Lecture 09

Queue

The Abstract Data Type Queue

- A queue
  - New items enter at the back, or rear, of the queue
  - Items leave from the front of the queue
  - First-in, first-out (FIFO) property
    - The first item inserted into a queue is the first item to leave

The Abstract Data Type Queue

- ADT queue operations
  - Create an empty queue
  - Determine whether a queue is empty
  - Add a new item to the queue
  - Remove from the queue the item that was added earliest
  - Remove all the items from the queue
  - Retrieve from the queue the item that was added earliest

Simple Applications of the ADT Queue: Reading a String of Characters

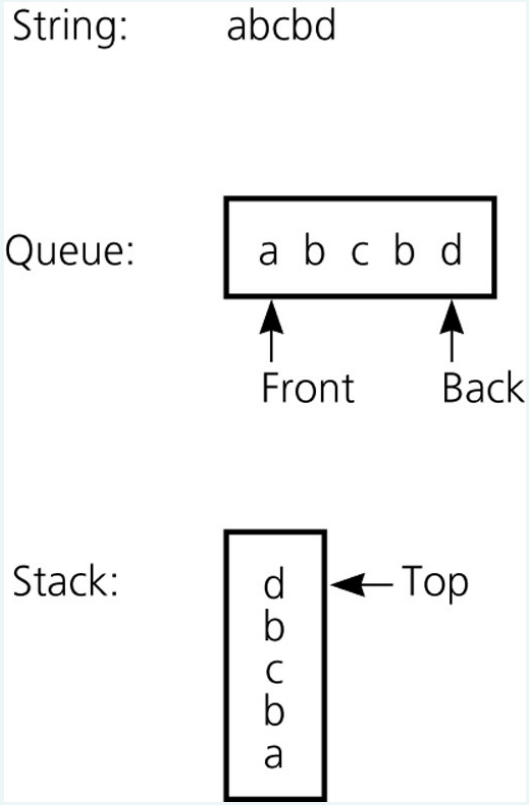
- A queue can retain characters in the order in which they are typed

```
Queue queue = new Queue();
while (not end of line) {
   Read a new character ch
   queue.enqueue(ch)
}
```

- Once the characters are in a queue, the system can process them as necessary

Recognizing Palindromes

- A nonrecursive recognition algorithm for palindromes
  - As you traverse the character string from left to right, insert each character into both a queue and a stack
  - Compare the characters at the front of the queue and the top of the stack



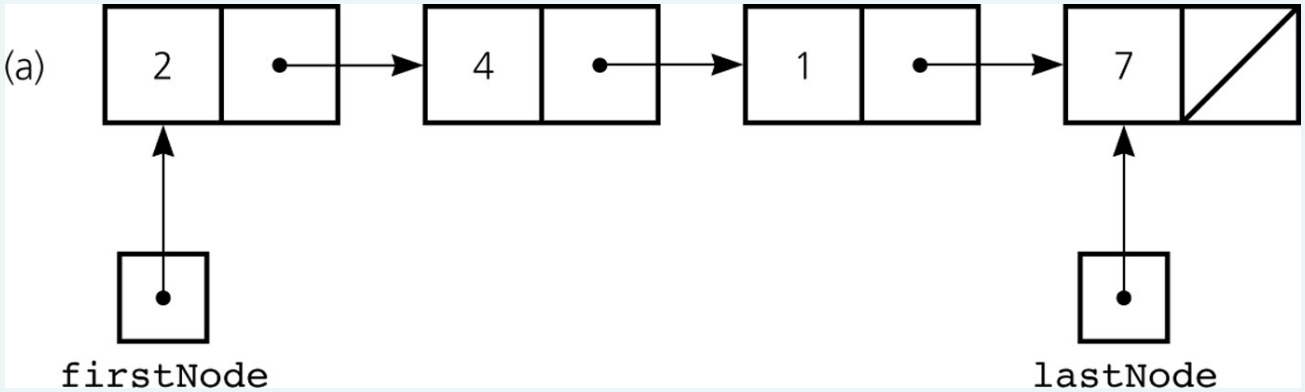
The results of inserting a string into both a queue and a stack

