รายงาน

เรื่อง โปรแกรมเครื่องจักรผลิตหน้ากากอนามัย

จัดทำโดย

1. นางสาวกรกช	ศิริกันทรมาศ	61090500401
2. นายคมศักดิ์	กรณย์ประกิตต์	61090500406
3. นายบวรศักดิ์	เหลือจันทร์	61090500418
4. นายวสิษฐ์พล	แก้วพูลศรี	61090500424
5. นางสาวสิริวิมล	กังสวณิช	61090500426

เสนอ

อาจารย์ชูเกียรติ วรสุชีพ

รายงานนี้เป็นส่วนหนึ่งของรายวิชา Operating Systems รหัสวิชา CSS226

ภาคเรียนที่ 2 ปีการศึกษา 2562

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา CSS226 Operating System ของสาขาวิชาวิทยาการคอมพิวเตอร์ประยุกต์ ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี โดยมีจุดประสงค์เพื่อนำ Bounded Buffer ไปประยุกต์ใช้ในการสร้างแอพลิเคชั่นด้วยภาษา JAVA ผ่านการใช้โปรแกรม NetBeans IDE 8.2 ในการสร้างแอพลิเคชั่นเพื่อศึกษาหลักการทำงานของ Bounded Buffer และนำมาประยุกต์ใช้กับโปรแกรมเครื่องจักรผลิตหน้ากากอนามัย ซึ่งจากสถานการณ์ COVID-19 ในปัจจุบันทำให้การผลิตหน้ากากอนามัยนั้นไม่เพียงพอต่อความต้องการของผู้คนเป็นอย่างมากรวมถึงราคาของหน้ากากอนามัยนั้นก็สูงมากเช่นกัน จึงทำให้คณะผู้จัดทำเกิดความสนใจในประเด็นปัญหานี้ และได้เลือกหัวข้อนี้มาทำการสร้างแอพลิเคชั่น รวมถึงต้องการนำความรู้เรื่อง Bounded Buffer ที่ได้จากการเรียนในห้องเรียนมาต่อยอดในการเขียนโปรแกรมภาษา JAVA คณะผู้จัดทำหวังเป็นอย่างยิ่งว่ารายงานฉบับนี้จะมีประโยชน์และเป็นตัวอย่างของการนำ Bounded Buffer มาประยุกต์ใช้ในการพัฒนาโปรแกรมให้แก่ผู้พบเห็นได้ในอนาคต

สารบัญ

คำนำ ก
สารบัญ ข
บทนำ 1
ความเป็นมาและความสำคัญของโปรแกรม 1
1) วัตถุประสงค์ 1
2) ผลที่คาดว่าจะได้รับ 1
คำอธิบายและขั้นตอนการทำงานของโปรแกรม 2
1. คำอธิบายโปรแกรม 2
2. ขั้นตอนการทำงานของโปรแกรม 2
ผลลัพธ์ของโปรแกรม 7
สรุปผล 8
บรรณานุกรม 9

บทนำ

ความเป็นมาและความสำคัญโปรแกรม

เนื่องจากสถานการณ์ COVID-19 ที่กำลังแพร่ระบาดในปัจจุบัน ทำให้การผลิตหน้ากากอนามัยนั้นไม่เพียงพอต่อความต้องการของผู้คน รวมไปถึงราคาของหน้ากากอนามัยนั้นมีราคาที่สูงมาก ทำให้คณะผู้จัดทำเกิดความสนใจในประเด็นปัญหานี้ จึงได้ทำการค้นคว้าและรวบรวมข้อมูลเกี่ยวกับการวิธีการผลิตหน้ากากอนามัย รวมถึงนำความรู้เรื่อง Bounded Buffer ที่ได้เรียนจากการเรียนในห้องเรียนมาประยุกต์ใช้ในการสร้างโปรแกรมเครื่องจักรผลิตหน้ากากอนามัย เพื่อศึกษาเรียนรู้และเพิ่มความเข้าใจเกี่ยวกับ Bounded Buffer และการเขียนโปรแกรมภาษา JAVA มากยิ่งขึ้น โดยใช้โปรแกรม NetBeans IDE 8.2 ในการสร้างโปรแกรมเครื่องจักรผลิตหน้ากากอนามัยขึ้นมา

