EASTERN WASHINGTON UNIVERSITY
start something big

Queue 1

Computer Science Department
Eastern Washington University
Yun Tian (Tony) Ph.D.

Recall

- Stack
  - first in last out
  - two representations
    - array based
    - linked list
  - applications
    - Infix and postfix expression
    - Parens matching

Today Class

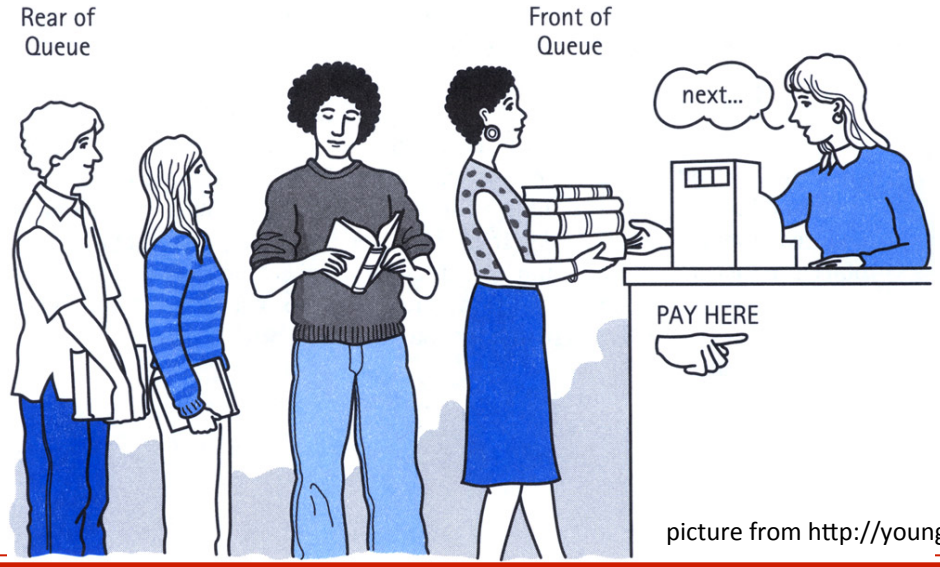
- Queue ADT is very similar to Stack ADT
- Concept of Queue

The Queue ADT

- The Queue ADT stores arbitrary objects.
- Insertions and deletions follow the **first-in first-out** scheme.
  - Element can be inserted at any time.
  - Element that has been in the queue the longest can be removed at any time.
  - Elements enter a queue at the rear and are removed from the front.

The Queue ADT

- Think of **people in a waiting line** to get on an amusement park ride.
  - people wait for such as ride enter at the **rear** of the line.
  - get on the ride(exit the line) from the **front** of the line.



picture from http://younginc.site11.com/source/5895/fos0039.html

The Queue ADT

- Compared with Stack ADT
- Similarity:
  - Linear Data Structure
  - Store arbitrary type of items
  - Two representations: array-based or linked list
  - All operations take O(1) time.
- Dissimilarity
  - Stack: First-in-last-out and operation restricted on one end (the top).
  - Queue: First-in-first-out, enter queue at the rear end and exit queue at the front end.

The Queue ADT

- Main queue operations:
  - **enqueue(object): inserts an element at the end of the queue.**
  - **object dequeue(): removes and returns the element at the front of the queue.**
  - object front(): returns the element at the front without removing it. (or another name peek())
  - integer size(): returns the number of elements stored
  - boolean isEmpty(): indicates whether no elements are stored.

The Queue ADT

- In the Queue ADT, operations **dequeue()** and **front()** cannot be performed if the queue is empty.
  - We should throw an EmptyQueueException object to indicate the error condition.
- Queue is linear in nature,
  - Grows in one direction as a sequence.

Queue Operation Examples

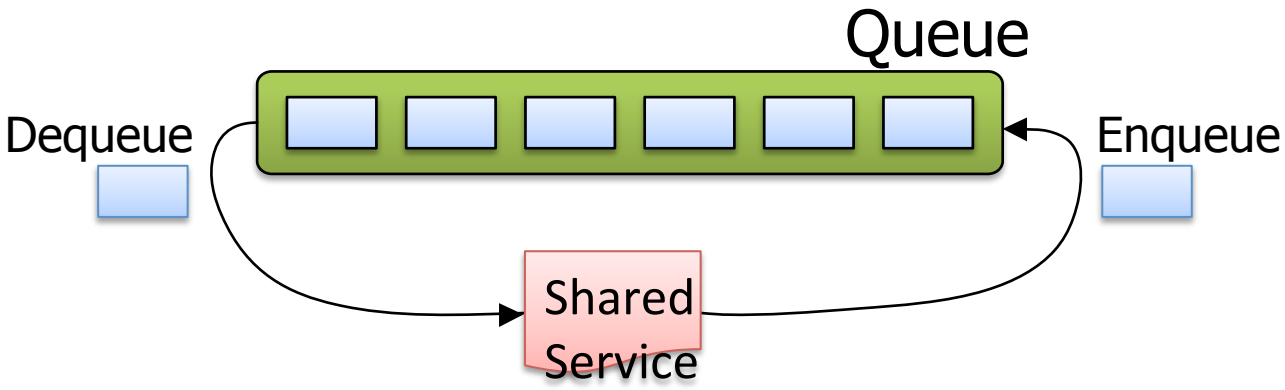
Operation	*Output*	*Q*
enqueue(5)	–	(5)
enqueue(3)	–	(5, 3)
dequeue()	*5*	(3)
enqueue(7)	–	(3, 7)
dequeue()	*3*	(7)
front()	*7*	(7)
dequeue()	*7*	()
dequeue()	*"error"*	()
isEmpty()	*true*	()
enqueue(9)	–	(9)
enqueue(7)	–	(9, 7)
size()	*2*	(9, 7)
enqueue(3)	–	(9, 7, 3)
enqueue(5)	–	(9, 7, 3, 5)
dequeue()	*9*	(7, 3, 5)

