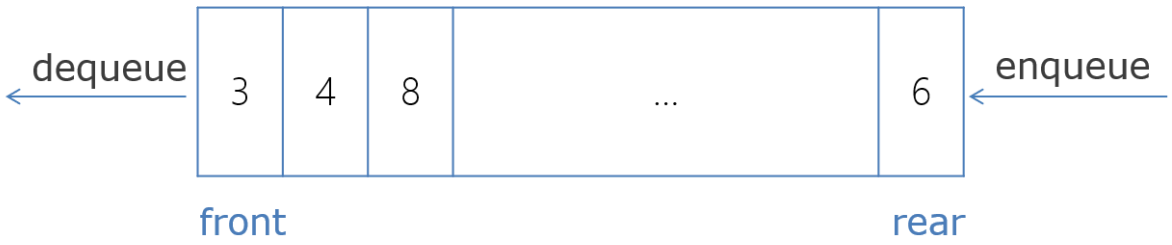
队列 Queues

队列是链表的另一种特殊形式，它可以模拟现实的队列进出逻辑来对数据进行存储处理。

简介

队列可以看作有前后两扇门，一个出口一个入口。入口用于元素入队，出口用于元素出队。因此元素总是先进先出后进后出。

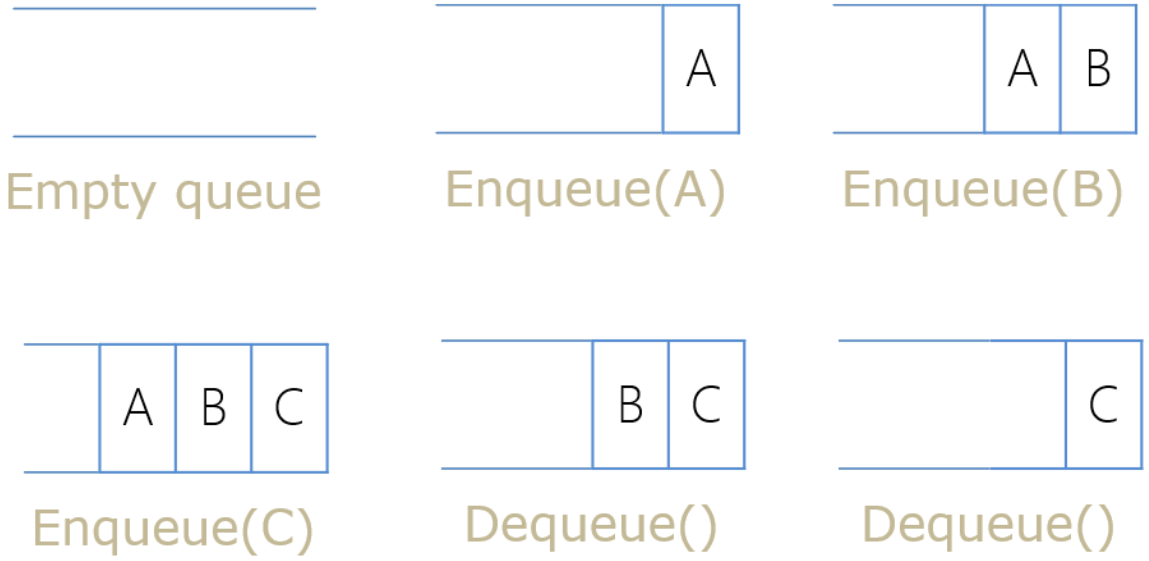


队列

操作

队列的基本操作的出队 (Dequeue) 与入队 (Enqueue) 。

- 入队
  > 在队列后部插入一个元素
- 出队
  > 从队列前部移除一个元素

