204700
Data Structure and Programming Languages

Jakarin Chawachat

From: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-092-introduction-to-programming-in-java-january-iap-2010/index.htm

QUEUE

Queue

Queue เป็นโครงสร้างข้อมูลที่ประกอบด้วยสมาชิกที่เรียงต่อกันเป็นแถว เมื่อมีสมาชิกใหม่เข้าไปเสริมใน queue จะต้องเสริมทางด้านหลัง (rear) และกรณีที่นำสมาชิกออกจากคิวจะต้องนำออกทางด้านหน้า (front)

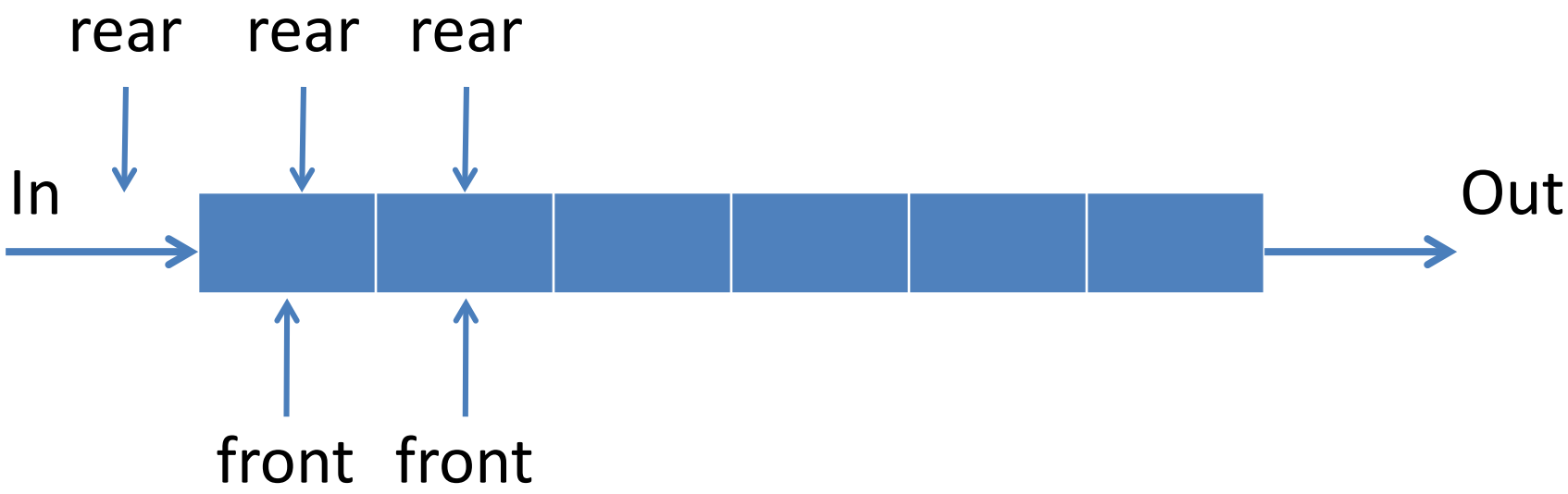
Queue เป็นโครงสร้างข้อมูลแบบเชิงเส้น เช่นเดียวกับ stack แต่มีความแตกต่างตรงที่ queue มีตัวชี้ตำแหน่ง 2 ตัวคือ front และ rear สำหรับการใส่ข้อมูลเข้าและนำข้อมูลออก

ดังนั้น queue จึงมีกระบวนการทำงานเป็นแบบ First In First Out: FIFO หรือ First Come First Serve: FCFS

มีตัวชี้ 2 ตัวในการลำดับคิวคือ front และ rear

front คือตำแหน่งที่ข้อมูลออก จะอยู่ด้านหน้าของคิว

rear คือตำแหน่งที่ข้อมูลจะเข้า จะอยู่ด้านท้ายของคิว



enqueue(3)
enqueue(9)
dequeue()

Queue Operation

- enqueue() เพิ่มข้อมูล
- dequeue() นำข้อมูลออกจาก queue

- isEmpty() ตรวจสอบว่า queue ว่างหรือไม่
- getFront() สอบถามค่าข้อมูลต้น queue

การสร้าง Queue

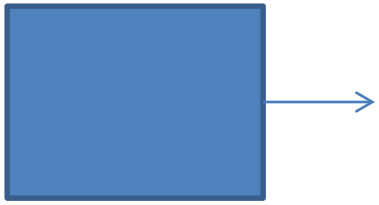
- เราจะสร้าง class สมมติว่าให้ชื่อ MyQueue
- ตอนเริ่มต้นเป็นการกำหนดค่าต่างๆ
  - สร้าง array ตามขนาดที่กำหนด และ กำหนดว่าตอนนี้ Queue ว่าง

```
public class MyQueue {
        int max=10;
        int front =0;
        int rear =-1;
        char arrq[] = new char[max];
        int count=0;
}
```