Implementations of the ADT Queue

- A queue can have either
  - An array-based implementation
  - A reference-based implementation

A Reference-Based Implementation

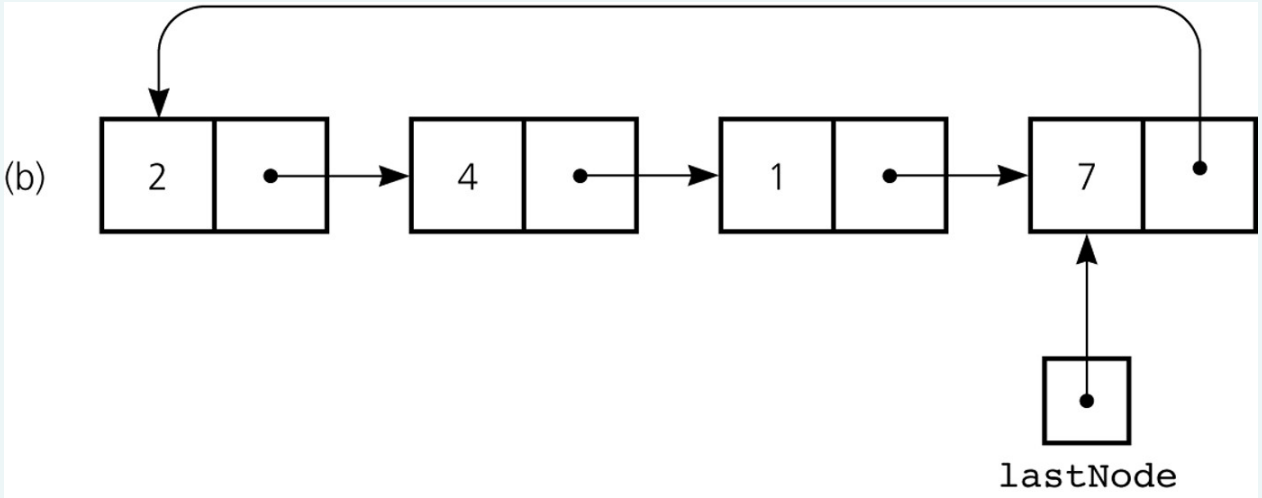
- Possible implementations of a queue
  - A linear linked list with two external references
    - A reference to the front
    - A reference to the back



A reference-based implementation of a queue: a) a linear linked list with two external references

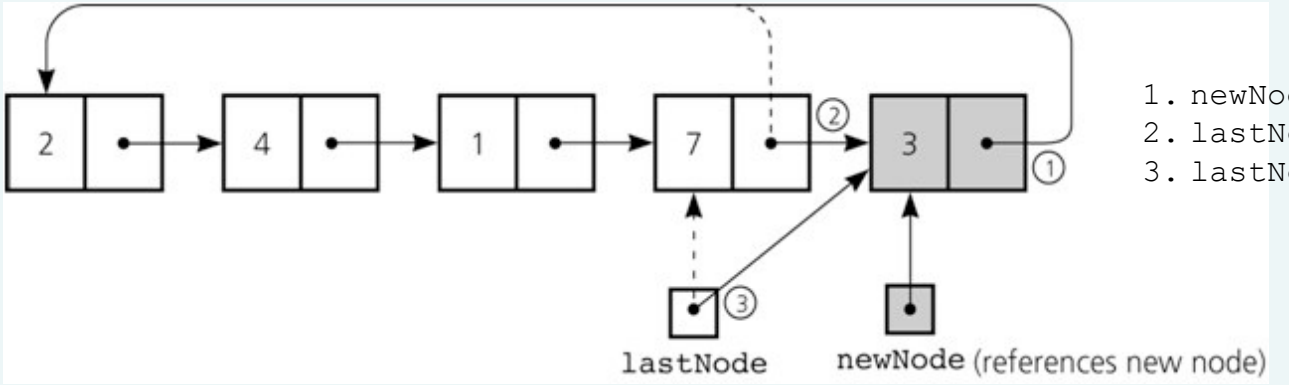
A Reference-Based Implementation

- Possible implementations of a queue (Continued)
  - A circular linked list with one external reference
    - A reference to the back



