

## 자료 구조 Lab 004 :

Lab17004.zip : LabTest.java, lab004.java, lab.in, lab.out, lab004.pdf

---

### 제출

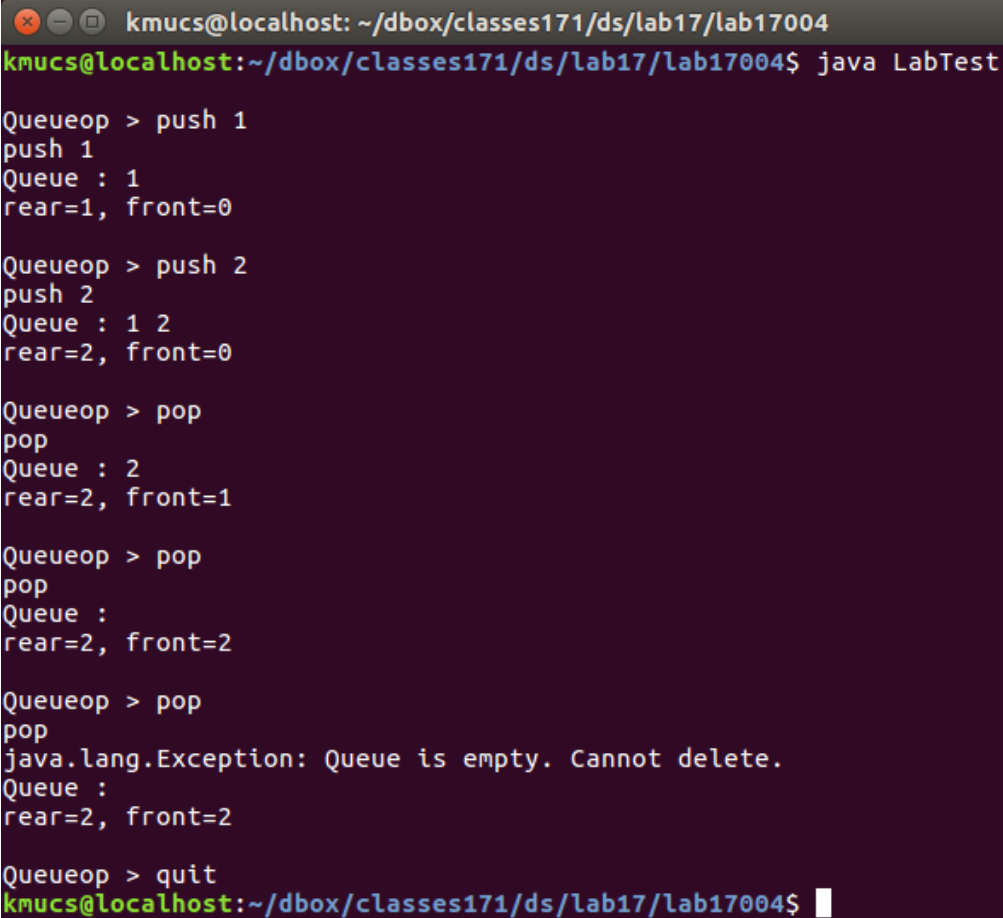
lab004.java 를 학번.java 로 변경하여 이 파일 한 개만 제출할 것.

---

---

이번 Lab은 원형큐 (Circular Queue)를 구현하는 것이다.

수행 예는 다음과 같다.



```
kmucs@localhost: ~/dbox/classes171/ds/lab17/lab17004
kmucs@localhost:~/dbox/classes171/ds/lab17/lab17004$ java LabTest

Queueop > push 1
push 1
Queue : 1
rear=1, front=0

Queueop > push 2
push 2
Queue : 1 2
rear=2, front=0

Queueop > pop
pop
Queue : 2
rear=2, front=1

Queueop > pop
pop
Queue :
rear=2, front=2

Queueop > pop
pop
java.lang.Exception: Queue is empty. Cannot delete.
Queue :
rear=2, front=2

Queueop > quit
kmucs@localhost:~/dbox/classes171/ds/lab17/lab17004$
```

push 명령을 입력하면 원소가 push 되는데, 그 명령을 보여주고, 현재 queue 상태를 보여주고, 마지막으로 rear와 front 변수의 값을 보여준다. 그리고 다음 명령을 입력 받을 준비를 한다.

명령어로 quit를 치면 프로그램에서 빠져 나온다.

이를 위해 구현이 필요한 부분은 lab004.java 파일 내의 아래 4개 함수이다.

```
● □ boolean isEmpty() { }  
    // 원소의 수가 0이면 true를 리턴, 아니면 false를 리턴  
  
● □ void Push (T x) throws Exception { }  
□    // Queue의 rear에 원소를 추가  
  
● □ T Pop() throws Exception { }  
□    // Queue의 front에서 원소를 제거하고, 제거된 원소를 return  
  
● □ public String toString() { }  
□    // Queue의 내용을 front 부터 rear까지 차례로 표시. 이와 더불어 front 변수와 rear 변수의 값도 출력. 실제로는 출력된 내용을 String에 담아서 이를 return함.  
    예) Queue : 1 2  
        rear=2, front=0
```

## 프로그램 테스트

### 컴파일

```
$ javac lab004.java LabTest.java
```

```
// warning 메시지는 무시해도 됨.
```

### 실행

```
$ java LabTest
```

### 주어진 input으로 실행

```
$ java LabTest < lab.in
```

주어진 **output**과 비교

```
$ java LabTest < lab.in > abc
```

```
$ diff abc lab.out
```

또는

```
$ diff -i --strip-trailing-cr -w abc lab.out
```