นาย ชุติโชค พิทยจำรัส 5907101009

**แบบฝึกหัด บทที่ 4**

**1. (5 คะแนน) เรื่อง User-Level Thread และ Kernel-Level Thread**

* **จงอธิบายความแตกต่างระหว่าง User-Level Thread และ Kernel-Level Thread**

- ใน Kernel Level ตัว kenel จะเห็นการทำงาน ของทั้งตัว process และ thread ทั้งหมด ในขณะที่ User Level จะเห็นเพียงตัว process อีกทั้ง Kernel thread จะทำงานได้ช้ากว่า เนื่องจาก ต้องถูกจัดการตามลำดับ ตามที่ OS กำหนดไว้

* **สถานการณ์ใดที่เหมาะสมกับการใช้ User-Level Thread และ Kernel-Level Thread**

- User-Level Thread เหมาะกับ Many-to-One คือ Kernel thread 1 หน่วย กับ User thread หลายหน่วย

- Kernel-Level Thread เหมาะกับ One-to-One คือ Kernel thread 1 หน่วย กับ User thread 1 หน่วย

* **สถานการณ์ใดที่เหมาะสมกับการใช้ Kernel-Level Thread**

- เหมาะกับ One-to-One

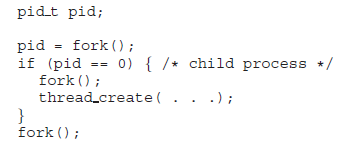
**2.** **(5 คะแนน)** **สถานการณ์ใดที่ Multithreaded Solution ที่ใช้ Multiple Kernel Thread ทำให้ระบบมีประสิทธิภาพดีกว่าการใช้ Single-Threaded Solution บนระบบที่มีโพรเซสเซอร์เดียว จงยกตัวอย่างพร้อมคำอธิบาย**

- ในการทำงานหากมี thread ใดตาย ก็จะไม่กระทบกับการทำงาน Thread หรือ Request อื่นๆ เพราะแต่ละ thread มีการทำงานแยกกัน ถ้าเป็น single thread หาก thread ตาย อาจทำให้ Request อื่นๆไม่สามารถ Process ต่อได้หรืออาจกระทบกับทั้งระบบให้ไม่สามารถรับ Request ต่อๆไปได้

**3. (5 คะแนน)** **สิ่งใดต่อไปนี้ Register Values , Heap Memory , Global Variables , Stack Memory ที่ถูกแชร์ระหว่าง Multithreaded Process**

