ใบงาน 7

วัตถุประสงค์ เพื่อศึกษาการใช้ thread ในภาษา java และปัญหาการใช้ข้อมูลร่วม

กิจกรรม

1. ศึกษาตัวอย่างโปรแกรมภาษาจาวา ตามภาพประกอบ ตาม requirement ต่อไปนี้

1.1 สร้าง thread โดยการ implement Runnable (บรรทัดที่ 15 และ 29) โดยกำหนดพฤติกรรมของ thread ใน public void run() (override void run())

1.2 thread t1 และ t2 จะเริ่มทำงานเมื่อเรียก .start() (ซึ่งจะไปเรียก .run() ดังนั้นหากกำหนด signature เป็นอย่างอื่นเช่น void run(int x) ก็จะไม่ถูกเรียก)

1.3 กลไกการใช้ข้อมูลร่วมกันที่ง่ายที่สุดคือ ให้ reference แก่ thread ทำให้ แต่ละ thread เห็น object (data) เดียวกัน (ถ้าต่างคนต่าง new L7_Obj() ก็เป็นคนละ object)

1.4 การจัดจังหวะที่ง่ายที่สุดคือให้ t1 และ t2 เรียก .join() เพื่อให้แม่รอให้ t1 และ t2 ทำงานเสร็จก่อน โดยการ join นั้นต้องทำใน try – catch block

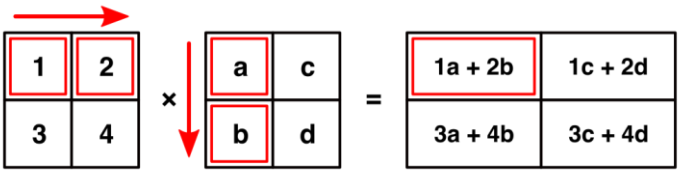
1.5 โปรแกรมนี้ถือว่ามี 3 thread โดย thread แรก คือ thread main (ตัวอย่างไม่ได้เรียก Thread.currentThread().getId() เพื่อแสดง thread_id ของ thread นั้นๆ ให้ดู)

```
/* 1*/ public class L7_Thread_demo1 {
/* 2*/   public static void main(String[] args) {
/* 3*/     L7_Obj x = new L7_Obj();
/* 4*/     Thread t1 = new Thread(new WorkerDemo1(x));
/* 5*/     t1.start();
/* 6*/     Thread t2 = new Thread(new WorkerDemo2(x));
/* 7*/     t2.start();
/* 8*/     try {
/* 9*/       t1.join();
/*10*/       t2.join();
/*11*/     } catch (Exception e) { System.out.println(e); }
/*12*/     System.out.println("from main x value is " +
                                x.getValue());
/*13*/  }
/*14*/ }
/*15*/ class WorkerDemo1 implements Runnable {
/*16*/   private L7_Obj o;
/*17*/   WorkerDemo1(L7_Obj y) { o = y; }
/*18*/   public void run() {
/*19*/     //o.inc(3);
/*20*/     System.out.println("From worker1 thread " +
                        Thread.currentThread().getId() +
                        " x value is " + o.getValue());
/*21*/   }
/*22*/ }
/*23*/ class L7_Obj {
/*24*/   private int value;
/*25*/   L7_Obj()       {  value = 0;   }
/*26*/   void inc(int x) {value += x;  }
/*27*/   int getValue()  { return value; }
/*28*/ }
/*29*/ class WorkerDemo2 implements Runnable {
/*30*/   private L7_Obj o;
/*28*/   WorkerDemo2(L7_Obj y) { o = y; }
/*28*/   public void run() {
/*29*/     o.inc(4);
/*30*/     System.out.println("From worker2 thread " +
                        Thread.currentThread().getId() +
                        " x value is " + o.getValue());
/*31*/   }
/*32*/ }
```

