CPE217 – Homework 4

Homework: Queues and Stacks

Homework Due Date: August 15, 2025

Patiwet Wuttisarnwattana, Ph.D.

Department of Computer Engineering

- คำชี้แจงการส่งงาน
- ให้ทุกคนเข้าโมดูล Assignment (Link สีแดง) อ่านคำอธิบายการบ้านใน PDF และโหลด Java Starter Code เพื่อทำความเข้าใจโจทย์
- หลังจากนั้นให้ทุกคนทำการบ้านพร้อมกันกับกลุ่มของตัวเอง พร้อมปรึกษากับ Core Person ในกลุ่มตัวเองว่าจะส่งคำตอบสุดท้ายว่าเป็นอะไร
- ทุกคนสามารถตรวจคำตอบของตัวเองได้ ว่าทำมาถูกต้องหรือไม่โดยใช้โมดูล Quiz (Link สีเหลือง) แต่อาจารย์จะตรวจคะแนนจาก Core Person เท่านั้น นศ ทุกคนจะได้คะแนนเท่ากันทั้งกลุ่ม แม้ว่าคุณจะส่งโค้ดใน Link สีเหลืองแตกต่างกันก็ตามที
- นศ ต้องเติมโค้ดในพื้นที่ที่กำหนดให้เท่านั้น ห้ามประกาศตัวแปรระดับคลาสเพิ่ม ห้ามสร้างฟังก์ชันอื่น ๆ หรือเรียกใช้ฟังก์ชันพิเศษเพิ่ม (ไม่แน่ใจสามารถสอบถามได้ที่อาจารย์) ให้เติมโค้ดใน Template ของอาจารย์เท่านั้น ฝ่าฝืนหักคะแนน 25% แม้ผลลัพธ์สุดท้ายจะทำงานได้ถูกต้อง
- เมื่อ Core Person ส่งคำตอบแล้ว ให้ Core Person เข้าโมดูล Assignment (Link สีแดง) และใส่รหัสของเพื่อนในกลุ่มลงใน ช่องคำตอบ
- TA จะตรวจคำตอบในโมดูล Quiz และนำคะแนนมาลงในโมดูล Assignment เพื่อให้ทุกคนในกลุ่มได้คะแนนเท่ากันครับ
- โค้ดของคุณต้องมี Comment เพื่ออธิบายว่าโค้ดดังที่เห็นอยู่นี้ทำงานอะไร หรือ if นี้ใช้เพื่อแยกกรณีใดออกมา กลุ่มไหนที่ไม่มีคอมเม้นต์จะถูกหักคะแนน 50% การเขียนคอมเม้นต์ไม่ต้องเขียนละเอียดยิบ เขียนเท่าที่คุณต้องการให้ผู้ตรวจทราบก็พอ

นศ ที่จะส่งคำตอบ ท่านต้องให้คำมั่นปฏิญาณต่อคำพูดดังต่อไปนี้ หากไม่สามารถทำได้ ท่านจะไม่มีสิทธิ์ส่งงาน

- *ข้าพเจ้าและเพื่อนในกลุ่มเข้าใจและตระหนักดีว่า ในการทำการบ้านนี้ ข้าพเจ้าและเพื่อนในกลุ่มจะช่วยกันทำงานนี้ให้เสร็จสิ้นเอง โดยไม่ปรึกษาหรือแบ่งปันข้อมูลกับกลุ่มอื่น ๆ หรือบุคคลภายนอก*
- *หากข้าพเจ้าเป็นรุ่นพี่ที่กลับมาเรียนวิชานี้อีกครั้ง ข้าพเจ้าตระหนักดีว่า ข้าพเจ้าจะทำงานให้เสร็จสิ้นเองอีกครั้ง โดยไม่ดูคำตอบของปีก่อน ๆ*
- *ข้าพเจ้าจะไม่เผยแพร่เนื้อหาโจทย์การบ้านนี้ออกสู่สาธารณะโดยเด็ดขาด*
- *หากข้าพเจ้าไม่สามารถปฏิบัติตามคำมั่นนี้ได้ ข้าพเจ้ายินดีที่จะยอมรับคะแนน ศูนย์คะแนน ในทุก ๆ การบ้านโดยไม่โต้แย้ง*