- Global Variables และ Heap Memory

**4. (5 คะแนน) จากส่วนของโค้ดต่อไปนี้**



* **มีโปรเซสที่สร้างขึ้นกี่โปรเซส**

- 6 โปรเซส

* **มี Thread ที่สร้างขึ้นกี่ Thread**

- 8 Thread

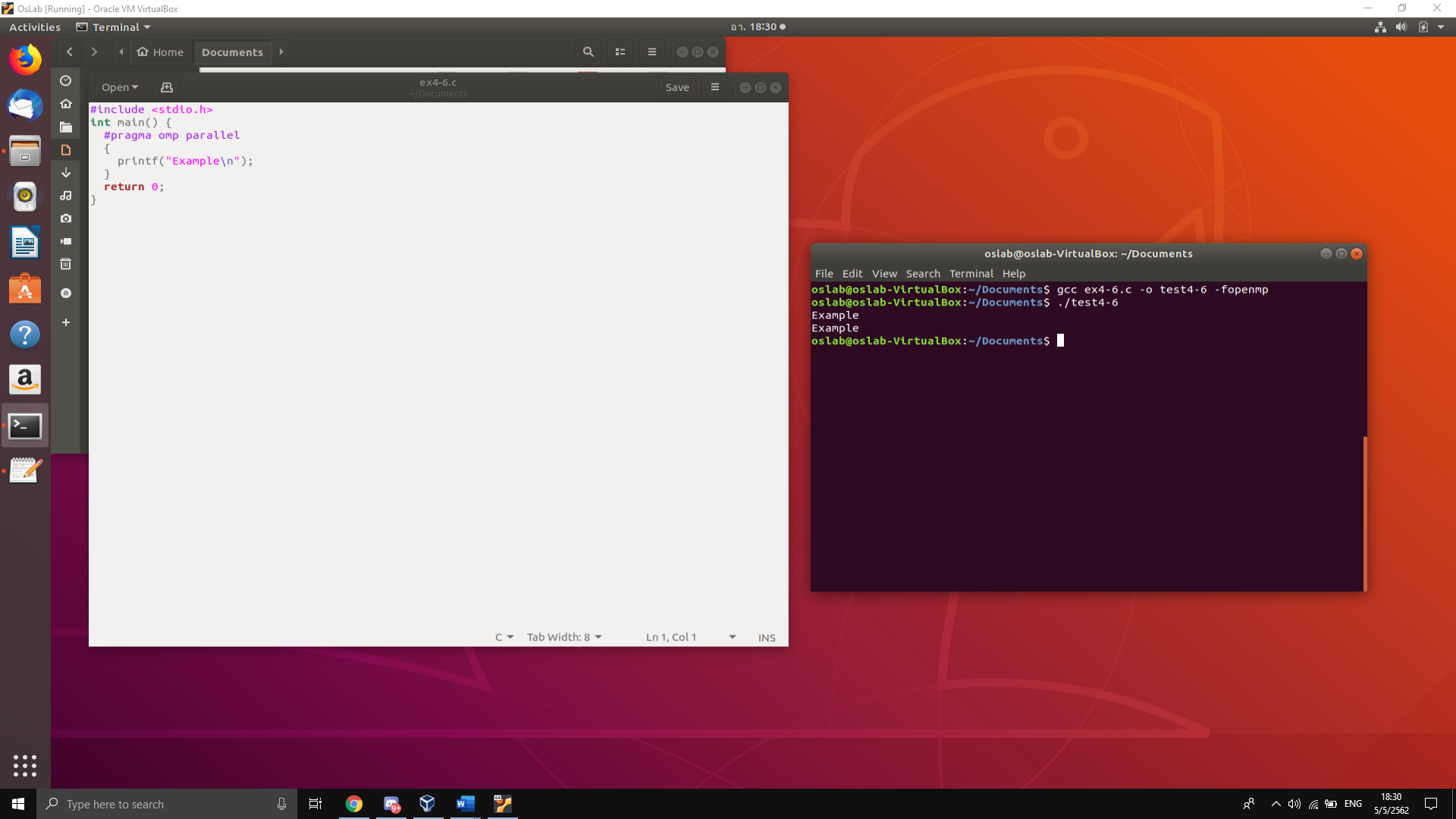
**5. (5 คะแนน) Thread Pool คืออะไร จงอธิบาย**

- คือกลุ่มของ thread ที่เตรียมรอถูกเรียกใช้งานแบบ Multi-thread ที่ง่ายๆ ไม่ซับซ้อน รวมถึงการจัดการกับการทำงานของ Method ที่ Run แบบ Asynchronous ด้วย

