Deques

Outline and Required Reading:

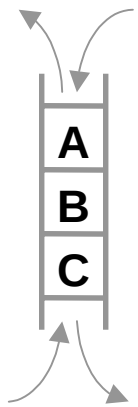
- **Deques (§ 4.4)**

COSC 2011, Fall 2003, Section A
Instructor: N. Vlajic

Deque ADT

Deque – **"double-ended queue" – queue-like linear data structure that supports insertion and deletion of items at both ends of the queue**

- **name "double-ended-stack" would be equally appropriate**
- **richer ADT than both the queue and the stack ADT**



An (efficient) existing Dequeu class can be used to implement Stack and/or Queue.

Deque ADT: Interface

Fundamental Methods
(Transformers)

```
public void insertFirst(Object element);
/* insert a new element at the front of the deque */
public void insertLast(Object element);
/* insert a new element at the rear of the deque */
public Object removeFirst();
/* remove the element at the front of the deque */
public Object removeLast();
/* remove the element at the rear of the deque */
```

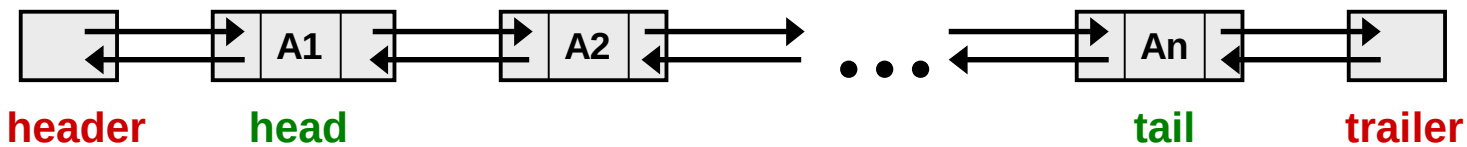
Supporting Methods
(Observers)

```
public int size();
/* return the # of objects in the deque */
public boolean isEmpty();
/* return true if the deque is empty */
public Object first();
/* get the front element without removing it */
public Object last();
/* get the rear element without removing it */
```

Deque ADT: Linked List Implementation

Singly Linked List Implementation – **Inefficient! Removal at the rear takes Θ(n) time.**

Doubly Linked List Implementation – **each node has "prev" and "next" link, hence all operations run at O(1) time**



Sentinel Nodes Header & Trailer – **"dummy" nodes that do not store any elements**

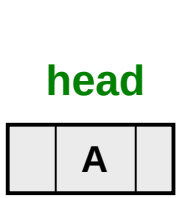
- **enable simpler handling of DLL in case of:**
  - **(1) removing the only remaining node in the list**
  - **(2) removing the head**
  - **(3) removing the tail**

Deque ADT: Linked List Implementation (cont.)

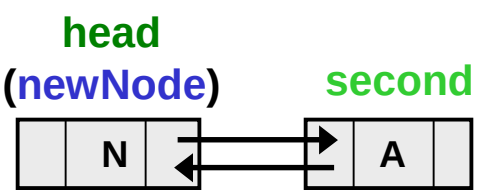
```
/* implementation without sentinels */
public void insertFirst(Object obj) {
        DLLNode newNode;
        if (head==null) {   newNode = new Node(obj, null, null);
                            head = newNode;
                            tail = newNode; }
        else {    DLLNode second = head;
                  newNode = new Node(obj, null, second);
                  head = newNode;
                  second.setPrev(newNode); }
        size ++;
}
```

Special care has to be taken in the case that this is the first node to be inserted.

DLLNode – 3 references in constructor!



(a) before insertion

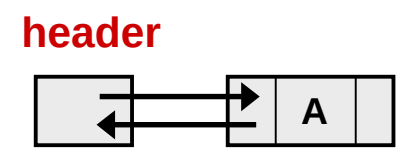


(b) after insertion

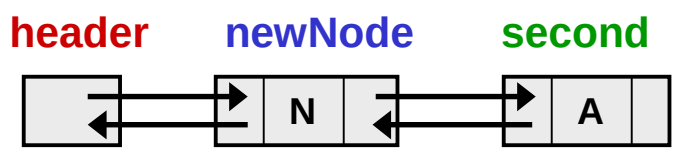
Deque ADT: Linked List Implementation (cont.)

```
/* implementation with sentinels */
public Object insertFirst(Object obj) {
        DLLNode second = header.getNext();
        DLLNode newNode = new DLLNode(obj, header, second);
        second.setPrev(newNode);
        header.setNext(newNode);
        size++; }
```

Works fine even if the list is empty !



(a) before insertion



(b) after insertion

Deque ADT: Performance of Linked List Implement.

Run Time – **Good! all methods run in constant O(1) time**

Method	Time
size	O(1)
isEmpty	O(1)
first	O(1)
last	O(1)
insertFirst	O(1)
insertLast	O(1)
removeFirst	O(1)
removeLast	O(1)

Space Usage – **Good! O(n), n – current # of elements in the stack**

General Note – **Implementation using DLL is most efficient, but**

- **nodes are slightly "larger" than in case of SLL**
- **more references to keep up to date (prev & next in each node)**

Deque ADT: Implementing Stacks/Queues with Deques

With an implementation of Deque ADT, we can easily implement the Stack/Queue interface.

Stack Method	Deque Implement.
size() isEmpty() top() push(obj) pop()	size() isEmpty() last() insertLast(obj) removeLast()

Queue Method	Deque Implement.
size() isEmpty() front() enqueue(obj) dequeue()	size() isEmpty() first() insertLast(obj) removeFirst()

Deque ADT: Impl. Stacks/Queues with Deques (cont.)

```
public class DequeStack implements Stack {

        private Deque D;

        public DequeStack() {
          D = new MyDeque(); }

        public int size() {
          return D.size(); }

        public boolean isEmpty() {
          return D.isEmpty(); }

        public void push(Object obj) {
          D.insertLast(obj); }

        public Object pop() throws StackEmptyException{
          try{  return D.removeLast();
          } catch (DequeEmptyException ece) {
                    throw new StackEmptyException("Stack is empty!") } }
        …
```