Method:isEmpty

- isEmpty เป็น method ที่คืนค่าว่าขณะนี้ queue ว่างหรือไม่
- queue ว่างมีลักษณะเป็นอย่างไร

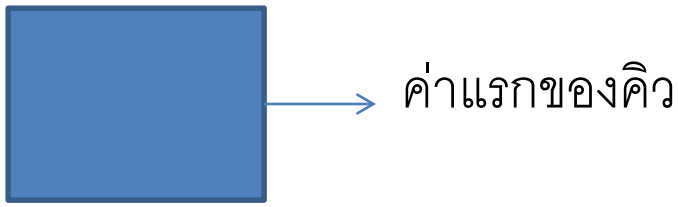
```
boolean isEmpty(){
        if(count==0){
                return true;
        } else {
                return false;
        }
}
```



Method:getFront

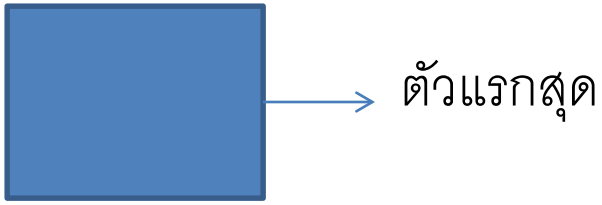
- getFront เป็น method ที่สอบถามว่าตัวต้นของ queue มีค่าเป็นอะไร
- ตัวต้นคิวคือตัวไหน

```
char getFront(){
        char ans='\0';
        if(!isEmpty()){
                ans =  arrq[front];
        }
        return ans;
}
```



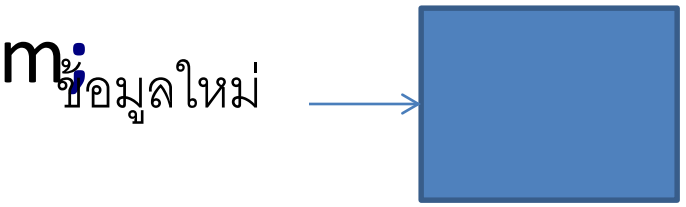
Method:dequeue

```
char dequeue(){
        char ans= '\0';
        if(!isEmpty()){
                ans=arrq[front];
                front++;
                count--;
        }
        return ans;
}
```



Method:enqueue

```
void enqueue(char newItem){
        if(rear+1<=max-1){
                rear++;
                arrq[rear] = newItem;
                count++;
        }
}
```

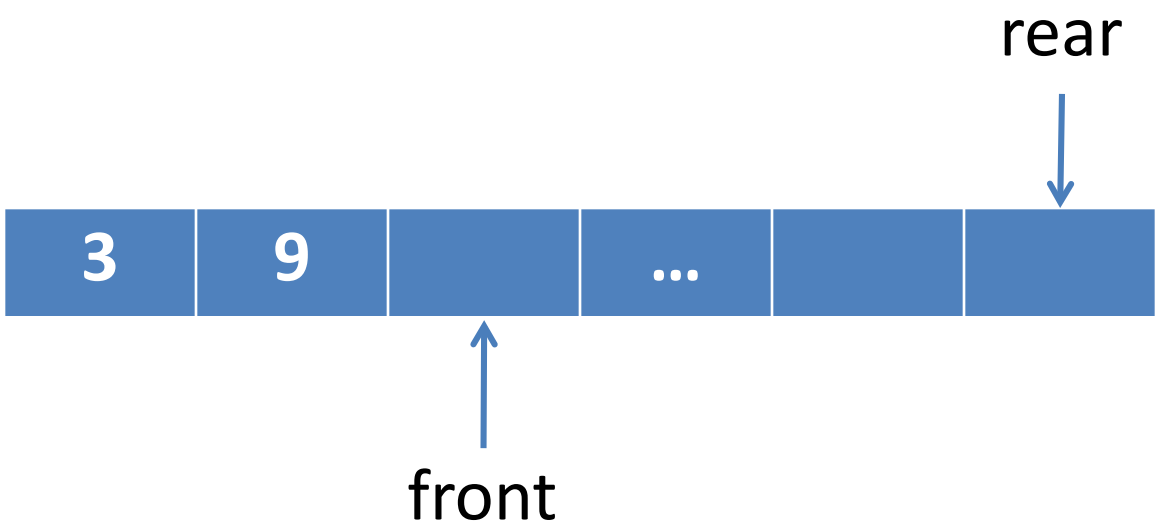


Method:main(6_1)

```
public static void main(String[] args) {
        MyQueue q=new MyQueue();
        q.enqueue('A');
        System.out.print(q.getFront());
        q.enqueue('B');
        System.out.print(q.getFront());
        System.out.print(q.deQueue());
        System.out.print(q.getFront());
}
```

AAAB

Is queue full?