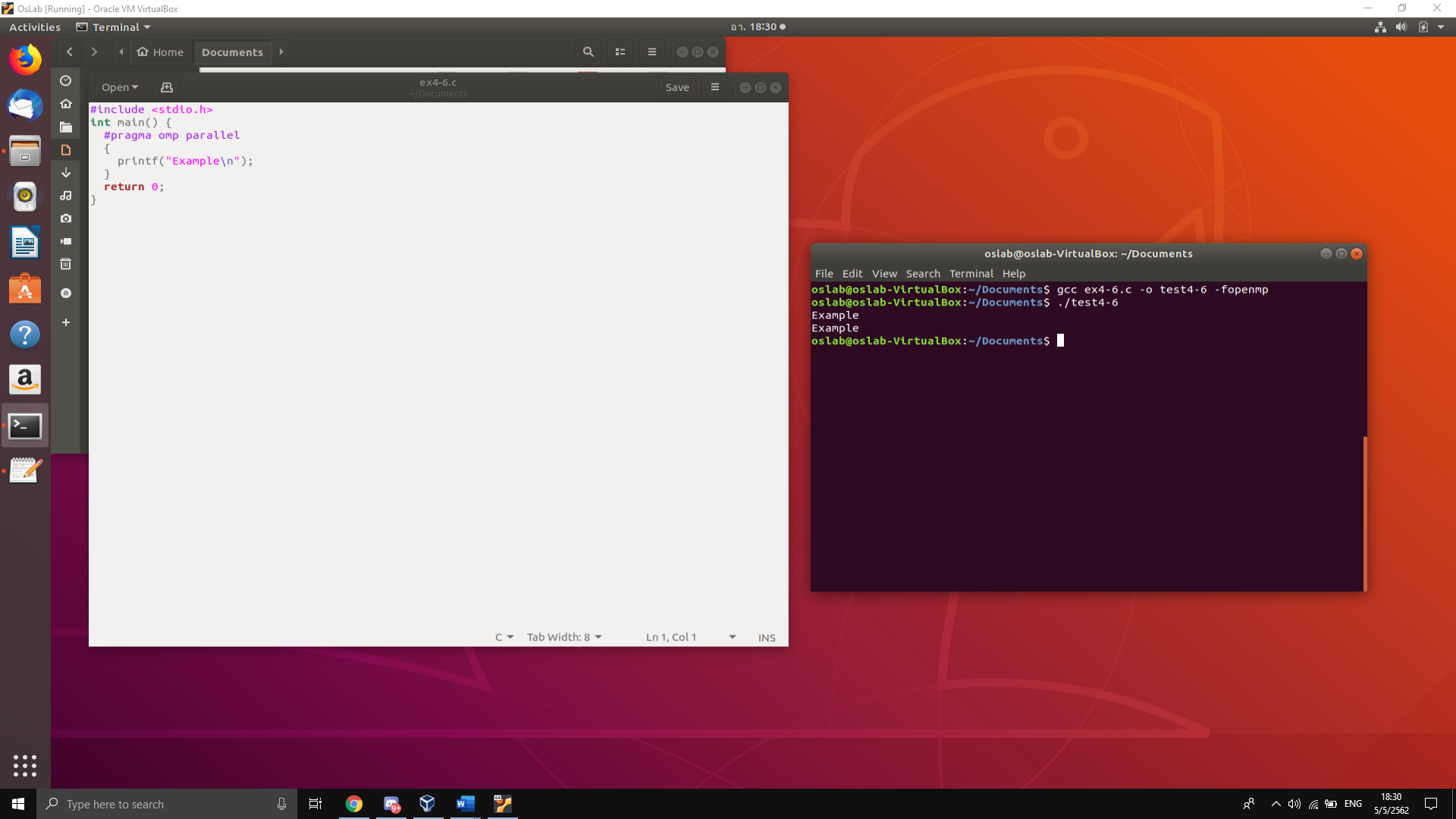
Thread Pool นั้นทำงานแบบ Background Thread เมื่อถูกเรียกใช้ thread pool ก็จะเรียก threads ออกมา หลังจากใช้เสร็จก็จะกลับเข้ามาใน thread pool โดยไม่ทิ้ง วนอยู่เรื่อยๆ

**6. (20 คะแนน) จงหาตัวอย่างของโปรแกรมที่เขียนด้วย Java Thread และ OpenMP แสดงโค้ดพร้อมผลลัพธ์ของการรันที่ได้ และอธิบายพฤติกรรมของโปรแกรม**