1) วัตถุประสงค์

1. เพื่อศึกษาวิธีการผลิตหน้ากากอนามัย
2. เพื่อศึกษาและเรียนรู้เกี่ยวกับหลักการทำงานของ Bounded Buffer ให้เข้าใจมากยิ่งขึ้น
3. เพื่อนำ Bounded Buffer มาประยุกต์ใช้ในการเขียนโปรแกรมภาษา JAVA
4. เพื่อเป็นการนำความรู้ที่ได้จากการเรียนในห้องเรียนมาต่อยอดและใช้ประโยชน์

2) ผลที่คาดว่าจะได้รับ

1. ได้ทราบวิธีการผลิตหน้ากากอนามัย
2. ได้ทราบถึงหลักการทำงานของ Bounded Buffer ชัดเจนยิ่งขึ้น
3. สามารถนำความรู้ที่ได้จากการศึกษาเรื่อง Bounded Buffer มาประยุกต์ใช้ในการเขียนโปรแกรมภาษา JAVA ได้
4. ได้ทราบถึงประโยชน์ของการนำ Bounded Buffer มาประยุกต์ใช้ในการเขียนโปรแกรม

คำอธิบายและขั้นตอนการทำงานของโปรแกรม

1. คำอธิบายโปรแกรม

โปรแกรมนี้เป็นโปรแกรมเครื่องจักรผลิตหน้ากากอนามัย โดยการควบคุมการทำงานผ่านระบบที่ติดตั้งอยู่ในเครื่อง เมื่อเราทำการเปิดโปรแกรมขึ้นมา จะเป็นการเปิดระบบควบคุมการทำงานของเครื่องจักร ระบบจะทำการผลิตหน้ากากอนามัยออกมาแบบอัตโนมัติไปเรื่อย ๆ จนกว่าจะมีการปิดระบบการทำงาน โดยเครื่องจักรจะมีช่องการผลิต (slot) ทั้งหมด 6 ช่อง เมื่อเครื่องจักรทำงานเสร็จในแต่ละครั้ง เครื่องจักรก็จะผลิตหน้ากากอนามัยออกมา 1 ชิ้น พร้อมกับแจ้งให้ผู้ใช้ทราบผ่านหน้าจอว่าหน้ากากอนามัยนั้นผลิตออกมาเรียบร้อยแล้วเพื่อให้ผู้ใช้นำไปใช้และเมื่อเราทำการปิดโปรแกรมก็จะเป็นการปิดระบบการทำงานของเครื่องจักรทำให้หยุดการผลิตหน้ากากอนามัยนั่นเอง ซึ่งโปรแกรมนี้เหมาะสำหรับโรงงานและผู้ที่ต้องการผลิตหน้ากากอนามัยเป็นจำนวนมาก โดยการนำเอา Bounced Buffer มาใช้ในโปรแกรมนั้นจะเป็นดังนี้

1. กำหนดให้ Buffer แต่ละช่องเก็บหน้ากากอนามัยเอาไว้ Producer จะทำการสร้างข้อมูลก่อน Consumer ในที่นี้ คือ การผลิตหน้ากากอนามัย
2. เมื่อ Producer ทำการผลิตหน้ากากอนามัยเสร็จ ก็จะให้ Consumer มาเอาหน้ากากอนามัยออกไปใช้ โดยจะตั้งเวลา Consumer ให้หลับก่อน 2 วินาที เพื่อให้ Producer ได้ไปสร้างข้อมูลใน Buffer ก่อน ซึ่ง Producer จะหลับทุกๆ 4 วินาที
3. หลังจากที่เข้าไปสร้างข้อมูลใน Buffer ในแต่ละช่อง เมื่อ Consumer ตื่นขึ้นมาหลัง 2 วินาที ก็จะเข้าไปดึงข้อมูลออกมา หลังจากนั้นจะหลับต่อ
4. จะมีการสุ่มเวลาตั้งแต่ 1-9 วินาที เหตุผลเพราะว่าในความเป็นจริง คนที่ใช้ข้อมูลไม่ได้ใช้ข้อมูลในเวลาที่เท่าเดิมตลอด ซึ่งจะมีช้าบ้างเร็วบ้าง จึงกำหนดให้หลับตั้งแต่ 1-9 วินาที เพื่อความสมจริง

