Stack, Queue ADT

Prof Bill - Jan 2020

Stack ADT

Stack is an ordered list data structure with LIFO access; all operations are O(1).
Pseudocode is for an array implementation.

```
Stack
     item array[]    // instance variables
     int top

     create( size)
       array = new array[size]
       top = 0

     push( item)
       if top > array.length   return Stack Overflow error
       array[top] = item
       top++

     item pop()
       if isEmpty() return null
       else
         top--
         return array[top]

     item peek()
       if isEmpty() return null
       else return array[top-1]

     boolean isEmpty()
       return (top == 0)

     int size()
       return top
```

Notes:
  ➢ Linked list implementation is straightforward: add and remove from front of list.
  ➢ JCF Stack has methods: push, pop, peek.
```
Stack<Integer> s = new Stack<>();
s.push( 42);
```

Queue ADT

Queue is an ordered list data structure with FIFO access; all operations are O(1).
Pseudocode is for an array implementation.

```
Queue
     item array[] // instance variables
     int front, rear, size

     create( size)
       array = new array[size]
       front = rear = size = 0

     enqueue( item)
       if size == array.length return Queue Overflow error
       array[rear] = item
       rear++ % array.length  // modulo for circular array
       size++

     item dequeue()
       if isEmpty() return null
       item = array[front]
       front++ % array.length  // modulo, circular array
       size--
       return item

     item peek()
       if isEmpty() return null
       return array[front]

     boolean isEmpty()
       return (size == 0)

     int size
       return size
```

Notes:
  ➢ Linked list implementation is straightforward... add to end, remove from front
  ➢ JCF Queue is an interface, use LinkedList as a concrete subclass

```
Queue<Integer> q = new LinkedList<>();
q.add( 7);
```