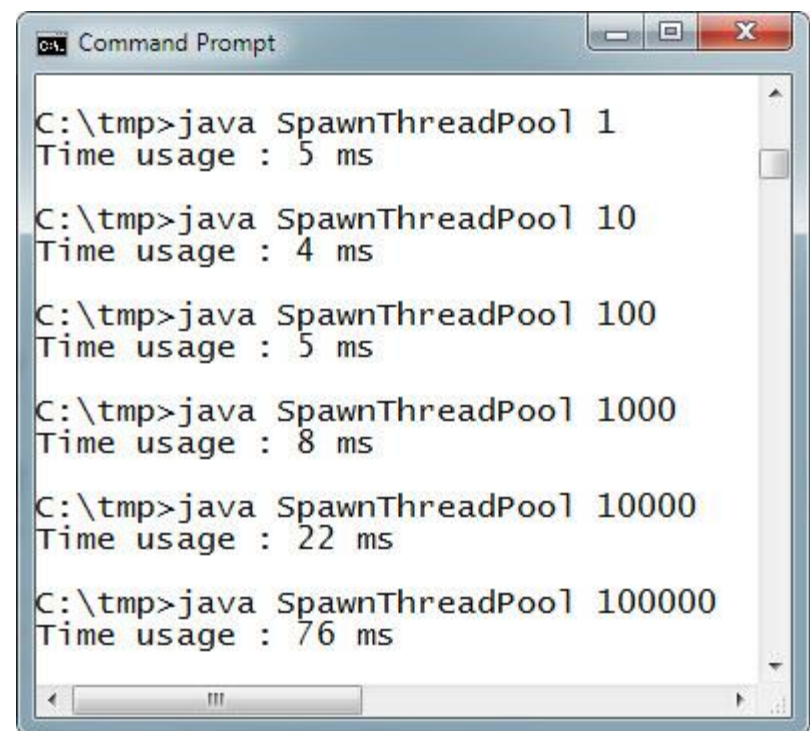
C:\tmp>java Spawn 10
Time usage : 2 ms

C:\tmp>java Spawn 100
Time usage : 17 ms

C:\tmp>java Spawn 1000
Time usage : 135 ms

C:\tmp>java Spawn 10000
Time usage : 1257 ms

C:\tmp>java Spawn 100000
Time usage : 12425 ms
```



```
C:\tmp>java SpawnThreadPool 1
Time usage : 5 ms

C:\tmp>java SpawnThreadPool 10
Time usage : 4 ms

C:\tmp>java SpawnThreadPool 100
Time usage : 5 ms

C:\tmp>java SpawnThreadPool 1000
Time usage : 8 ms

C:\tmp>java SpawnThreadPool 10000
Time usage : 22 ms

C:\tmp>java SpawnThreadPool 100000
Time usage : 76 ms
```