A reference-based implementation of a queue: b) a circular linear linked list with one external reference

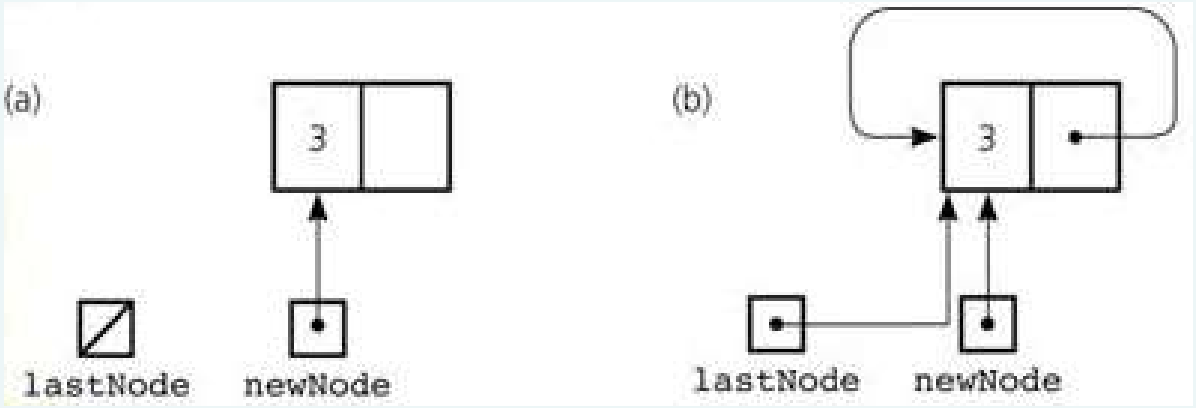
A Reference-Based Implementation




```
1. newNode.next = lastNode.next;
2. lastNode.next = newNode;
3. lastNode = newNode;
```

Inserting an item into a nonempty queue

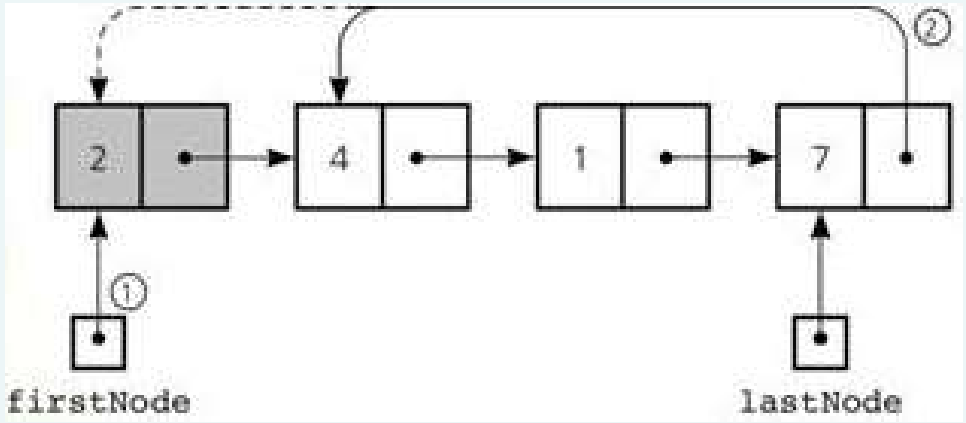
A Reference-Based Implementation




```
newNode.next = newNode;
lastNode = newNode;
```

Inserting an item into an empty queue: a) before insertion; b) after insertion

A Reference-Based Implementation




```
1. firstNode = lastNode.next;
2. lastNode.next = firstNode.next;
```