การบ้านนี้นักศึกษาจะได้ทำโจทย์การใช้ Queue และ Stack ในระบบโรงหนัง

การประยุกต์ใช้ Queue และ Stack กับโรงหนัง Major

โรงภาพยนตร์ Major Cineplex สาขาใหญ่ ต้องการมอบประสบการณ์สุดพิเศษให้กับคออนิเมะ จึงจัดกิจกรรมพิเศษฉลองการเข้าฉายของภาพยนตร์ "ดาบพิฆาตอสูร: ภาคปราสาทไร้ขอบเขต" ทุกคนที่ซื้อบัตรชมภาพยนตร์เรื่องนี้จะได้รับของที่ระลึกสุดลิมิเต็ด ซึ่งเป็นการ์ดลายตัวละคร โดยของที่ระลึกจะถูกวางกองรวมกันไว้ที่เคาน์เตอร์แจกของ และทุกคนที่ต้องการรับของจะต้องมาต่อแถวตามคิวเพื่อรับของทีละคนก่อนเข้าโรงภาพยนตร์

เพื่อจัดการการแจกของที่ระลึกนี้ ทางโรงภาพยนตร์ได้ว่าจ้างนักศึกษาจากภาควิชาวิศวกรรมคอมพิวเตอร์มาช่วยพัฒนาระบบ โดยใช้ความรู้ด้านโครงสร้างข้อมูลที่ได้เรียนมา ซึ่งระบบนี้จะช่วยให้การแจกของที่ระลึกเป็นไปอย่างราบรื่นและเป็นระเบียบ

ให้นักศึกษาทำความเข้า Starter Code ของอาจารย์ที่เตรียมไว้ให้ ซึ่งมีรายละเอียดคร่าว ๆ ดังต่อไปนี้

- Class TicketNode ทำหน้าที่ บรรจุ List ของ คนที่ถือตั๋วหนังต่อแถวรับของที่ระลึกก่อนเข้าโรง โดยมีข้อมูลไอดีตั๋วหนังและที่นั่ง
- Class GiftNode ทำหน้าที่ บรรจุข้อมูลของของขวัญที่กองอยู่ในเคาน์เตอร์ โดยมีข้อมูลไอดีและชื่อของที่ระลึก
- Class Queue ทำหน้าที่เป็นโครงสร้างข้อมูล Queue ที่สามารถบรรจุ Node ไว้ตามหลักการ FIFO ที่เรียนในห้อง โดยอาจารย์กำหนดให้ Queue นี้ ต้องสร้างโดยการใช้ Singly Linked List with Tail เท่านั้น มีฟังก์ชันที่เกี่ยวข้องคือ
    - public void push(TicketNode node) ทำหน้าที่ นำ Node เข้าต่อด้านหลังของคิว
    - public void pop() ทำหน้าที่ นำ Node ที่อยู่ด้านหน้าสุดของคิวออกจากคิว โดยไม่ต้อง return Node นั้นออกมา
    - public Node top() ทำหน้าที่ return Node ที่อยู่ด้านหน้าสุดของคิวให้กับ Caller โดยไม่ต้อง Pop Node นั้นออกมา
    - ฟังก์ชันทั้งสามนี้ ไม่สมบูรณ์ คุณต้องแก้ไข/เพิ่มเติมโค้ดให้สามารถทำงานได้
- Class Stack ทำหน้าที่เป็นโครงสร้างข้อมูล Stack ที่สามารถบรรจุ Node ไว้ตามหลักการ LIFO ที่เรียนในห้อง โดยอาจารย์กำหนดให้ Stack นี้ ต้องสร้างโดยการใช้ Singly Linked List without Tail เท่านั้น มีฟังก์ชันที่เกี่ยวข้องคือ
    - public void push(GiftNode node) ทำหน้าที่ นำ Node เข้าต่อด้านบนของสแตก
    - public void pop() ทำหน้าที่ นำ Node ที่อยู่ด้านบนสุดของสแตกออกจากสแตก โดยไม่ต้อง return Node นั้นออกมา
    - public Node top() ทำหน้าที่ return Node ที่อยู่ด้านบนสุดของสแตกให้กับ Caller โดยไม่ต้อง Pop Node นั้นออกมา
    - ฟังก์ชันทั้งสามนี้ ไม่สมบูรณ์ คุณต้องแก้ไข/เพิ่มเติมโค้ดให้สามารถทำงานได้
- Class Major ทำหน้าที่รับข้อมูลลูกศิษย์ที่เข้าศึกษาในสำนัก และประมวลผลโจทย์ ตามที่อาจารย์กล่าวข้างต้น โดยมีฟังก์ชันที่เกี่ยวข้องดังต่อไปนี้
    - public Major(String name) ทำหน้าที่ สร้าง Object ของโรงหนัง Major