**OpenMP**



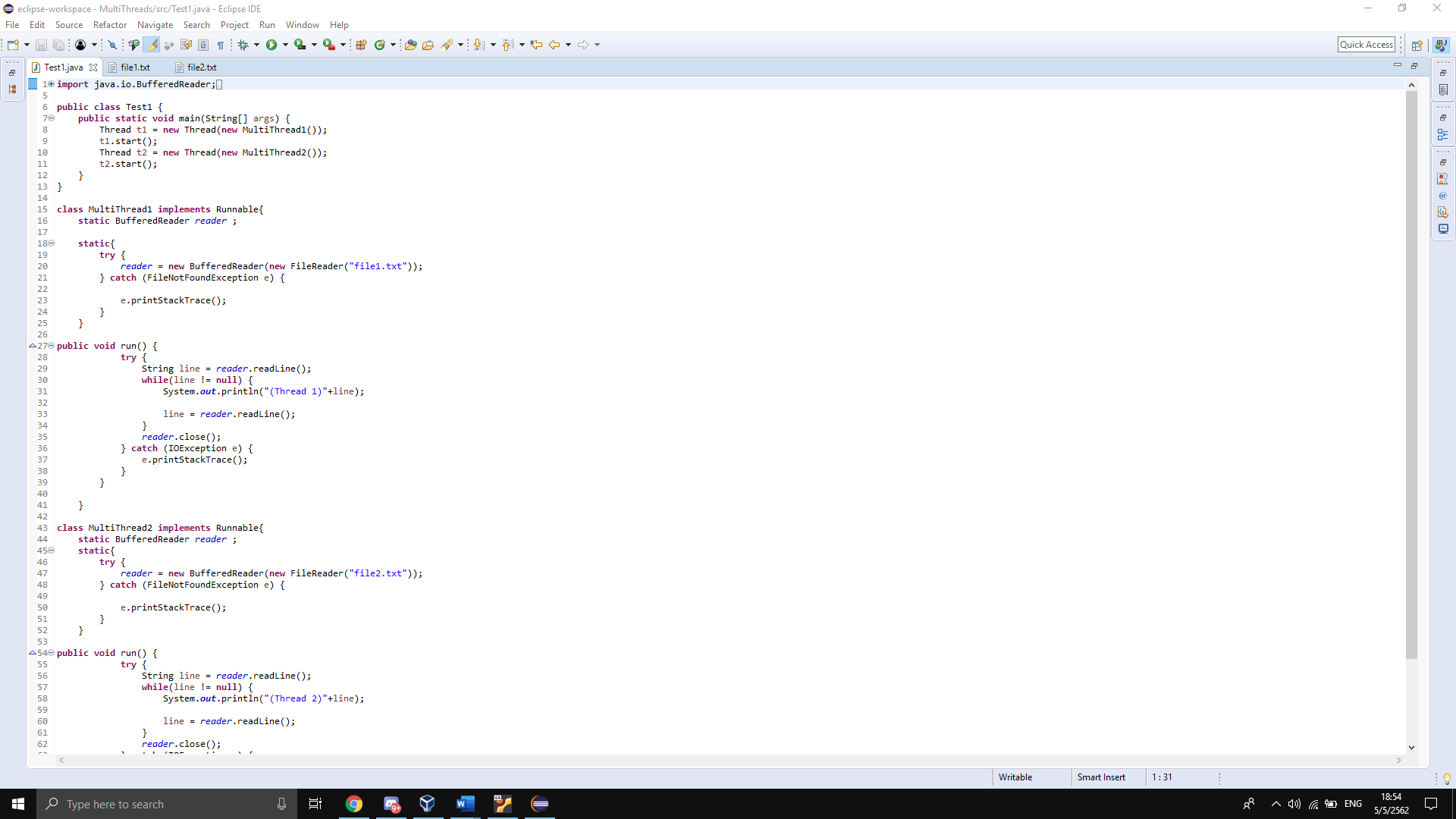
ผลลัพธ์

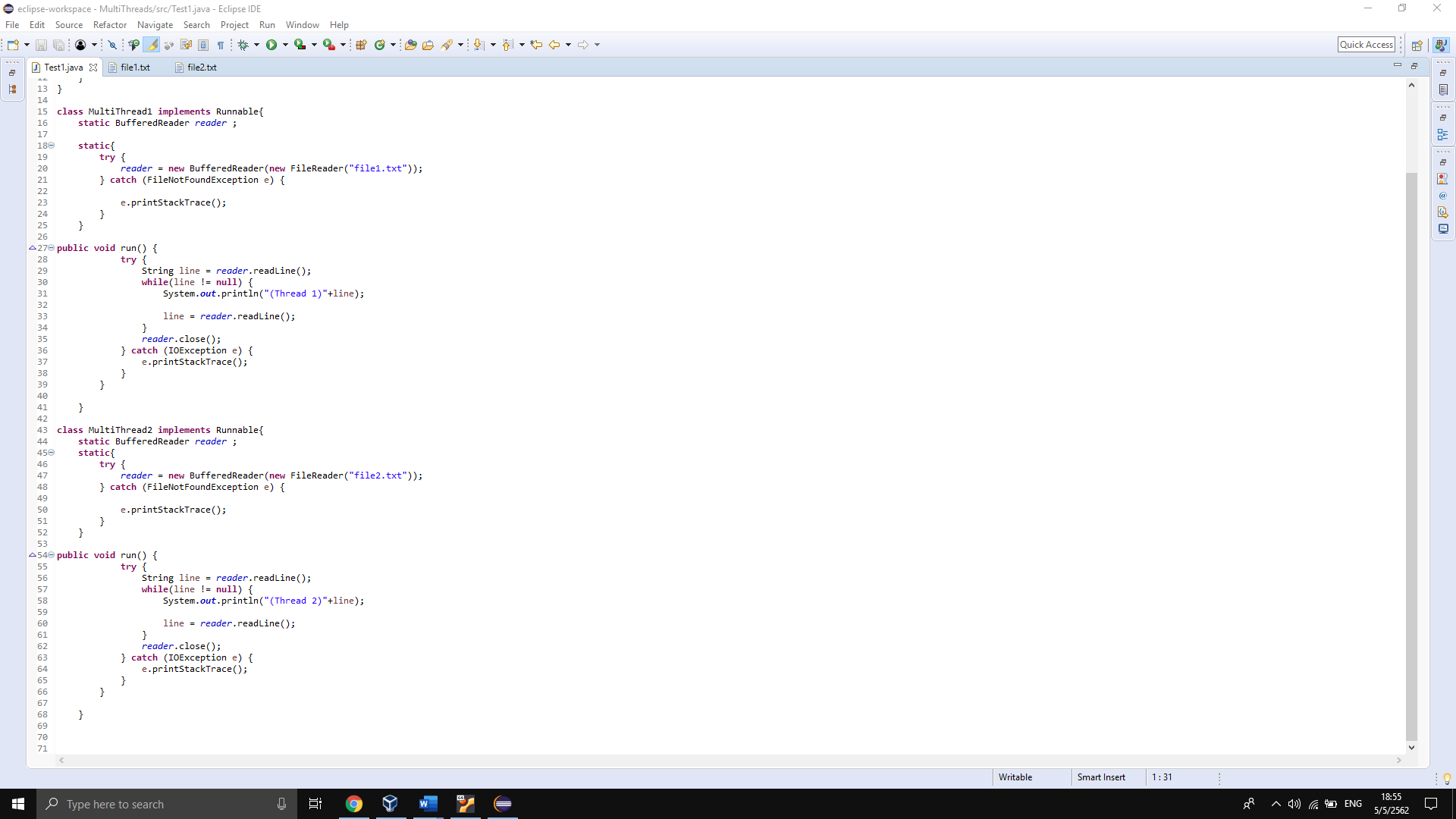


