وزارة التعليم
Ministry of Education
043

جامعة الإمام عبدالرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

المملكة العربية السعودية
Kingdom of Saudi Arabia

Lab 9: Queue (using Array)

Objective(s)

- Types of Queue
- Queue operations
- Queue implementation using Array (Linear and Circular)

Tool(s)/Software

Java programming language with NetBeans IDE.

Description

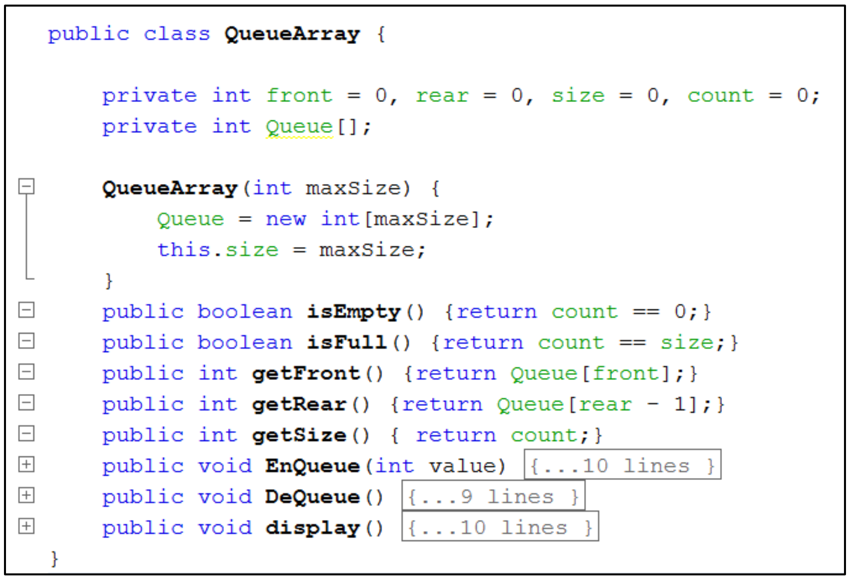
Types of Queue

- Linear Queue
- Circular Queue
- Double Ended Queue
- Priority Queue

Queue Main Operations:

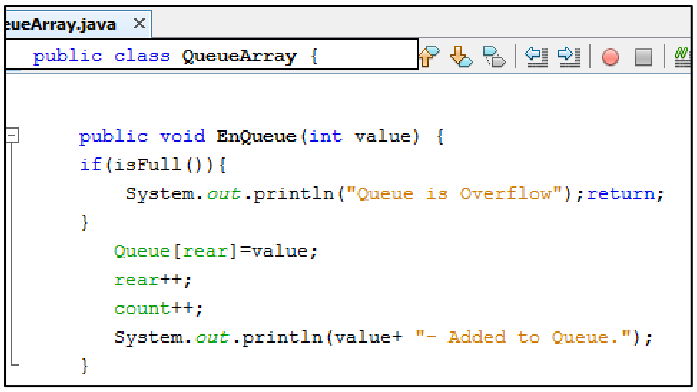
- Enqueue – To insert an element in Queue (at the rear of the queue)
- Dequeue – To delete an element from Queue (from the front of the queue)

A. **Linear Queue implementation using array (*QueueArray.java*):**

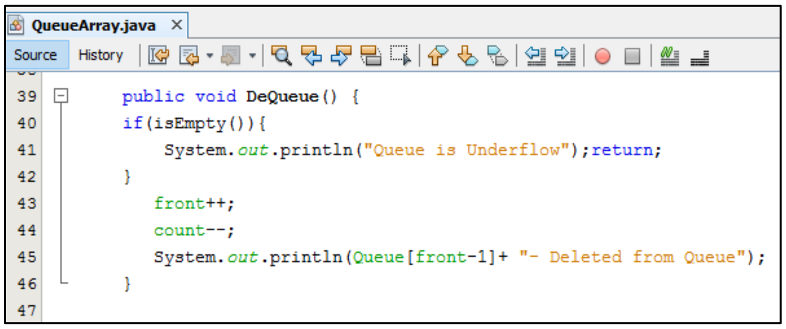


```
public class QueueArray {

    private int front = 0, rear = 0, size = 0, count = 0;
    private int Queue[];

    QueueArray(int maxSize) {
        Queue = new int[maxSize];
        this.size = maxSize;
    }
    public boolean isEmpty() {return count == 0;}
    public boolean isFull() {return count == size;}
    public int getFront() {return Queue[front];}
    public int getRear() {return Queue[rear - 1];}
    public int getSize() { return count;}
    public void EnQueue(int value) {...10 lines }
    public void DeQueue() {...9 lines }
    public void display() {...10 lines }
}
```