- public void addGift(String giftId, String cardName) ทำหน้าที่เพิ่มข้อมูลกองการ์ดในเคาน์เตอร์
- public void addTicket(String tickId, String seatId) ทำหน้าที่เพิ่มข้อมูลของคนที่มาต่อแถว
- public void processAll() ทำหน้าที่ประมวลผลคนที่มาต่อแถวเข้ารับของที่ระลึกจนกว่าคนที่มาต่อแถวรับของที่ระลึกหมดหรือของที่ระลึกหมด
- public void processOne() ทำหน้าที่ประมวลผลคนที่มาต่อแถวเข้ารับของที่ระลึก จำนวน 1 คน ถ้าไม่มีคนต่อแถวหรือของที่ระลึกหมดไม่ต้องแสดงผลอะไร
- public void showStack() ทำหน้าที่แสดงข้อมูลที่อยู่ใน Stack ไม่ต้องแก้ไขฟังก์ชันนี้
- public void showQueue() ทำหน้าที่แสดงข้อมูลที่อยู่ใน Queue ไม่ต้องแก้ไขฟังก์ชันนี้

ตัวอย่างการทำงาน

Java code #1

```
public static void main(String[] args) {
        Major major = new Major("Chiang Mai");
        major.addGift("0001", "Tanjiro");
        major.addTicket("0001", "A1");

        major.showStack();
        major.showQueue();
}
```

Output

```
head -> Gift ID: 0001, Card Name: Tanjiro

head -> Ticket ID: 0001, Seat ID: A1
 -> tail
```

Java code #2

```
public static void main(String[] args) {
        Major major = new Major("Chiang Mai");
        major.addGift("0001", "Tanjiro");
        major.addGift("0002", "Nezuko");
        major.addGift("0003", "Zenitsu");

        major.addTicket("0001", "A1");

        major.showStack();
        major.showQueue();
```

```
        major.processOne();


        major.showStack();
        major.showQueue();
}
```

Output

```
head –> Gift ID: 0003, Card Name: Zenitsu
–> Gift ID: 0002, Card Name: Nezuko
–> Gift ID: 0001, Card Name: Tanjiro

head –> Ticket ID: 0001, Seat ID: A1
 –> tail
Ticket ID: 0001, Seat ID: A1
Gift ID: 0003, Card Name: Zenitsu
Processed one ticket and one gift.
Ticket ID: 0001, Seat ID: A1
Gift ID: 0003, Card Name: Zenitsu
head –> Gift ID: 0002, Card Name: Nezuko
–> Gift ID: 0001, Card Name: Tanjiro

head  –> tail
```

Java code #3

```
public static void main(String[] args) {
        Major major = new Major("Chiang Mai");
        major.addGift("0001", "Tanjiro");
        major.addGift("0002", "Nezuko");
        major.addGift("0003", "Zenitsu");


        major.addTicket("0001", "A1");
        major.addTicket("0021", "B1");


        major.showStack();
        major.showQueue();


        major.processAll();


        major.showStack();
```

```
        major.showQueue();
}
```

Output

```
head –> Gift ID: 0003, Card Name: Zenitsu
–> Gift ID: 0002, Card Name: Nezuko
–> Gift ID: 0001, Card Name: Tanjiro

head –> Ticket ID: 0001, Seat ID: A1
–> Ticket ID: 0021, Seat ID: B1
 –> tail
Processed all tickets and all gifts.
Ticket ID: 0001, Seat ID: A1
Gift ID: 0003, Card Name: Zenitsu
Ticket ID: 0001, Seat ID: A1
Gift ID: 0003, Card Name: Zenitsu
Ticket ID: 0021, Seat ID: B1
Gift ID: 0002, Card Name: Nezuko
Ticket ID: 0021, Seat ID: B1
Gift ID: 0002, Card Name: Nezuko
Finished!!! There is no remaining queue.
head –> Gift ID: 0001, Card Name: Tanjiro

head  –> tail
```

Java code #4

```
public static void main(String[] args) {
        Major major = new Major("Chiang Mai");
        major.addGift("0003", "Zenitsu");

        major.addTicket("0001", "A1");
        major.addTicket("0021", "B1");

        major.showStack();
        major.showQueue();

        major.processAll();

        major.showStack();
        major.showQueue();
}
```

Output

```
head –> Gift ID: 0003, Card Name: Zenitsu


head –> Ticket ID: 0001, Seat ID: A1
–> Ticket ID: 0021, Seat ID: B1
 –> tail
Processed all tickets and all gifts.
Ticket ID: 0001, Seat ID: A1
Gift ID: 0003, Card Name: Zenitsu
Ticket ID: 0001, Seat ID: A1
Gift ID: 0003, Card Name: Zenitsu
The gift stack is empty, but there are still tickets in the queue.
head
head –> Ticket ID: 0021, Seat ID: B1
 –> tail
```

