List VS DLL User Methods

List			DLL		
Method name	Return type	Explanation	Method name	Return type	Explanation
findFirst()		void		C = head	
findNext()		void		C = C.next	
--	--	--	findPrevious()	void	C = C.previous
retrieve()		T		return C.data	
update(T e)		void		C.data = e	
full()		boolean		return false	
insert(T e)		void		Add node contain e after C and make it the C	
remove()		void		remove C element and make the next of it C	
empty()		boolean		Is the list empty? True if yes and false if not	
--	--	--	first()	boolean	C.previous == null
last()		boolean		C.next == null	

[ ] Require list not full

[ ] Require list not empty

- **For list:** You have interface List<T> , class Node<T>, class LinkedList<T> implements List<T>, ArrayList<T> implements List<T>.
- **For DLL:** You have class Node<T> , DoubleLinkedList<T>.
- All member or implementer methods you must write it in class DoubleLinkedList<T>
- Head, current, previous, next.
- Most to use p and q here.
- Check for empty, one, two and three situations.
- Static methods always <T> before return type.

Queue (FIFO) VS Priority Queue User Methods

Queue			Priority Queue		
Method name	**Return type**	**Explanation**	**Method name**	**Return type**	**Explanation**
enqueue(T e)	void	Add e to the queue in the end (tail)	enqueue(T e, Priority p)	void	Add e to the queue according to p
serve()	T	Remove the head element and its value returned also head = head.next	serve()	PQElement<T>	e and p in head of PQ removed and returned
length()		int		return size of the queue	
full()		boolean		return false	

[   ] Require Queue / PQ not full

[   ] Require Queue / PQ not empty

- **For Queue:** you have interface Queue<T> , class Node<T>, class LinkedQueue<T>, class ArrayQueue<T> .
- **For Priority Queue:** you have class PQNode<T>, class LinkedPQ<T>, class PQElement<T>.
- Head, tail, next.
- Tail is not used for PQ.
- Most to use p and q here.
- Maxsize not equal index, (starting with 1)
- ArrayQueue:
  Enqueue? Tail is in the empty index
  Serve? Head is in the index

Double Ended Queues (deque)

Method name	Return type	Explanation
addFirst(T e)	void	Add e to DQ as first element
addLast(T e)	void	Add e to DQ as last element
removeFirst()	T	remove and return the first element
removeLast()	T	remove and return the last element
getFirst()	T	return the first element
getLast()	T	return the last element
isEmpty()	boolean	DQ is empty? True Otherwise false
size()	int	return size

[ ] Require DQ not full

[ ] Require DQ not empty

Stacks (LIFO) User Methods

Method name	Return type	Explanation
push(T e)	void	Add e to the stack
pop()	T	Remove the last one added and return it also top = top.next
empty()	boolean	Stack is empty? True Otherwise false
full()	boolean	Return false

[ ] Require Stack not full

[ ] Require Stack not empty

- **For Stack:** you have class Node<T>, LinkedStack<T>, class ArrayStack<T>.
- top, next.

Stack Operations

Postfix: Only one Stack (numbers stack) **n = number , op = operation**

Infix: two Stacks (numbers stack, operations stack)

Postfix		**Infix**					
n?	op?	Is it (?	Is it n?	Is it op?			Is it) ?
push(n)	y = pop() x = pop() push(x op y)	op.push()	n.push()	Is it the top op? هل هناك عمليات أخرى في الستاك؟			op1= op.pop() op2= op.pop() …. Until op.pop() == (
				Yes?		No?	
				Is top op >= op? (precedence)		op.push(op)	
				Yes?	No?		
				y = op.pop() x = op.pop() n.push(x op y)	push(op)		