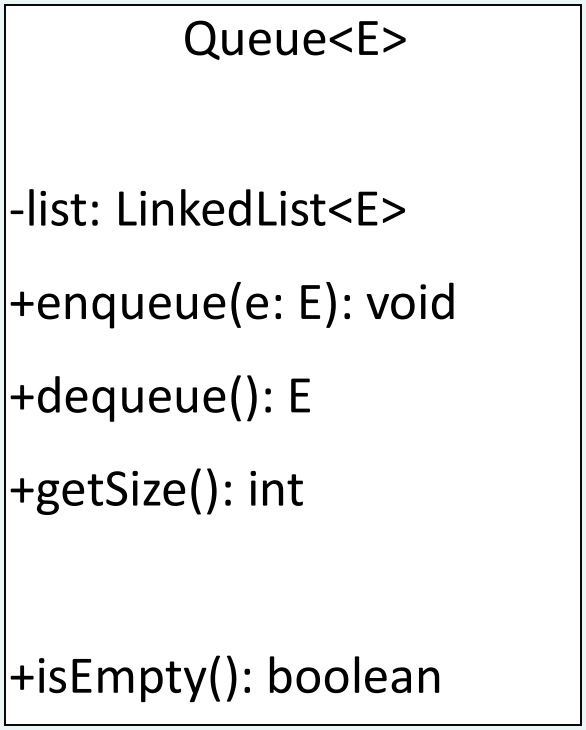
Deleting an item from a queue of more than one item

A Reference-Based Implementation

Using inheritance

Using composition

A Reference-Based Implementation



Adds an element to this queue.

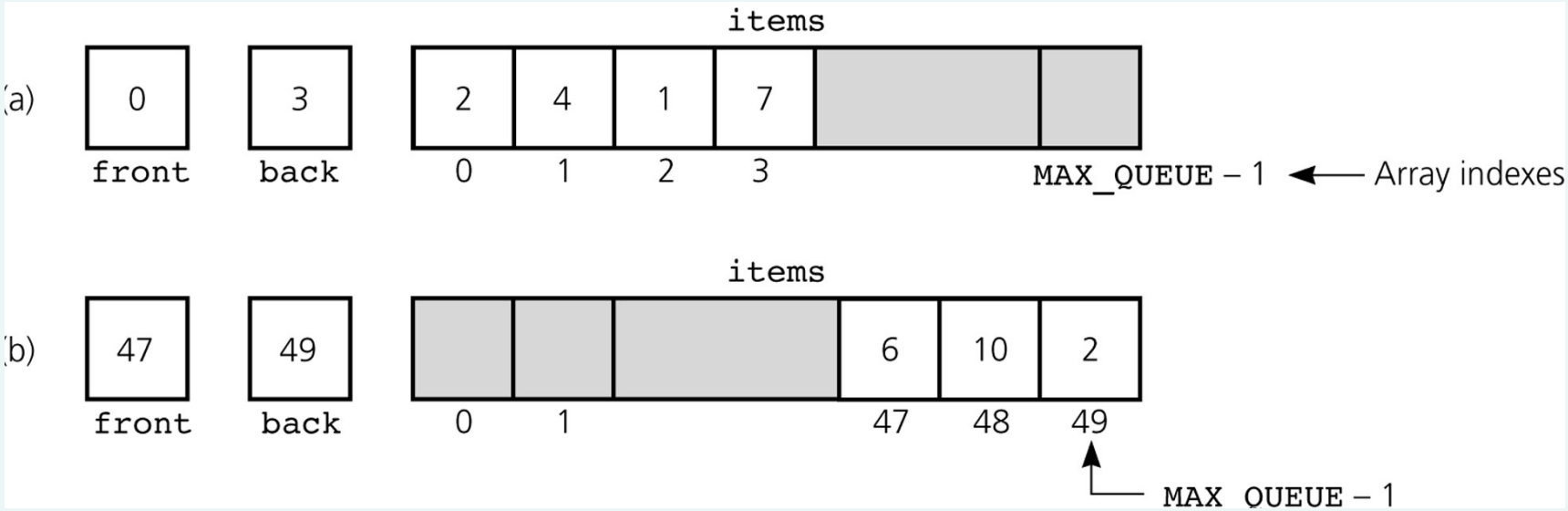
Removes an element from this queue.

