Implementation of Lists:

Entry class:
store one element of list, reference to adjacent ~~elem~~ entries

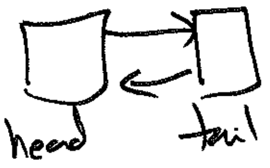
1. Linked Lists:

Singly linked, Doubly linked, Circular

Java: LinkedList = circular, doubly linked list.

Implementation of a doubly linked list, dummy head, tail ~~el~~ entries

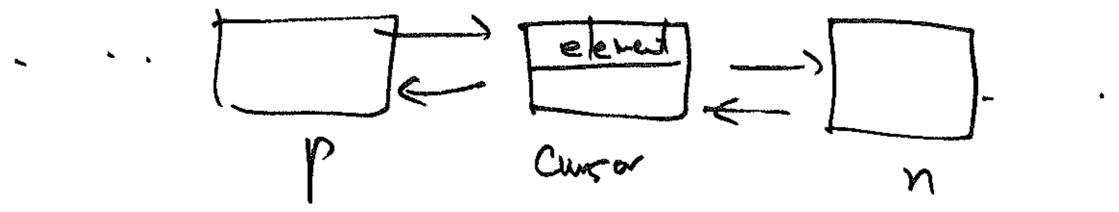
```
Entry<T>:  T element;                T = arbitrary type
           Entry<T> prev, next;          of elements | list
           Entry(~~E~~ x, p, n):  element ← x
                                  ~~E~~ prev ← p
                                  ~~E~~ next ← n
           Entry(x):  element ← x, prev ← null; next ← null

DoublyLinkedList (DLL):   DLL<T>
           Entry<T> head, tail;    size (int)
           DLL():   head ← new Entry(null)
                    tail ← new Entry(null, head, null)
                    head.next ← tail
                    size ← 0
```

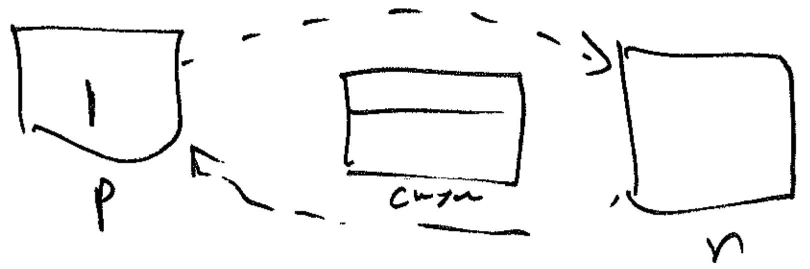


Helper functions:

```
find(x):   // find Entry with element = x  (first)
    cursor ← head.next;    // first Entry
    while cursor.element ≠ x and cursor ≠ tail do
        cursor ← cursor.next
    return cursor ≠ tail ? cursor : null
```

remove (cursor) : // removing element stored at cursor.





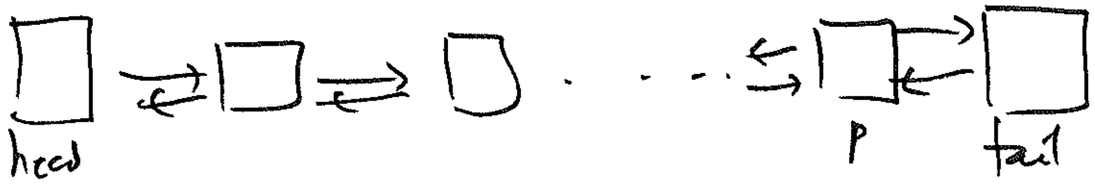
```
p ← cursor. prev          n ← cursor.next
p.next ← n                n.prev ← p
size --
return cursor.element
```

find (index) : // find entry at index
// first element has index = 0

```
if index < 0 or index ≥ size then return null
cursor ← ~~header~~ head. next        i ← 0
while i < index do
    cursor ← cursor.next        i++
return cursor.
```

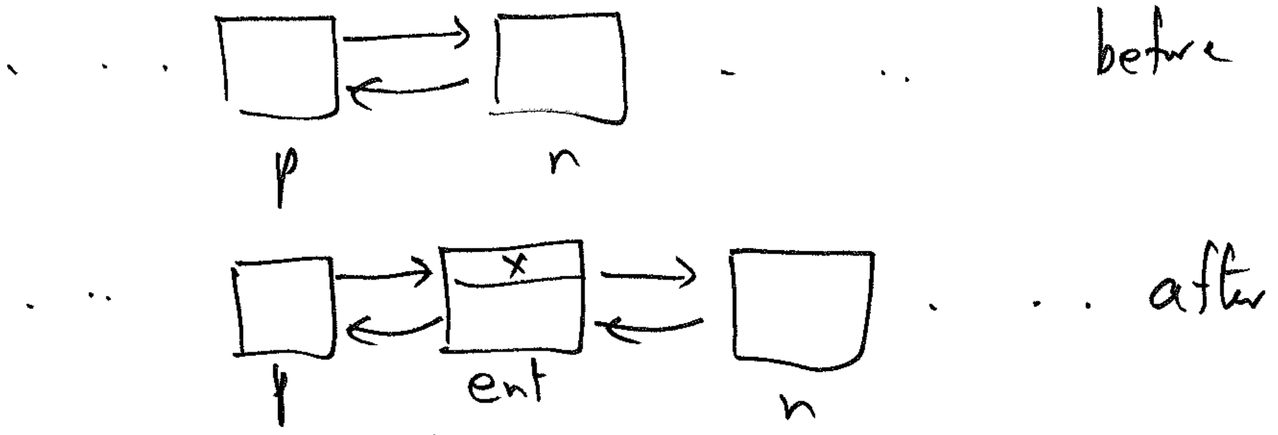
---

add(x) : // add ~~a~~ new element x to list



```
p ← tail. prev
add( p, x)
```

add(p, x) : // add x after entry p .



```
n ← p.next
ent ← new Entry(x, p, n)
p.next ← ent        n.prev ← ent
size ++
```