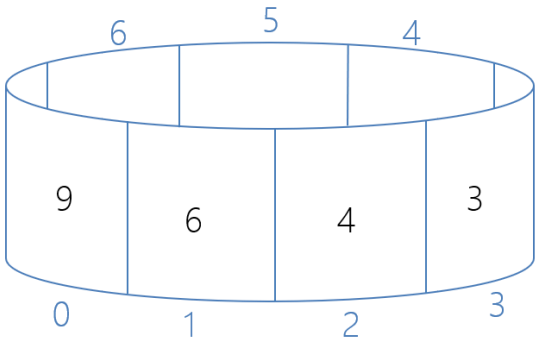


出队与入队

实现

在空间有限的情况之下，实现入队与出队可以选择的最佳方式是使用一个环状数组 array 来模拟。

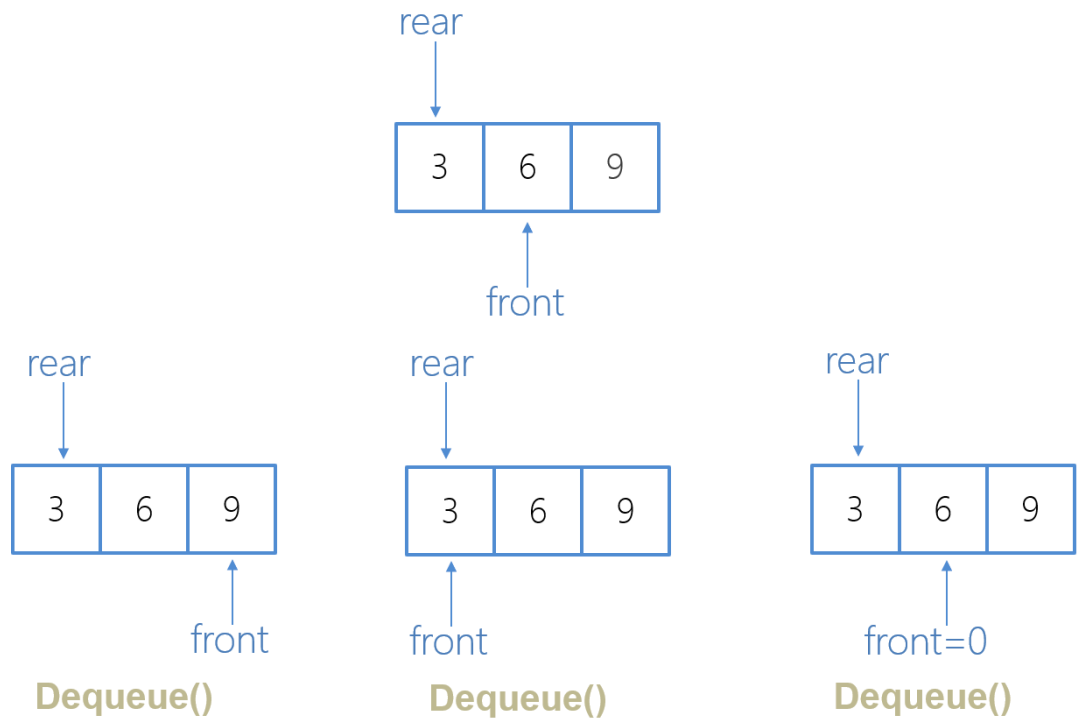
- Circular array
  - When an element moves past the end of a circular array, it wraps around to the beginning



Index Growth: 0 → 1 → 2 → 3 → 4 → 5 → 6 → 0 → ...