2. ขั้นตอนการทำงานของโปรแกรม

การทำงานของโปรแกรมจะเริ่มที่ main() โดยมี Class ConsumerProducerMask เป็น class หลัก ซึ่งในขั้นตอนแรกโปรแกรมจะ print ข้อความออกมาว่า Started working on the Medical Mask system.

บรรทัดที่ (14) จะสร้าง Object Buffer และส่ง Buffer size ไปใน Class Buffer

บรรทัดที่ (15) และ (16) จะเป็นการสร้าง Object Producer และ Consumer ขึ้นมาและร่วมกันใช้ Buffer เดียวกัน

บรรทัดที่ (17) และ (18) เป็นการเริ่มต้นการทำงานของ Thread

บรรทัดที่ (19) และ (20) จะเป็นการกำหนดให้ Thread ของ Producer ทำงานก่อน Consumer

```
10  public class ConsumerProducerMask  {
11
12      public static void main(String[] args) throws InterruptedException  {
13      System.out.println("Started working on the Medical Mask system.\n");
14      Buffer c = new Buffer(6); // buffer has size 6
15      Producer t1 = new Producer(c);
16      Consumer t2 = new Consumer(c);
17      t1.start();
18      t2.start();
19      t1.join();
20      t2.join();
21      }
22  }
```

บรรทัดที่ (24) เป็น Class Buffer ที่เป็นที่เก็บข้อมูลโดยใน Class นี้จะมีตัวแปร MaxBuffsize ที่เป็นตัวแปรที่รับ Parameter จาก main เป็นขนาดของ Buffer ตัวแปร store เก็บข้อมูลเป็น Array จากที่ Producer ทำการสร้างไว้ ตัวแปร BufferStart เป็นตัวแปรกำหนดการใช้งานในบัฟเฟอร์ช่องแรก ตัวแปร BufferEnd เป็นตัวแปรที่เปรียบเสมือน Pointer ที่เลื่อนตำแหน่งไปเรื่อยๆ จนสุดช่องบัฟเฟอร์และวนกลับมาที่ช่องแรก BufferSize เป็นตัวแปรที่บอกถึงขนาดของบัฟเฟอร์ที่มีข้อมูลอยู่ในช่องว่ามีการสร้างข้อมูลไว้กี่ช่องแล้ว

บรรทัดที่ (28) คือ Contructor ที่รับค่าต่างๆ

```
24  class Buffer {
25  private final int MaxBuffSize;
    private String[] store;
27  private int BufferStart, BufferEnd, BufferSize;
28  public Buffer(int size) {
29  MaxBuffSize = size;
30  BufferEnd = -1;
31  BufferStart = 0;
32  store = new String[MaxBuffSize];
33  }
```

บรรทัดที่ (34) คือ Insert() ที่ทำหน้าที่ในการสร้างข้อมูลลงในช่องบัฟเฟอร์มีการรับ Parameter คือค่าที่ส่งมาจาก Producer

บรรทัดที่ (35) เป็นการเช็คว่าถ้าหาก Producer สร้างข้อมูลครบทุกช่องแล้วให้ Producer หยุดรอ Consumer มาปลุกให้ตื่น

บรรทัดที่ (43) BufferEnd จะ +1 ไปเรื่อยๆทุกครั้งที่ Producer ทำการเข้ามาสร้างข้อมูลโดยถ้าหากสร้างข้อมูลจนครบทุกช่องแล้วจะวนกลับมาสร้างใหม่ในช่องแรก และวนไปแบบนี้เรื่อยๆ