Java code #5

```
public static void main(String[] args) {
        Major major = new Major("Chiang Mai");


        major.addGift("0001", "Doma");
        major.addGift("0002", "Akaza");
        major.addGift("0003", "Kokushibo");
        major.addGift("0004", "Gyutaro");
        major.addGift("0005", "Daki");



        major.addTicket("0001", "A1");
        major.addTicket("0021", "B1");
        major.addTicket("0035", "C10");



        major.showStack();
        major.showQueue();


        major.processAll();
```

```
        major.showStack();
        major.showQueue();
}
```

Output

```
head –> Gift ID: 0005, Card Name: Daki
–> Gift ID: 0004, Card Name: Gyutaro
–> Gift ID: 0003, Card Name: Kokushibo
–> Gift ID: 0002, Card Name: Akaza
–> Gift ID: 0001, Card Name: Doma

head –> Ticket ID: 0001, Seat ID: A1
–> Ticket ID: 0021, Seat ID: B1
–> Ticket ID: 0035, Seat ID: C10
 –> tail
Processed all tickets and all gifts.
Ticket ID: 0001, Seat ID: A1
Gift ID: 0005, Card Name: Daki
Ticket ID: 0001, Seat ID: A1
Gift ID: 0005, Card Name: Daki
Ticket ID: 0021, Seat ID: B1
Gift ID: 0004, Card Name: Gyutaro
Ticket ID: 0021, Seat ID: B1
Gift ID: 0004, Card Name: Gyutaro
Ticket ID: 0035, Seat ID: C10
Gift ID: 0003, Card Name: Kokushibo
Ticket ID: 0035, Seat ID: C10
Gift ID: 0003, Card Name: Kokushibo
Finished!!! There is no remaining queue.
head –> Gift ID: 0002, Card Name: Akaza
–> Gift ID: 0001, Card Name: Doma

head  –> tail
```

Java code #6

```
public static void main(String[] args) {
        Major major = new Major("Phuket");

        major.addGift("0001", "Giyu");
        major.addGift("0002", "Shinobu");
        major.addGift("0003", "Tengen");
```

```
        major.addTicket("0201", "F1");
        major.addTicket("0202", "F2");

        major.showStack();
        major.showQueue();

        major.processOne();



        major.addGift("0004", "Muzan");
        major.addGift("0005", "Doma");

        major.addTicket("0203", "F3");
        major.addTicket("0204", "F4");
        major.addTicket("0205", "F5");



        major.showStack();
        major.showQueue();



        major.processAll();



        major.showStack();
        major.showQueue();
}
```

Output

```
head -> Gift ID: 0003, Card Name: Tengen
-> Gift ID: 0002, Card Name: Shinobu
-> Gift ID: 0001, Card Name: Giyu

head -> Ticket ID: 0201, Seat ID: F1
-> Ticket ID: 0202, Seat ID: F2
 -> tail
Ticket ID: 0201, Seat ID: F1
Gift ID: 0003, Card Name: Tengen
Processed one ticket and one gift.
Ticket ID: 0201, Seat ID: F1
```

Gift ID: 0003, Card Name: Tengen

head –> Gift ID: 0005, Card Name: Doma

–> Gift ID: 0004, Card Name: Muzan

–> Gift ID: 0002, Card Name: Shinobu

–> Gift ID: 0001, Card Name: Giyu

head –> Ticket ID: 0202, Seat ID: F2

–> Ticket ID: 0203, Seat ID: F3

–> Ticket ID: 0204, Seat ID: F4

–> Ticket ID: 0205, Seat ID: F5

–> tail

Processed all tickets and all gifts.

Ticket ID: 0202, Seat ID: F2

Gift ID: 0005, Card Name: Doma

Ticket ID: 0202, Seat ID: F2

Gift ID: 0005, Card Name: Doma

Ticket ID: 0203, Seat ID: F3

Gift ID: 0004, Card Name: Muzan

Ticket ID: 0203, Seat ID: F3

Gift ID: 0004, Card Name: Muzan

Ticket ID: 0204, Seat ID: F4

Gift ID: 0002, Card Name: Shinobu

Ticket ID: 0204, Seat ID: F4

Gift ID: 0002, Card Name: Shinobu

Ticket ID: 0205, Seat ID: F5

Gift ID: 0001, Card Name: Giyu

Ticket ID: 0205, Seat ID: F5

Gift ID: 0001, Card Name: Giyu

Finished!!! There is no remaining queue.

head

head –> tail