CS 313
- Kangmei Yang
Stack

- Elements are “stacked” on top of each other.
- Take off / put on to the top.
- It’s last-in-first-out (LIFO) data structure
- Can’t access bottom elements without removing the top ones.
- push: add an element to the top
- pop: removes the top element
Stack Uses

- to track function calls
- to check whether braces matches
- evaluate postfix expressions

main() (fun1() fun2())
Stack Implementation

- It may be implemented by either an array or linked list structure which add/remove to the same end.

  - **Array**
    
    | first | second | third |
    |-------|--------|-------|

    add/remove to the end

  - **LinkedList**
    
    | first | second | third |
    |-------|--------|-------|

    add/removes to the head

- The JCF does not have a Stack interface. The API does not recommend using the Stack class.
Queue

- A sequence of elements add to one end (tail) and remove from the other end (head).
- It’s first-in-first-out (FIFO) data structure.
- Can’t get the middle or the end without removing the front ones.

- enqueue: add an element to the end
- dequeue: removes the front element

| first | second | third |
Queue Uses

- For scheduling
- For breadth-first search
Queue Implementation

- It may be implemented by either an array or singly linked list structure which add/remove to the different end.
  - LinkedList
    
    ![LinkedList Diagram]
    
    add to the tail and remove from the head

  - Array
    
    ![Array Diagram]
    
    neither add nor remove from the front is efficient

- Circularly Array
  - Allow empty space in the front to reuse after the end
Circularly Array implements Queue

- Maintain front index, size, capacity
  - Add

<table>
<thead>
<tr>
<th>first</th>
<th>second</th>
<th>third</th>
<th>forth</th>
<th>fifth</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

- Remove

Calculating next available index: (front index + size) % capacity
Deque

- Short for double-ended queue
- add/remove from both ends
- Can’t get the middle ones without removing from one end.
- can be used as either a stack or a queue
Deque Implementation

- Similar to queue, it may be implemented by either a circularly array or doubly linked list structure which add/remove to both ends.

- JCF provides two these two implementation of Deque interface:
  - ArrayDeque
  - LinkedList

Note: LinkedList implements both Deque and List interface
Java Collection Framework

List
(add
remove
get
set
indexOf
contains)

Deque
(addFirst (offerFirst)
addLast (offerLast)
removeFirst (pollLast)
removeLast (pollLast)
getFirst (peekFirst)
getLast (peekLast)
queue: add, remove
stack: push, pop
contains)

ArrayList

LinkedList

ArrayDeque