Applications of Queue

- Direct applications (first-come-first-served)
  - Waiting lists, waiting lines
  - Access to shared resources (e.g., printer),
  - Process Scheduler in OS
  - Message Queue in Distributed Systems.
  - Connection Queue in Web Server
  - Shared buffer by video downloading thread and video decompression thread.
- Indirect applications
  - Auxiliary data structure for algorithms.
  - Component of other data structures.

Applications: Round Robin Scheduler

- We can implement a round robin scheduler using a queue Q by repeatedly performing the following steps:
  1. e = Q.dequeue()
  2. Service element e
  3. Q.enqueue(e)



Demo of Queue Implementation

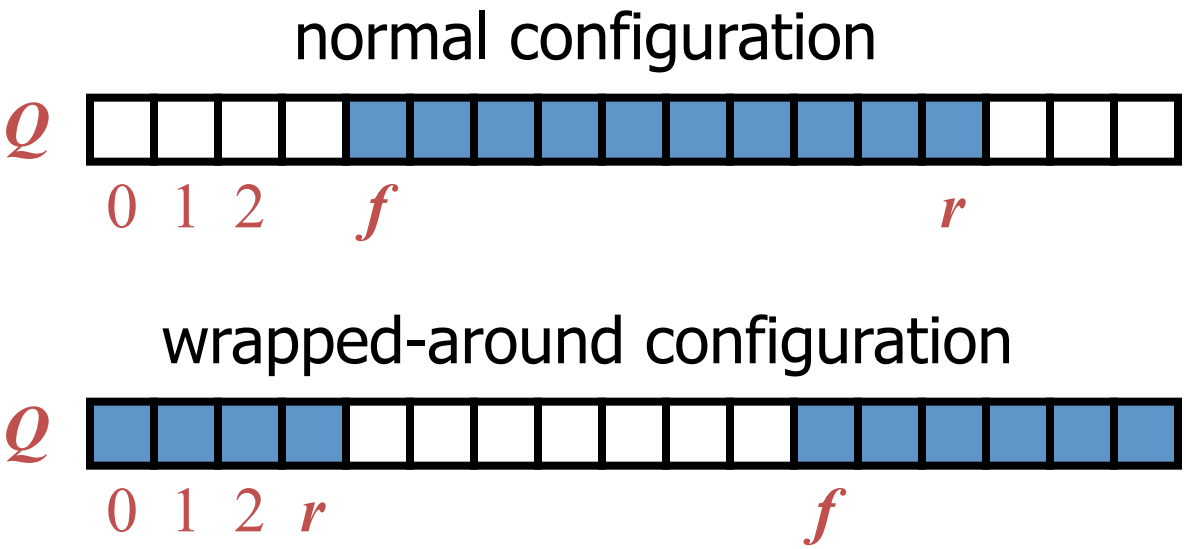
- Java has build-in queue Interface,
- Implemented class shown here:

  http://docs.oracle.com/javase/7/docs/api/java/util/Queue.html

Demo of our own implementation for Array Based Queue.

Array-based Queue in OUR demo

- 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 points to the rear element
- Array location r points to last inserted item, f points to the item that is ready to be dequeued next.
  - You could have a **different design** than this.



Summary Today

- Concept of Queue
  - First-in First-Out property
  - Elements enter a queue at the rear and are removed from the front.
    - Operations: enqueue(), dequeue(), front(), isEmpty(), size().
  - Two ways of representations.

ADT Deque

- Need for an ADT which offers
  - Add, remove, retrieve
  - At both front and back of a queue
- Double ended queue
  - Called a *deque*
  - Pronounced “deck”
- Actually behaves more like a double ended stack

