public class ArrayQueue <E>implements Queue<E> {  
 ////Q1  
 public void rotate() {  
 if (isEmpty() || size() == 1)  
 return;  
  
 // Dequeue the front element  
 E element = dequeue();  
  
 // Enqueue the dequeued element  
 enqueue(element);  
 }  
 //Q2  
 @Override  
 public ArrayQueue<E> clone() {  
 ///System.arraycopy copies the elements from the source array (this.data) to the destination array (clonedQueue.data).  
 // By copying the entire array, we create a new independent copy of the elements.  
 ArrayQueue<E> clonedQueue = new ArrayQueue<>(data.length);  
  
 clonedQueue.sz = this.sz;  
 clonedQueue.f = this.f;  
  
 // Copy the elements from the original queue to the cloned queue  
 System.arraycopy(this.data, 0, clonedQueue.data, 0, this.data.length);  
  
 return clonedQueue;  
 }  
 ///Q3  
  
 private E data[];  
 private int sz=0;  
 private int f=0;  
 public ArrayQueue(int cap)  
 {  
 data= (E[])new Object[cap];  
 }  
 @Override  
 public int size() {  
 return sz;  
 }  
  
 @Override  
 public boolean isEmpty() {  
 return sz==0;  
 }  
  
 @Override  
 public void enqueue(E e) {  
if (size()== data.length)  
 throw new IllegalStateException("Queue is full");  
int x=(f+sz)% data.length;  
data[x]=e;  
sz++;  
  
  
 }  
  
 @Override  
 public E dequeue() {  
 if (isEmpty())return null;  
 E deleted=data[f];  
 data[f]=null;  
 f=(f+1)% data.length;  
 sz--;  
 return deleted;  
 }  
  
 @Override  
 public E first() {  
 if (isEmpty())return null;  
 return data[f];  
 }  
}