<u>Iterator</u> : hasNext ( ), next ( ), remove ( )

a class DLLIterator <T> : - fields to store state of iteration

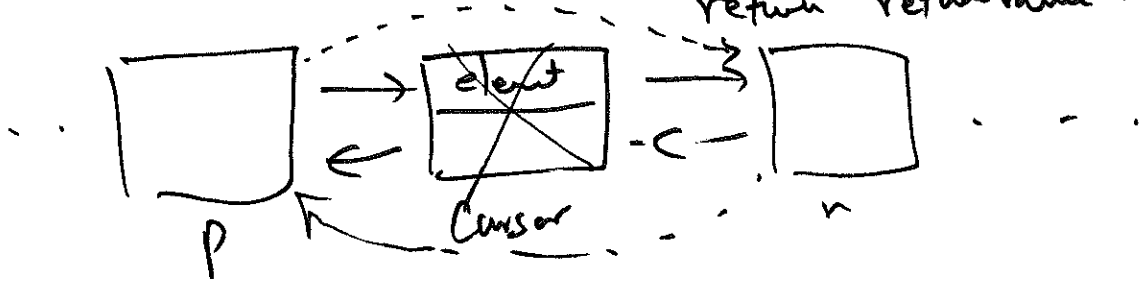
Entry<T> cursor, boolean ready.

(Java: Iterator, ListIterator)
↳ additional ops (previous ( ), hasPrevious ( ), add (x))

Convention: object returned recently by next() is at cursor.

```
DLLIterator ( ) :   cursor ← head ; ready ← false
  hasNext ( ) :   return cursor.next ≠ tail
    next ( ) :   ready ← true
                 cursor ← cursor.next ; return cursor.element
  remove ( ) :   if not ready ~~then~~ throw exception ....
                 ~~else~~  remove (cursor) ; ready ← false
                 ~~return cursor.element~~
```

```
remove( ):  if not ready   exception
            else  returnValue ← cursor.element
                  remove(cursor)
                  ready ← false
                  cursor ← cursor.prev
                  return returnValue
```



```
T remove(x):   //remove first occurrence of x from list
        ent ← find(x)
        if ent = null then  return null
        else return remove(ent)

bool contains(x):  return find(x) ≠ null
void addFirst(x):    add(head, x)
void addLast(x):     add(tail.prev, x)
 T removeFirst():    size = 0 ? ___ exception
                     return remove(head.next)
 T removeLast():     size = 0 ? __ exception
                     return remove(tail.prev).
  indexing ops:   index :  index < 0 or index ≥ size  — exception
    get(index): return find(index).element   |  add(index, x):
    set(index, x):  find(index).element ← x  |      index = 0 ? addFirst(
                                             |  else add(find(index-1), x
                                             remove(index): ret remove(find(in
```

Collection: a group of objects of the same type; null and duplicate elements allowed.
List: ordered collection (sequence) of items drawn from some type T: Element → Element → etc
Null and duplicate elements are allowed.

Main operations on lists (LL = LinkedList implementation, AL = ArrayList implementation):

Operation	Meaning	Java	LL	AL
insert(x)	insert x at the end	add(x)	O(1)	O(1), amort
delete(x)	remove first occurrence of x	remove(x)	O(n)	O(n)
find(x)	is x there?	contains(x)	O(n)	O(n)
iterator( )	create iterator for list	iterator( )	O(1) per item	O(1) per item
size( )	number of items in list	size( )	O(1)	O(1)
isEmpty( )	is list empty?	isEmpty( )	O(1)	O(1)
clear( )	discard all elements, making list empty	clear( )	O(1)+garbColl	O(1)+garbColl
toArray( )	return array of items in list	toArray( )	O(n)	O(n)

Indexing operations (first element is at index 0):

Operation	Meaning	Java	LL	AL
insert(i, x)	insert x at the index i	add(i, x)	O(n)	O(n)
delete(i)	remove item at index i	remove(i)	O(n)	O(n)
get(i)	get element at index i	get(i)	O(n)	O(1)
set(i, x)	set element at index i to x	set(i, x)	O(n)	O(1)

Iterator operations:

Operation	Meaning	Java	LL	AL
hasNext( )	is there another item?	hasNext( )	O(1)	O(1)
next( )	return next item	next( )	O(1)	O(1)
delete( )	delete item at cursor (current item)	remove( )	O(1)	O(n)

Additional operations in ListIterator class:

Operation	Meaning	Java	LL	AL
insert(x)	insert x before item returned by next( )	add(x)	O(1)	O(n)
hasPrevious()	is there a previous item?	hasPrevious()	O(1)	O(1)
previous( )	return previous item	previous( )	O(1)	O(1)