พฤติกรรมของโปรแกรม

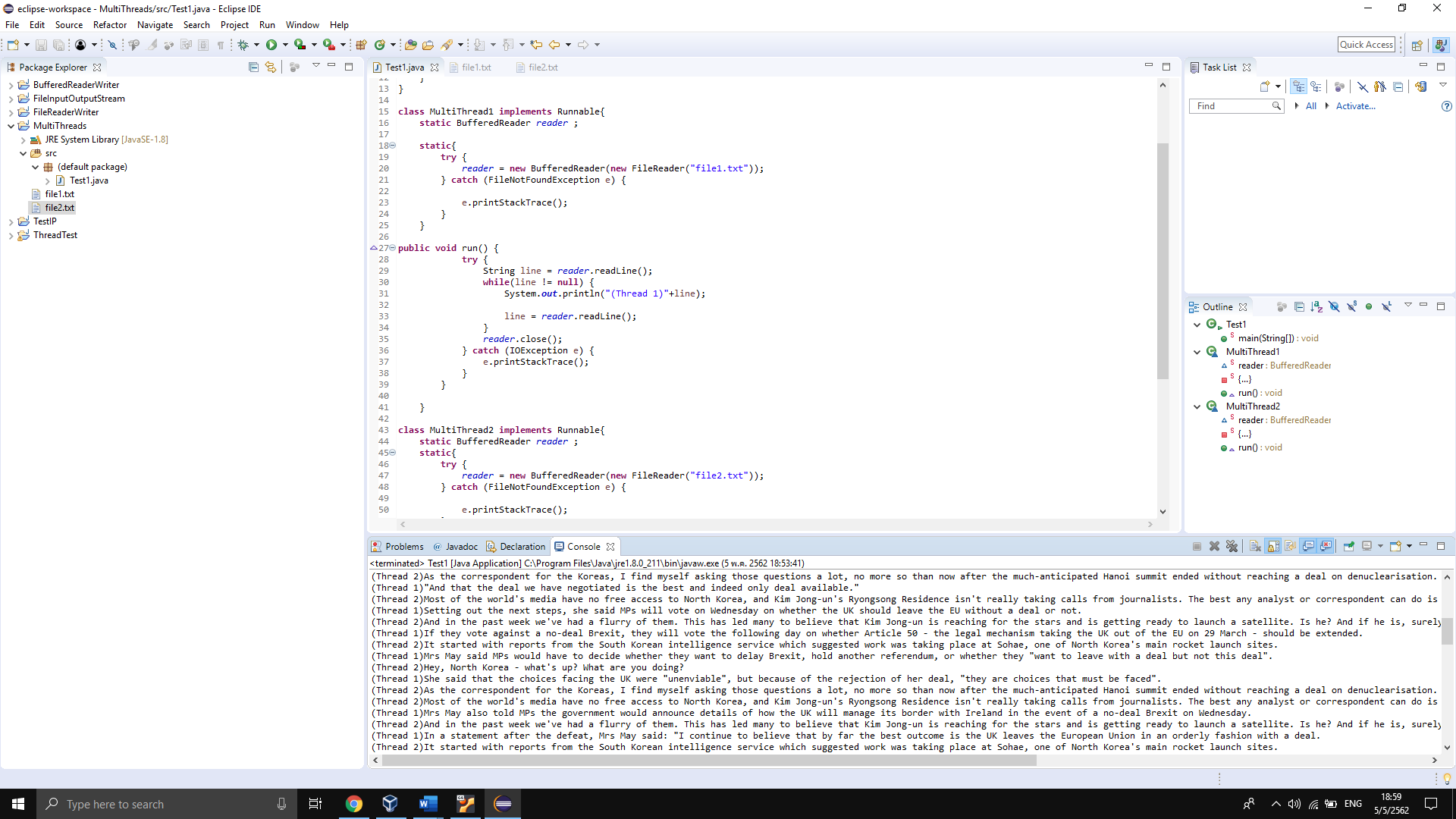
- โปรแกรมทำการปริ้นคำว่า “Example” ออกมาในเวลาเดียวกันตามจำนวน thread ซึ่งขึ้นอยู่ว่ามีกี่ core (ในที่นี้มีจำนวน 2 core )