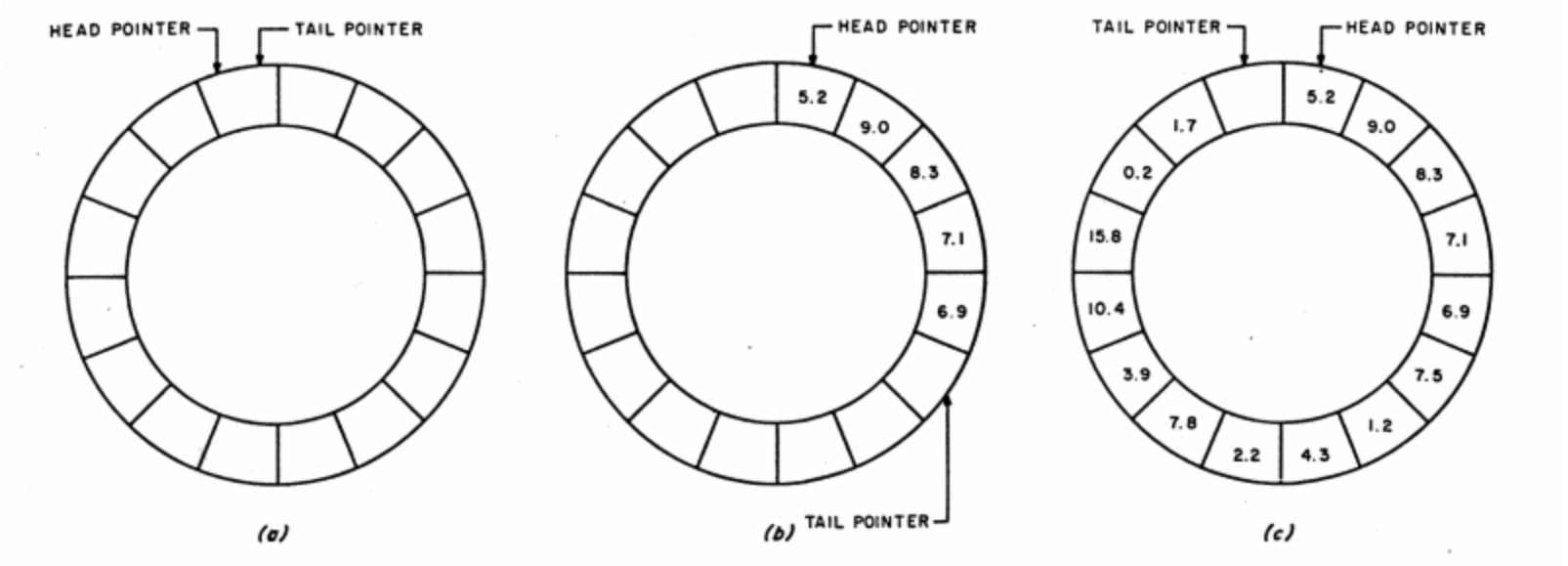


Circular Queue

ถ้ามีการเพิ่มของเข้าไปใน queue จนเต็มแล้ว แต่ว่ามีการนำเอาของออกจาก queue จะพบว่ามีเนื้อที่ว่างของ queue ที่ไม่ได้นำมาใช้ประโยชน์ เมื่อตัวชี้ front ผ่านช่องนั้นไปแล้ว

จึงได้มีการนำปลายของ queue ทั้งสองด้านมาเชื่อมต่อกันในลักษณะวงกลม เรียกว่า **คิววงกลม** (**Circular Queue**)

Circular Queue



Circular Queue Operators

เหมือนเดิม

- isEmpty()
- getFront()

มีการเปลี่ยนแปลง

- enqueue
- dequeue

dequeue

```
char dequeue(){
        char ans= '\0';
        if(!isEmpty()){
                ans=arrq[front];
                front=(front+1)% max;
                count--;
        }
        return ans;
}
```

enqueue

```
void enqueue(char newItem){
        if(count<max){
                rear=(rear+1)% max;
                arrq[rear] = newItem;
                count++;
        }
}
```

Assignment

- Implement: Circular queue (6_2)