บรรทัดที่ (44) เป็นการเก็บข้อมูลที่ Producer ทำการสร้างไว้

บรรทัดที่ (45) จะเป็นการเพิ่มจำนวนช่องของบัฟเฟอร์ที่มีข้อมูลไปเรื่อยๆทุกครั้งที่มีการสร้างข้อมูล

```
34      public synchronized void insert(String data) {
35          while (BufferSize == MaxBuffSize){
36         try {
37        wait();
38         }
39    catch (InterruptedException e) {
40            Logger.getLogger(Consumer.class.getName()).log(Level.SEVERE, null, e);
41                }
42            }
43       BufferEnd = (BufferEnd + 1) % MaxBuffSize;
44       store[BufferEnd] = data;
45       BufferSize++;
46       notifyAll();
47       }
```

บรรทัดที่ (48) คือ useData() ทำหน้าที่ในการดึงข้อมูลของ Consumer

บรรทัดที่ (49) ได้ทำการเช็คค่าหาก BufferSize มีค่าเท่ากับ 0 ให้ทำการหยุดรอ Producer มาปลุก

บรรทัดที่ (57) เป็นการดึงข้อมูลจากในบัฟเฟอร์มา

บรรทัดที่ (58) BufferSize จะ -1 ไปเรื่อยๆทุกครั้งที่ Consumer ทำการเข้ามาดึงข้อมูลโดยถ้าหากดึงข้อมูลจนครบทุกช่องแล้วจะวนกลับมาดึงข้อมูลใหม่ในช่องแรก และวนไปแบบนี้เรื่อยๆ

บรรทัดที่ (59) เป็นการลดจำนวนช่องที่มีข้อมูลลง

บรรทัดที่ (60) เป็นการปลุก Producer ให้ตื่นขึ้น และสุดท้ายจะ return ค่ากลับไปที่ consumer

```
48      public synchronized String useData() {
49          while (BufferSize == 0) {
50         try {
51             wait();
52              }
53       catch (InterruptedException e) {
54         Thread.currentThread().interrupt();
55        }
56             }
57      String data = store[BufferStart];
58       BufferStart = (BufferStart + 1) % MaxBuffSize;
59       BufferSize--;
60       notifyAll();
61       return data;
62         }
    }
```

บรรทัดที่ (64) เป็น Class Consumer ทำหน้าที่ในการเป็นผู้ใช้ข้อมูล มีตัวแปร Buffer ที่เป็น Object ในการอ้างถึง Class Buffer โดยมี Contructor ในการรับค่า Object ที่ส่งมาจาก main()

บรรทัดที่ (70) จะเป็นการเริ่มต้นการทำงาน Thread โดยจะให้หลับก่อน 2000 เพื่อให้ Producer ได้เข้าไปทำการสร้างข้อมูลก่อน

บรรทัดที่ (72) จะให้ Consumer หลับเป็นเวลา 2 วินาที เพื่อให้ Producer ได้ทำการเข้าไปสร้างข้อมูลได้ก่อน

บรรทัดที่ (76) เป็นการบอกว่าโปรแกรมที่จะทำงานไปเรื่อยๆ

บรรทัดที่ (77) เป็นการทำลูปที่ให้ Consumer เข้าไปดึงข้อมูลจำนวนครั้งตามช่องของบัฟเฟอร์ โดยในลูป for จะมีตัวแปร c ที่ทำหน้าที่ในการรับค่าจาก useData()

