顺序队列

什么是队列?

队列（**Queue**）是一种特殊的线性表，**插入**和**删除**分别在**队头（Front）**和**队尾（Rear）**进行，没有元素的队列称为空队列。

先进先出，插入元素称为**add**，删除元素称为**Pop**。

抽象数据类型ADT

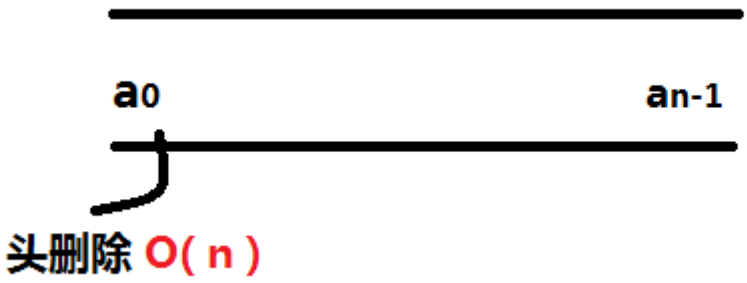
接口

```
package DS;
public interface Queue<T> {
    public abstract boolean isEmpty();
    public abstract boolean add(T x);//入队返回布尔值
    public abstract T peek();//返回队头元素或者null
    public abstract T poll();//出队返回对头元素或者null
}
```

顺序队列

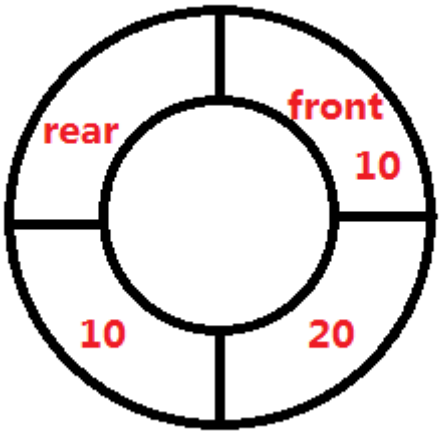
顺序队列有三种实现方式：

1.使用顺序表，出队还好说（O（1）），但是入队每一次都是一次顺序表的删除操作，效率极低（O（n））。



2.使用数组，不移动元素，用front和rear来表示头和尾，效率很好但是会造成伪溢出。

3.顺序循环队列



第一个：

$$front = 0,\ rear = 1$$

入队列：

$$rear = （rear + 1）\%length$$

出队列：

$$front = （front + 1）\%length$$

为 空：

$$front = rear$$

为 满：

$$front = （rear + 1）\%length$$

> 这里的**front**是指第一个元素，但是**rear**指的是最后一个元素的下一个元素，为什么是最后一个元素的下一个元素？如果不这样写的话，判断队列为空之后会变得非常的麻烦，当队列里还剩一个元素时，**front**和**rear**重合，再删除最后一个时，判断是否为空就会出现一些麻烦。但是同时我们也会失去一个存储空间作为代价。

代码：

```
package DS;
//记住在循环队列中,rear储存的是最后一个元素的下一个元素,而front储存的是第一个元素
public final class SeqQueue<T> implements Queue<T>{
    private Object element[];
    private int front,rear;
    public SeqQueue(int length){
        if(length < 64){
            length = 64;
        }
        this.element = new Object[64];
        this.front = this.rear = 0;

    }
    public SeqQueue(){
        this(64);
    }
```

```
    public boolean isEmpty(){
        return this.rear == this.front;
    }
    public boolean add(T x){
        if(x==null){
            return false;
        }
        if(this.front == (this.rear+1)%this.element.length){
            Object[] temp = this.element;
            this.element = new Object[temp.length*2];
            int j = 0;
            for(int i = this.front;i!=this.rear;i=(i+1)%temp.length){
                //记住这里的判断条件很重要哦。
                this.element[j++] = temp[i];
            }
            this.front = 0;
            this.rear = j;
        }
        this.element[this.rear] = x;
        this.rear = (this.rear+1)%this.element.length;
        //数组拓容之后，front会回到第 0 位。
        return false;
    }
    public T peek(){
        return this.isEmpty()?null:(T)this.element[this.front];
    }
    public T poll(){
        if(this.isEmpty()){
            return null;
        }
        T temp = (T)this.element[this.front];
        this.front = (this.front+1)%this.element.length;
        return temp;
    }
}
```

> 删除和取值的时候只需要判断数组是否为空，但是插入的时候要考虑数组拓容。