ADT Deque

- Note deque interface,

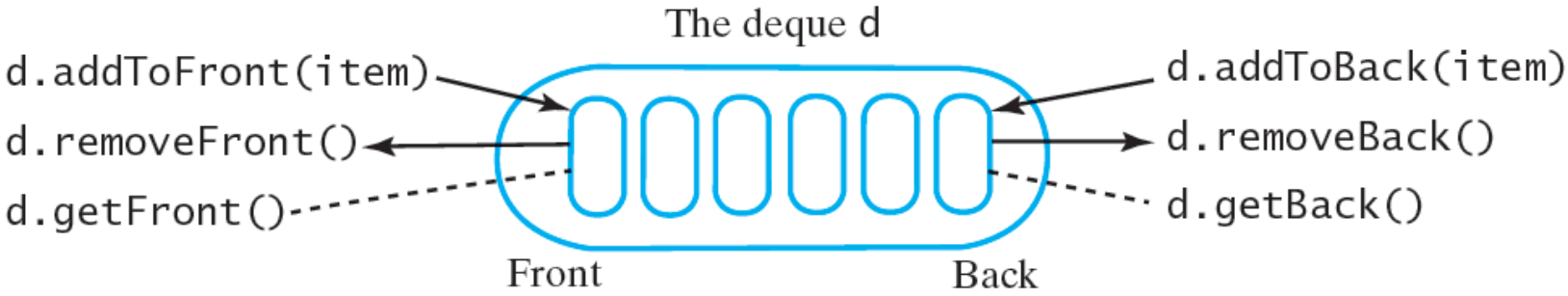
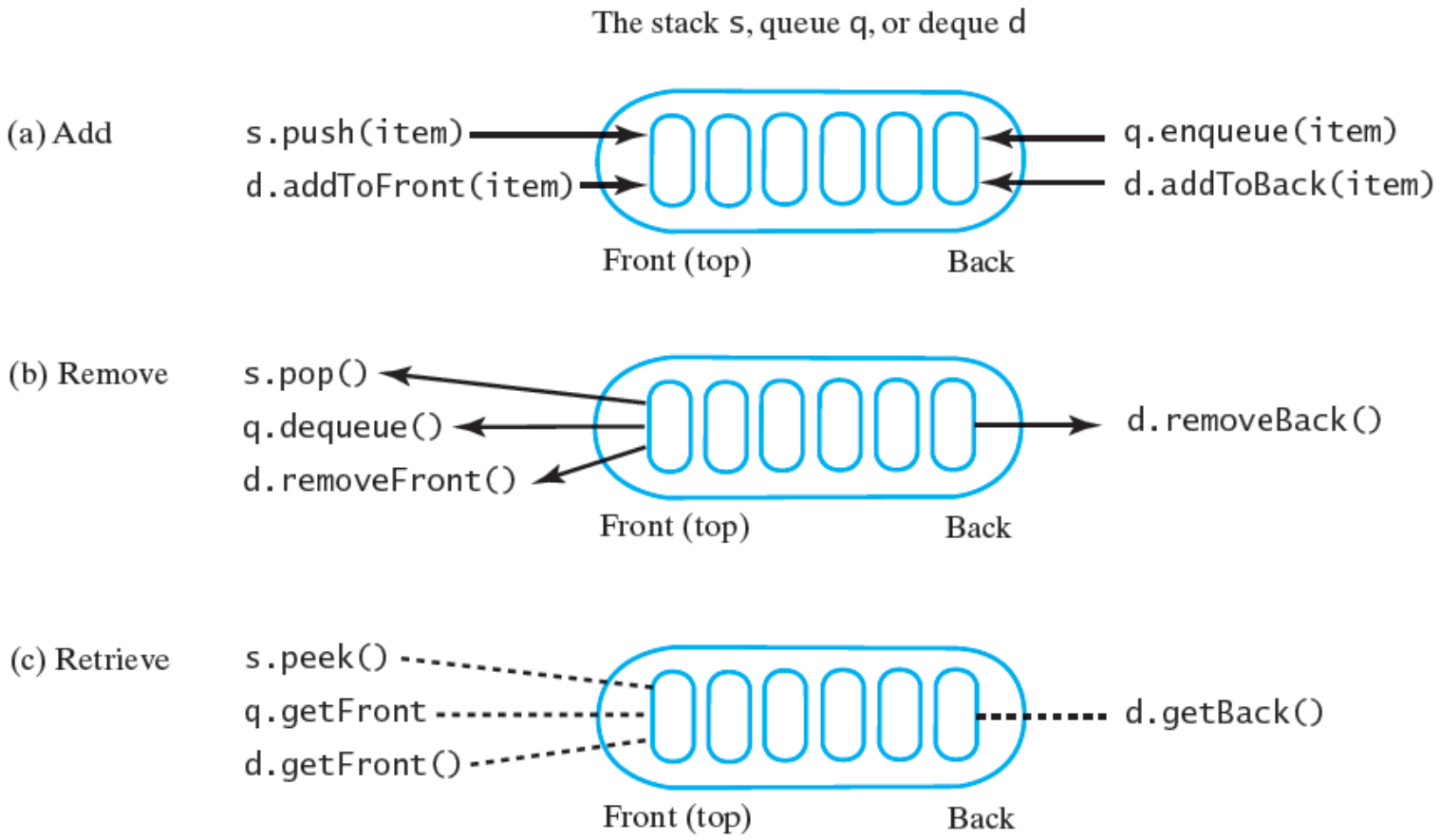


Figure 10-10 An instance *d* of a deque



A comparison of operations for a stack s, a queue q, and a deque d: (a) add; (b) remove; (c) retrieve

EASTERN WASHINGTON UNIVERSITY
start something big

Next Class

- Linked List Implementation