บรรทัดที่ (79) แสดงผลลัพธ์ออกมา

บรรทัดที่ (81) เป็นการกำหนดตัวแปรขึ้นมาเพื่อรับค่าจาก math.random() เพื่อใช้ในการสุ่มเวลาในการหลับของ Consumer หลังจากที่เข้าไปทำการดึงข้อมูลเสร็จในทุกๆครั้ง โดยจะมีค่าระหว่าง 1-9 เหตุผลทีเราทำการสุ่มเวลาเพราะว่าจะได้มีความสมจริงที่ Consumer ที่เปรียบเหมือนพนักงานอาจจะมาหยิบหน้ากากอนามัยออกไปจากเครื่องในเวลาที่ไม่เท่ากัน ซึ่งในบรรทัดถัดมาจะสังเกตว่าเราได้ทำการคูณด้วย 1000 โดย Consumer จะทำการหลับได้ในช่วง 1-9 วินาที ทุกครั้งหลังดึงข้อมูลเสร็จ

```
64      class Consumer extends Thread{
65    private final Buffer buffer;
66        public Consumer(Buffer b) {
67        buffer = b;
68        }
69    @Override
70    public void run() {
71    try {
72                sleep(2000); // ให้ Thread หลับไปก่อนครั้งแรกเพื่อให้ producer ได้ไปสร้างข้อมูลไว้ก่อน
73              } catch (InterruptedException ex) {
74                 Logger.getLogger(Consumer.class.getName()).log(Level.SEVERE, null, ex);
75              }
76    while(true){
77    for(int i=1;i<7;i++){
78    String c = buffer.useData();
79    System.out.println("\tSlot"+i+":"+"receive "+c);
80    try {
81          int targetNumber = (int)(Math.random()* 10); // การดึงข้อมูลอาจจะใช้เวลาไม่เท่ากันเพื่อความสมจริงจะใช้เป็น math.random()
82                Thread.sleep(targetNumber*1000);  // Thread หลับไปตามเวลาที่ math.random() สุ่มออกมา
83            } catch (InterruptedException ex) {
84                Logger.getLogger(Consumer.class.getName()).log(Level.SEVERE, null, ex);}
85             }
86            }
87       }
88    }
```

บรรทัดที่ (89) เป็น Class Producer ทำหน้าที่ในการเป็นผู้สร้างข้อมูล มีตัวแปร buffer ที่เป็น Object ในการอ้างถึง Class Buffer โดยมี Contructor ในการรับค่า Object ที่ส่งมาจาก main()

บรรทัดที่ (95) เป็นการเริ่มต้นการทำงานของ Thread มีลูป for ทีวนไปให้ Producer ทำการสร้างข้อมูลตามจำนวนช่องของบัฟเฟอร์

บรรทัดที่ (98) เป็นการใช้งาน insert() และทำการส่งข้อความ “Medical Mask” ไปให้ใน Insert() ใน Class buffer

บรรทัดที่ (99) แสดงผลลัพธ์ออกมา

บรรทัดที่ (101) ให้ Producer หลับไปเป็นเวลา 4 วินาที หลังจากที่ทำการสร้างข้อมูลเสร็จในแต่ละครั้ง

```
89    class Producer extends Thread {
90    private final Buffer buffer;
91    public Producer(Buffer b) {
92    buffer = b;
93    }
94    @Override
95        public void run() {
96       while(true){
97        for(int i =1;i<7;i++){
98        buffer.insert("Medical Mask");
99        System.out.println("Slot"+i+":"+" produce Medical Mask");
100       try{
101           Thread.sleep(4000); // Thread หลับไป 4 วิ
102       }
103       catch(InterruptedException ex){
104            Logger.getLogger(Producer.class.getName()).log(Level.SEVERE, null, ex); }
105
106         }
107       }
108    }
109   }
```

ผลลัพธ์ของโปรแกรม

```
run:
Started working on the Medical Mask system.
Slot1: produce Medical Mask
        Slot1:receive Medical Mask
Slot2: produce Medical Mask
        Slot2:receive Medical Mask
Slot3: produce Medical Mask
Slot4: produce Medical Mask
        Slot3:receive Medical Mask
Slot5: produce Medical Mask
Slot6: produce Medical Mask
        Slot4:receive Medical Mask
Slot1: produce Medical Mask
Slot2: produce Medical Mask
        Slot5:receive Medical Mask
Slot3: produce Medical Mask
Slot4: produce Medical Mask
        Slot6:receive Medical Mask
        Slot1:receive Medical Mask
        Slot2:receive Medical Mask
Slot5: produce Medical Mask
        Slot3:receive Medical Mask
Slot6: produce Medical Mask
Slot1: produce Medical Mask
        Slot4:receive Medical Mask
        Slot5:receive Medical Mask
```