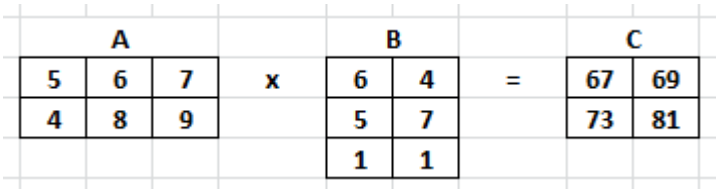
ตอบคำถามต่อไปนี้

1.6 หาก uncomment บรรทัดที่ 19 o.getValue() จากบรรทัดที่ 20 ตอบค่าอะไรได้บ้าง

2. การคูณเมตริกซ์ทำได้โดยหาผลบวกของการคูณสมาชิกในแต่ละแถวของเมตริกซ์ตัวตั้งกับแต่ละหลักของเมตริกซ์ตัวคูณ โดย ผลบวกของการคูณสมาชิกในแถวที่ 0 ของตัวตั้งกับหลักที่ 0 ของตัวคูณ จะได้ผลลัพธ์เป็นหลักที่ 0 และ คอลัมน์ที่ 0 ของ เมตริกซ์ผลลัพธ์ (ผลบวกของการคูณสมาชิกในแถวที่ 0 ของตัวตั้งกับหลักที่ 1 ของตัวคูณ จะได้ผลลัพธ์เป็นหลักที่ 0 และ คอลัมน์ที่ 1 และทำเช่นนี้ต่อไปเรื่อย ๆ) ด้วยแนวคิดนี้ทำให้เราสามารถประยุกต์แนวคิดของ Thread เข้ามาใช้งานเพื่อให้ได้ผลลัพธ์เร็วขึ้น (กรณีมีหลายหน่วยประมวลผล) โดยการสร้าง Thread หนึ่งตัวสำหรับหาผลบวกของการคูณในแต่ละแถวของตัวตั้งและแต่ละหลักของตัวคูณ (ขนาด p x q คูณกับ ขนาด q x r) โดยให้แม่รอจน Thread ลูก ทั้ง p x r ตัว คำนวณผลของตนใส่เมตริกซ์ผลลัพธ์ขนาด p x r แล้วจึงแสดงผล



1http://catlikecoding.com/unity/tutorials/rendering/part-1/



<u>คำสั่ง</u>

จงใช้แนวคิดนี้ในการเขียนโปรแกรม **Lab_MatrixMul.java** (Thread(0,0) ทำ C(0,0))

ตอบ Q1 -Q5

<u>กำหนดส่ง (TBA)</u>

```
/* 1*/ import java.util.Arrays;
/* 2*/ public class Lab_MatrixMul {
/* 3*/   public static void main(String[] args) {
/* 4*/     int[][] inputA = { { 5, 6, 7 }, { 4, 8, 9 } };
/* 5*/     int[][] inputB = { { 6, 4 }, { 5, 7 }, { 1, 1 } };
/* 6*/     MyData matA = new MyData(inputA);
/* 7*/     MyData matB = new MyData(inputB);
/* 8*/     int matC_r = matA.data.length;
/* 9*/     int matC_c = matB.data[0].length;
/*10*/     MyData matC = new MyData(matC_r, matC_c);
/*11*/     // Q4 construct 2D array for each "thread"
            with respected to each cell in matC,
            then start each thread
/*12*/     // Q5 join each thread
/*13*/       matC.show();
/*14*/     }
/*15*/ }
/*16*/ class MatrixMulThread implements Runnable {
/*17*/   int processing_row;     int processing_col;
/*18*/   MyData datA;    MyData datB;    MyData datC;
/*19*/   MatrixMulThread(int tRow, int tCol,
                MyData a, MyData b, MyData c) {
/*20*/     // Q1 code here
/*21*/   }
/*23*/   /* Q2 */ void run() {
/*24*/     // Q3
           // System.out.println("perform sum of
                multiplication of assigned row and col");
/*25*/   }
/*26*/ } //class
/*27*/ class MyData {
/*28*/   int[][] data;
/*29*/   MyData(int[][] m) {  data = m;    }
/*30*/   MyData(int r, int c) {
/*31*/     data = new int[r][c];
/*32*/     for (int[] aRow : data)
/*33*/         Arrays.fill(aRow, 9);
             // 9 will be overwritten anyway
/*34*/   }
/*35*/   void show() {
           System.out.println(Arrays.deepToString(data));
/*36*/   }
/*37*/ } //class
```

วิธีหา ขนาดของ matrix