**Java Thread**





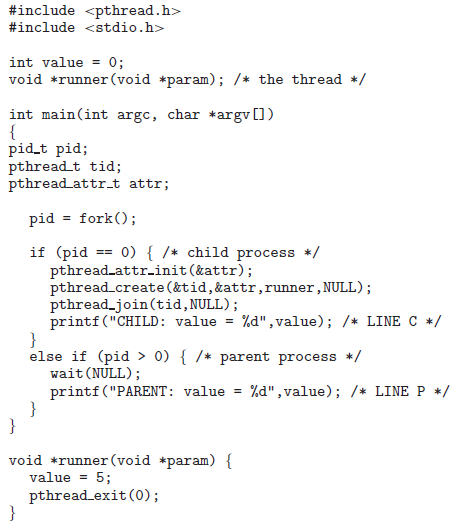
ผลลัพธ์

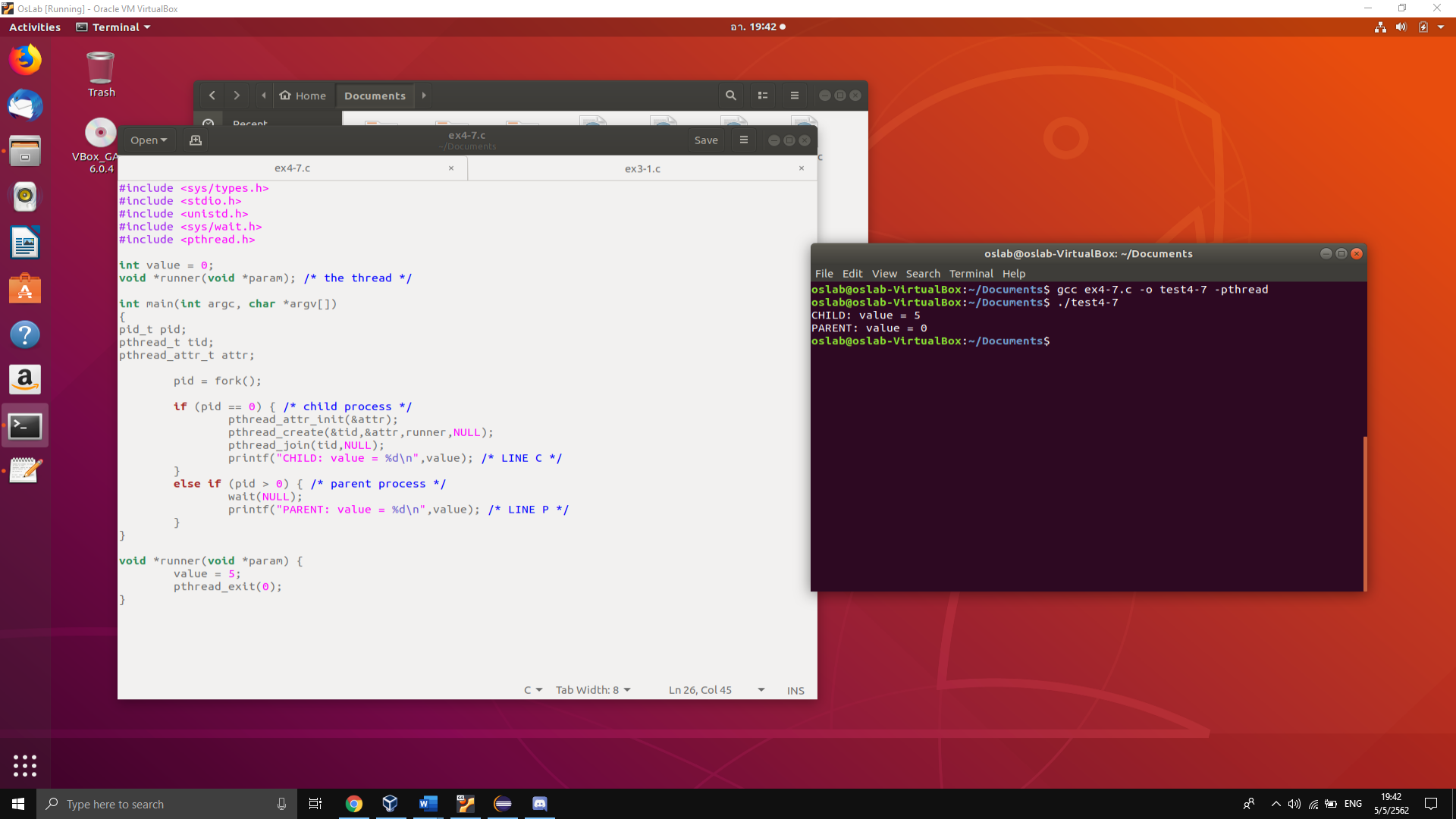


พฤติกรรมของโปรแกรม

-โปรแกรมทำการสร้าง thread ขึ้นมา 2 Thread และเริ่มการทำงาน Thread ทั้ง 2 จากคลาส MultiThread1 และ MultiThread2 โดยมีการทำงานคืออ่านค่าจากไฟล์ text ที่ต่างกันในเวลาเดียวกันและแสดงค่าที่อ่านได้ออกมาเป็นบรรทัด Thread ไหนทำเสร็จก่อนก็จะแสดงออกมาก่อน