Enqueue and Dequeue in ***QueueArray*** class:

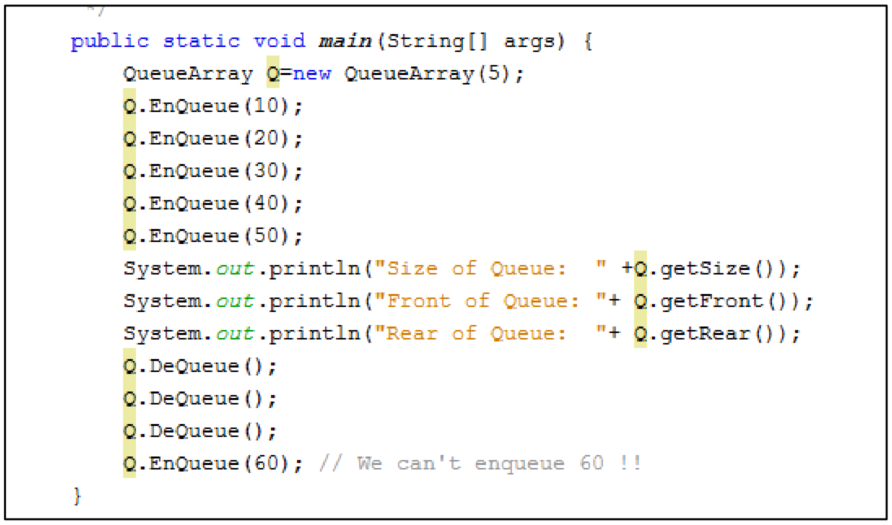


```
public class QueueArray {

    public void EnQueue(int value) {
    if(isFull()){
        System.out.println("Queue is Overflow");return;
    }
       Queue[rear]=value;
       rear++;
       count++;
       System.out.println(value+ "- Added to Queue.");
    }
```



```
    public void DeQueue() {
    if(isEmpty()){
        System.out.println("Queue is Underflow");return;
    }
        front++;
        count--;
        System.out.println(Queue[front-1]+ "- Deleted from Queue");
    }
```

In the main, create object and try the queue operations:



```
public static void main(String[] args) {
    QueueArray Q=new QueueArray(5);
    Q.EnQueue(10);
    Q.EnQueue(20);
    Q.EnQueue(30);
    Q.EnQueue(40);
    Q.EnQueue(50);
    System.out.println("Size of Queue:  " +Q.getSize());
    System.out.println("Front of Queue: "+ Q.getFront());
    System.out.println("Rear of Queue:  "+ Q.getRear());
    Q.DeQueue();
    Q.DeQueue();
    Q.DeQueue();
    Q.EnQueue(60); // We can't enqueue 60 !!
}
```

☒ **<u>The problem with this implementation is:</u>**

Suppose that the array size is 5, and 5 calls to EnQueue() have been made, now the queue array is full and Rear is 5.

front Rear=5

0	1	2	3	4
10	20	30	40	50

Assume 3 calls to DeQueue( ) are made:

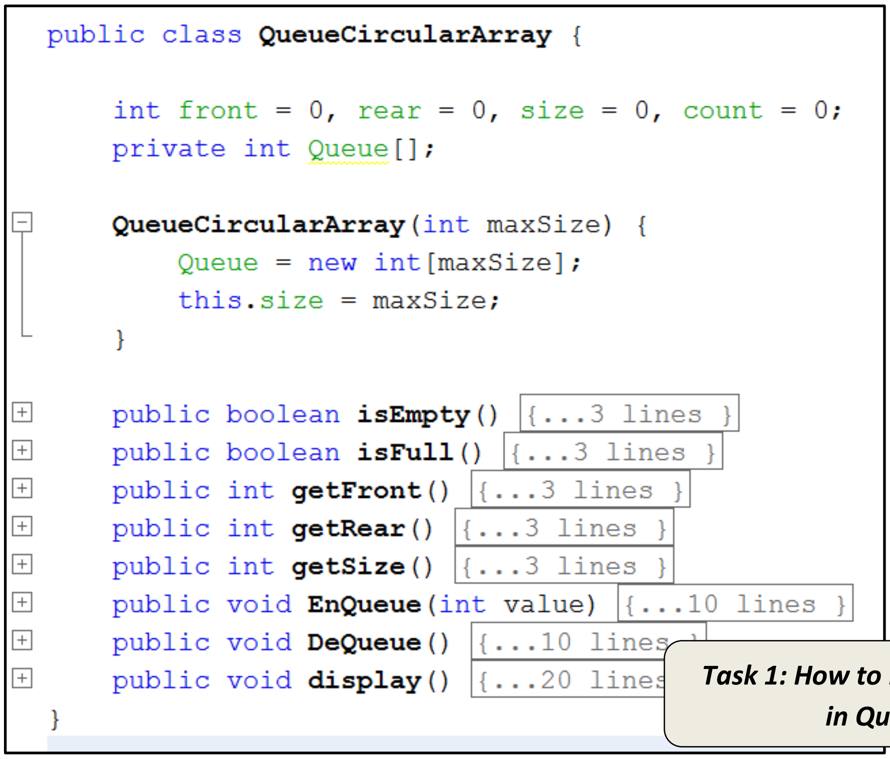
front Rear=5

0	1	2	3	4
			40	50

The queue is not full, but it is WRONG to add at position indexed by Rear as it is out of bound!

How can we solve this problem?!

B. **<u>Circular Queue implementation using array (*QueueCircularArray.java*):</u>**



```
public class QueueCircularArray {

    int front = 0, rear = 0, size = 0, count = 0;
    private int Queue[];

    QueueCircularArray(int maxSize) {
        Queue = new int[maxSize];
        this.size = maxSize;
    }

    public boolean isEmpty() {...3 lines }
    public boolean isFull() {...3 lines }
    public int getFront() {...3 lines }
    public int getRear() {...3 lines }
    public int getSize() {...3 lines }
    public void EnQueue(int value) {...10 lines }
    public void DeQueue() {...10 lines }
    public void display() {...20 lines
}
```

Task 1: How to Implement display () method in QueueCircularArray ?

Enqueue and Dequeue in ***QueueCircularArray*** *class*:

```
public void EnQueue(int value) {
    if (isFull()) {
        System.out.println("Queue is Overflow");
        return;
    }
    Queue[rear] = value;
    rear = (rear + 1) % size;
    count++;
    System.out.println(value + "- Added to Queue.");
}
```

```
public void DeQueue() {
    if (isEmpty()) {
        System.out.println("Queue is Underflow");
        return;
    }
    System.out.println(Queue[front] + "- Deleted from Queue");
    front = (front + 1) % size;
    count--;

}
```

Tasks/Assignments(s)

1. Create ***QueueCircularArray*** class in Java that implements Circular Queue using Array and implement all the following methods: *enqueue, dequeue, getSize, getFront, getRear, isFull, isEmpty* and ***display***. In the main, create object **myQueue** from ***QueueCircularArray*** with the size of 5. Then do the followings:

a. Enqueue the values (10,20,30,40,50) and call **display** method to display queue elements.
b. Dequeue 3 elements and call **display** method to display queue elements.
c. Enqueue (60) and call **display** method to display queue elements.

Deliverables(s)

You are required to implement and deliver a Java program(s) as described in the previous section.