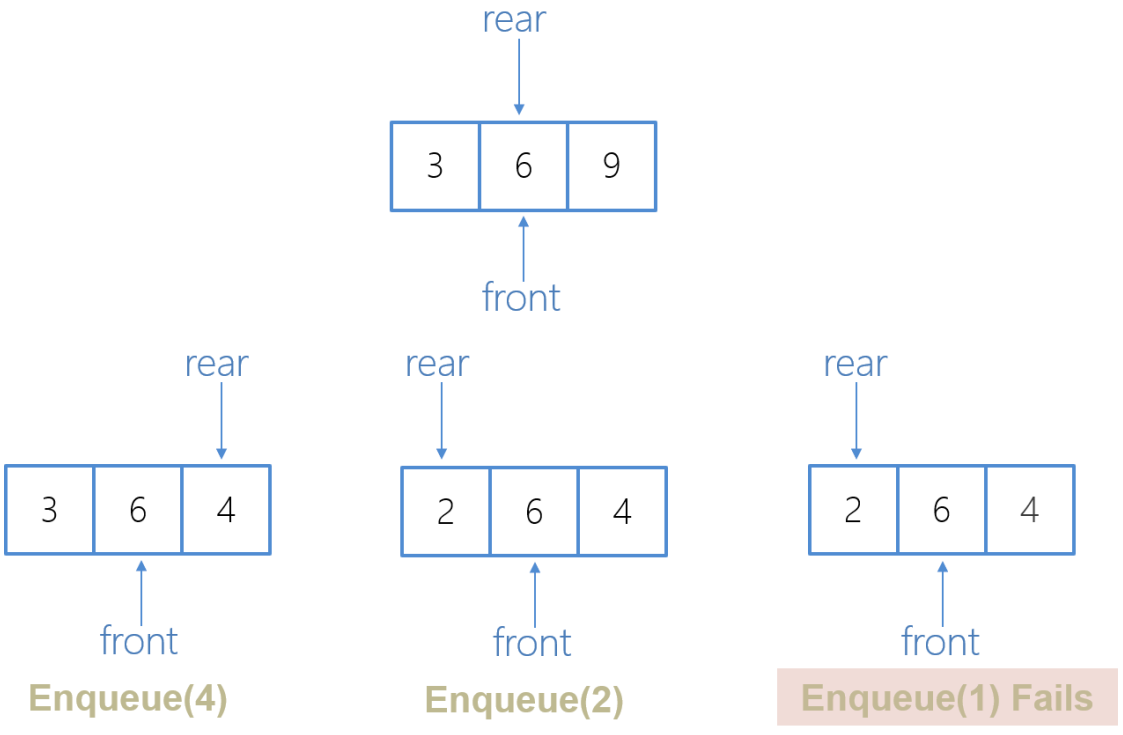
环状数组模拟

因此，用 L 代替队列的长度，可以把后端设为 $rear$，前端设为 $front$，让 $front$ 从 0 开始，让 $rear$ 从 -1 开始。$rear$ 的位置可以是 $(rear + 1)\%L$，front 则是 $(front + 1)\%L$。相应地，还可以用 $counter$ 代表计数器，来计算队列中元素的数量，来判断队列是否已满。


rear
3
6
9
front
rear
3
6
9
front
Dequeue()
rear
3
6
9
front
Dequeue()
rear
3
6
9
front=0
Dequeue()

16

出队


rear
3
6
9
front
rear
3
6
4
front
Enqueue(4)
rear
2
6
4
front
Enqueue(2)
rear
2
6
4
front
Enqueue(1) Fails

17

入队

代码实现

```
/*Add some comments by yourself.*/
public class Queue {
    private Double[] values;
    private int front, rear, counter;
    public Queue(int size) {
        values = new Double[size];
        front = 0;
        rear = -1;
        counter = 0;
    }
    public boolean isEmpty() {
        return counter == 0;
    }
    public boolean isFull() {
        return counter == values.length;
    }
    public Double enqueue(double x) {
        if(isFull())
            return null;
        rear = (rear + 1) % values.length;
        values[rear] = Double.valueOf(x);
        counter ++;
        return values[rear];
    }
    public Double dequeue() {
        if(isEmpty())
            return null;
        int oldFront = front;
        front = (front + 1) % values.length;
        counter --;
        return values[oldFront];
    }
    public void displayQueue() {
        if(isEmpty()) {
```

```
            System.out.println("Empty queue!");
            return;
        }
        System.out.print("front->");
        for(int i = front; i < front + counter; i ++)
            if(i != front + counter -1)
                System.out.println("\t|\t " + String.format("%,
.4f", values[i % values.length].doubleValue()) + "\t|");
            else
                System.out.println("\t|\t " + String.format("%,
.4f", values[i % values.length].doubleValue()) + "\t|<-rear");
    }
}
```