Returns the number of elements from this queue.

Returns true if queue is empty, otherwise false

GenericQueue

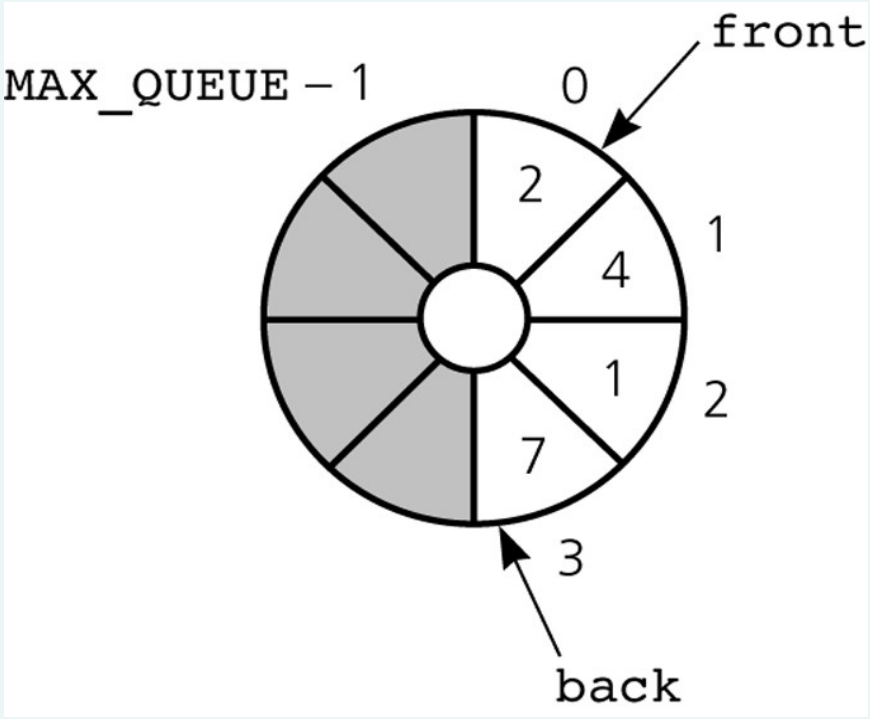
An Array-Based Implementation



a) A naive array-based implementation of a queue; b) rightward drift can cause the queue to appear full