จะเห็นได้ว่า Producer(produce Medical Mask) จะเข้าไปสร้างข้อมูลก่อนที่ Consumer (receive Medical Mask) จะเข้าไปดึงข้อมูล โดยที่โปรแกรมจะ Run ไปเรื่อย ๆ จนกว่าโปรแกรมจะปิดไปเอง

สรุปผล

จากการที่คณะผู้จัดทำได้สร้างโปรแกรมเครื่องจักรผลิตหน้ากากอนามัย ทำให้คณะผู้จัดทำได้ทราบถึงประโยชน์ของการนำความรู้ในเรื่องของการนำ Bounded Buffer มาประยุกต์ใช้ให้เกิดประโยชน์ ซึ่ง Bounced Buffer เป็นพื้นที่พักเก็บข้อมูลเอาไว้เป็นช่องๆเพื่อให้ผู้ใช้ดึงออกไปใช้ได้ ซึ่งเปรียบช่อง Buffer เหมือนเป็นทางออกสำหรับการผลิตหน้ากากอนามัยแต่ละช่องของเครื่องจักร แล้วผู้ใช้ก็สามารถหยิบหน้ากากเอาไปใช้ได้ตามช่องของ Buffer โดยคณะผู้จัดทำได้บรรลุตามวัตถุประสงค์ที่ได้กำหนดเอาไว้ ซึ่งได้เล็งเห็นว่า Bounced Buffer เป็นรากฐานของหลายๆโปรแกรมในปัจจุบัน ซึ่งในอนาคตคณะผู้จัดทำจะนำความรู้จากการทำโปรแกรมในครั้งนี้และความรู้เกี่ยวกับ Bounced Buffer ไปพัฒนาต่อยอดสำหรับการสร้างโปรแกรมต่าง ๆ ให้เกิดประโยชน์สูงสุดต่อผู้ใช้งานทุกคนต่อไป

บรรณานุกรม

Course Hero, Inc. (2563, เมษายน 16). *Bounded Buffer in Java*. Retrieved from Course Hero: https://www.coursehero.com/file/6744593/Bounded-Buffer-in-Java/

Geeksforgeeks. (2563, เมษายน 16). *Producer-Consumer solution using threads in Java*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/producer-consumer-solution-using-threads-java/

GitHub, Inc. (2563, เมษายน 18). *GitHub - promix01/Medical-Mask-with-Producer-Consumer*. Retrieved from GitHub: https://github.com/promix01/Medical-Mask-with-Producer-Consumer

Green World Media (Thailand) Co., Ltd. (17 เมษายน 2563). *การผลิตหน้ากากอนามัยทำกันอย่างไร?* เข้าถึงได้จาก Modern Manufacturing: https://www.mmthailand.com/%E0%B8%9C%E0%B8%A5%E0%B8%B4%E0%B8%95%E0%B8%AB%E0%B8%99%E0%B9%89%E0%B8%B2%E0%B8%81%E0%B8%B2%E0%B8%81%E0%B8%AD%E0%B8%99%E0%B8%B2%E0%B8%A1%E0%B8%B1%E0%B8%A2/

Trustees of Dartmouth College. (2563, เมษายน 17). *Bounded Buffers*. Retrieved from Dartmouth: https://www.cs.dartmouth.edu/~scot/cs10/lectures/27/27.html

University of Hawai‘i. (2563, เมษายน 17). *The buffer and the producer and consumer threads.* Retrieved from University of Hawai‘i: http://www2.hawaii.edu/~walbritt/ics240/synchronization/Solution.java