**7. (10 คะแนน) ทดลองโค้ดภาษาซีต่อไปนี้ ผลลัพธ์ที่ได้จาก Line C และ Line P เป็นอย่างไร**





child process เกิดการ fork จากนั้น parent process รอ child process ทำงานจนเสร็จ จะมี Thread ถูกสร้างขึ้นสำหรับ child process และเรียกใช้ฟังก์ชัน runner() ซึ่งมีการกำหนดค่า value = 5 เมื่อจบการทำงานของ child process ที่ parent process จะมีค่า value = 0

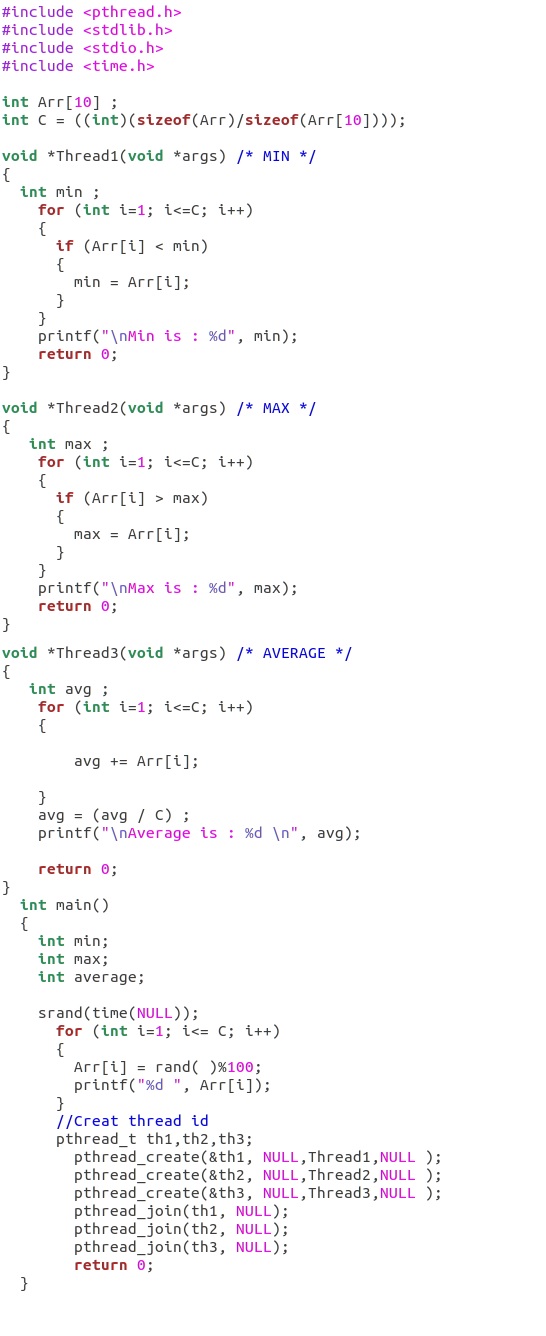
**8. (30 คะแนน) จงเขียนโปรแกรมแบบ Multithread โดยทีการทำงานดังนี้**

* **Main Thread สร้างเลขจำนวนเต็มแบบสุ่ม 10 ค่า แล้วแสดงออกทางจอภาพ**
* **Main Thread เตรียมตัวแปร min , max และ avg สำหรับเก็บค่าต่ำสุด , สูงสุด , และค่าเฉลี่ยตามลำดับ**
* **สร้าง Thread ย่อย 3 Thread มีหน้าที่ดังนี้ จากเลขจำนวนเต็มที่สุ่มมาจาก Main Thread**

1. **Thread#1: ค้นหาค่าต่ำสุด เก็บลงใน min**
2. **Thread#2: ค้นหาค่าสูงสุด เก็บลงใน max**
3. **Thread#3: ค้นหาค่าเฉลี่ย เก็บลงใน avg**

* **Main Thread แสดงผลลัพธ์ของ min , max , avg ออกทางจอภาพ**

Code



ผลลัพธ์