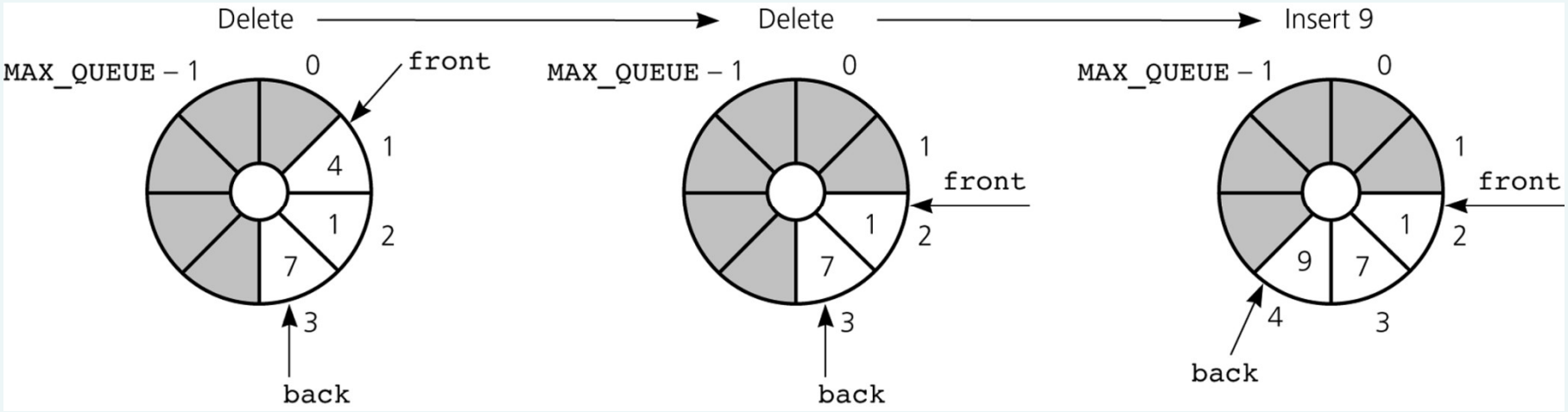
An Array-Based Implementation

- A circular array eliminates the problem of rightward drift



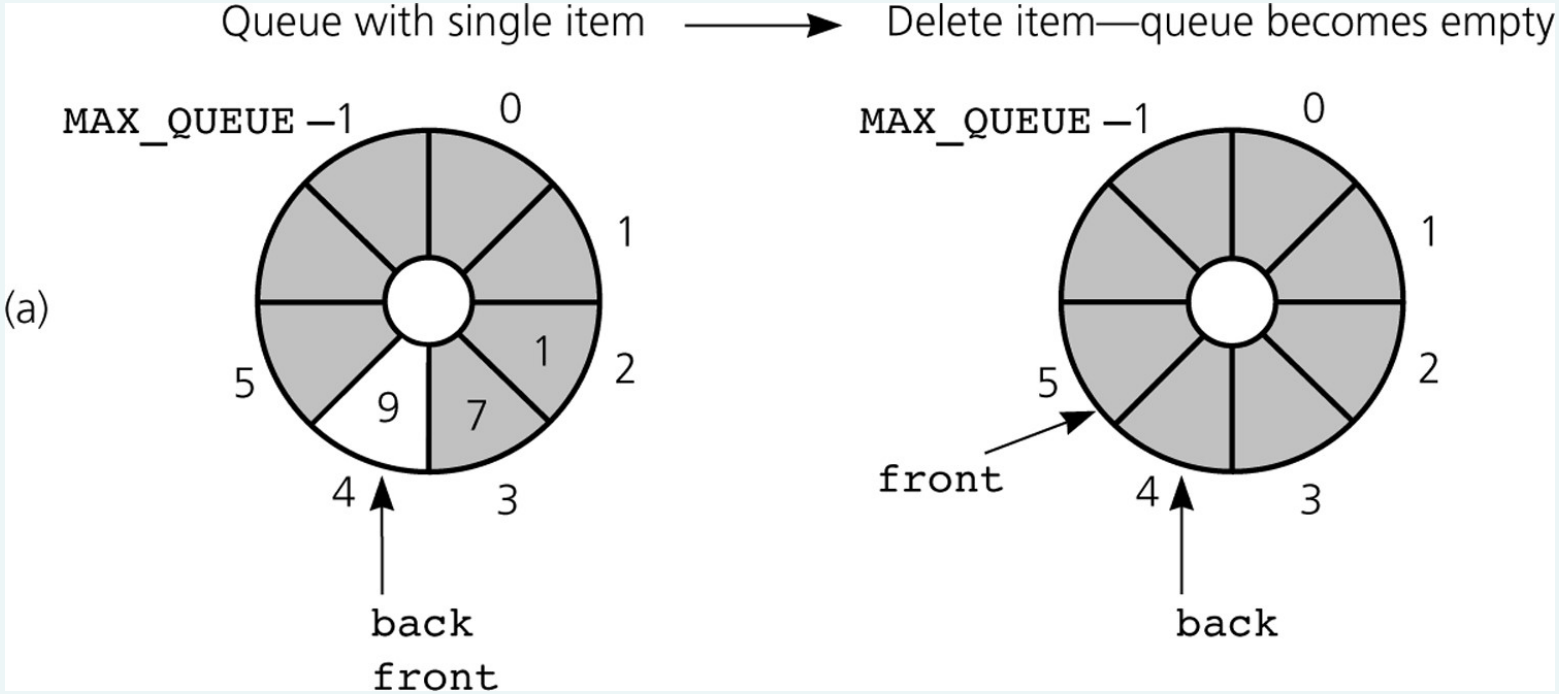
A circular implementation of a queue

An Array-Based Implementation



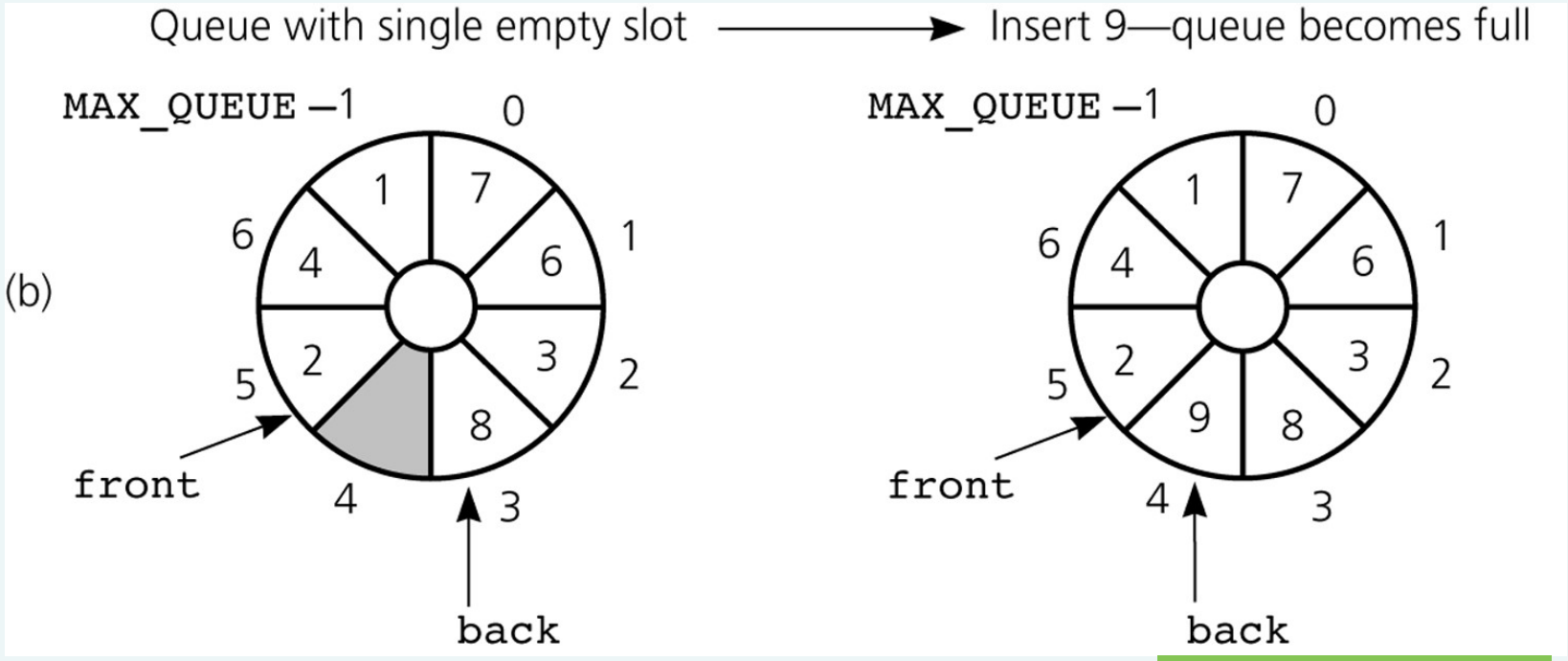
The effect of some operations of the queue

An Array-Based Implementation



a) ***front*** passes ***back*** when the queue becomes empty

An Array-Based Implementation



b) ***back*** catches up to ***front*** when the queue becomes full

GenericQueue

TestGenericQueue

The Java Collections Framework Interface `Queue`

- JCF has a queue interface called **`Queue`**
- Derived from interface **`Collection`**
- Adds methods:
  - `element:` retrieves, but does not remove head
  - `offer:` inserts element into queue
  - `peek:` retrieves, but does not remove head
  - `poll:` retrieves and removes head
  - `remove:` retrieves and removes head