Outline and Reading

- The Queue ADT (§2.1.2)
- Implementation with a circular array (§2.1.2)
- Growable array-based queue
- Queue interface in Java
The Queue ADT

- The Queue ADT stores arbitrary objects
- Insertions and deletions follow the first-in first-out scheme
- Insertions are at the rear of the queue and removals are at the front of the queue
- Main queue operations:
  - enqueue(object): inserts an element at the end of the queue
  - dequeue(): removes and returns the element at the front of the queue
- Auxiliary queue operations:
  - object front(): returns the element at the front without removing it
  - integer size(): returns the number of elements stored
  - boolean isEmpty(): indicates whether no elements are stored
- Exceptions
  - Attempting the execution of dequeue or front on an empty queue throws an EmptyQueueException
Applications of Queues

Direct applications
- Waiting lists, bureaucracy
- Access to shared resources (e.g., printer)
- Multiprogramming

Indirect applications
- Auxiliary data structure for algorithms
- Component of other data structures
Array-based Queue

- Use an array of size $N$ in a circular fashion
- Two variables keep track of the front and rear
  - $f$ index of the front element
  - $r$ index immediately past the rear element
- Array location $r$ is kept empty

**normal configuration**

```
Q
0 1 2  f  r
```

**wrapped-around configuration**

```
Q
0 1 2  r  f
```
Queue Operations

We use the modulo operator (remainder of division)

Algorithm `size()`
return $(N - f + r) \mod N$

Algorithm `isEmpty()`
return $(f = r)$
Queue Operations (cont.)

Operation enqueue throws an exception if the array is full.
This exception is implementation-dependent.

Algorithm `enqueue(o)`
if `size() = N - 1` then
    throw `FullQueueException`
else
    \[ Q[r] \leftarrow o \]
    \[ r \leftarrow (r + 1) \mod N \]
Queue Operations (cont.)

- Operation dequeue throws an exception if the queue is empty.
- This exception is specified in the queue ADT.

Algorithm dequeue()

if isEmpty() then
    throw EmptyQueueException
else
    o ← Q[f]
    f ← (f + 1) mod N
    return o

```
Q  0 1 2  f  r
```

```
Q  0 1 2  r  f
```
Growable Array-based Queue

In an enqueue operation, when the array is full, instead of throwing an exception, we can replace the array with a larger one.

Similar to what we did for an array-based stack.

The enqueue operation has amortized running time:
- \( O(n) \) with the incremental strategy
- \( O(1) \) with the doubling strategy
Queue Interface in Java

- Java interface corresponding to our Queue ADT
- Requires the definition of class EmptyQueueException
- No corresponding built-in Java class

```java
public interface Queue {
    public int size();
    public boolean isEmpty();
    public Object front() throws EmptyQueueException;
    public void enqueue(Object o);
    public Object dequeue() throws EmptyQueueException;
}
```