List Invariants

Thursday, September 17, 2015 11:39 AM

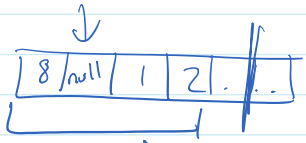
Lab 4 → due 9/23 11:55 PM

next SI 412 (TCL) 6:00

Oct 6th?

[1, 2, 3, ..., n]

[8, 1, 2, ...] (null)

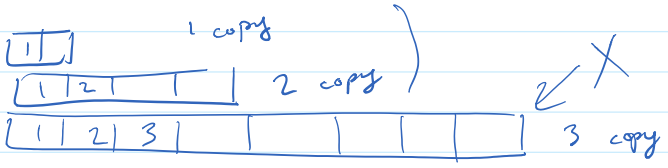


Within the array, list elements are "neighbors"

list.add(null) [8, 1, 2, null]

Wrong?

add(1)
add(2)
add(3)



Only expand the list when necessary
The size of the array is at most 2x the size of the list

java.util.ArrayList
java.util.LinkedList
} implement java.util.List

get
add
remove

Quiz 02

Write a recursive function that takes an array and prints every element (in the order that the array contains it)

Pseudocode

If
While
⋮
print(...)
len(...)

Quiz 2 Solution

```
# printArray(___, 0)
function printArray(arr, index=0):
    if (index >= arr.length):
        return
    EndIf
```

} base

```
        return                    } base
    EndIf
    print(arr[index]);
    printArray(arr, index+1)
End Function

function printArray(arr):
    printArray(arr, 0)
EndFunction
```

ap = new ArrayPrinter
ap.printArray

```
public static void printArray(Object[] arr, int index) {
    if (index >= arr.length) {
        return;
    }
    System.out.println(arr[index]);
    printArray(arr, ++index);
}
public static ...
    printArray(arr, 0);
}
```

arr = {1, 2, 3, 4};

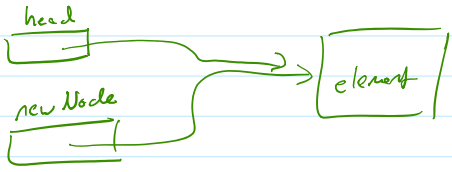
1
2
⋮
4

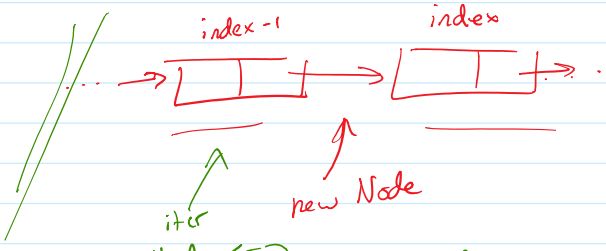
Linked Lists (Singly)


```
public class LinkedList<E> implements List<E> {
    private class Node<E> {
        public E element;
        public Node<E> next;
    }
    private int size;
    private Node<E> head;

    public LinkedList() {
    }

    public void insert(int index, E element) {
```

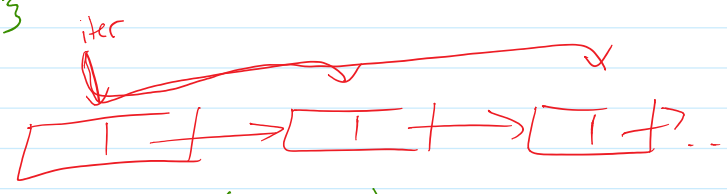
insert
append
delete
get
size

```
if (index < 0 || index > size) {
        throw new ListIndex... Exception();
}
if (head == null) {
    //Node<E> newNode = new Node<E>(); //all members are null
    newNode.element = element;
    size++;
    head = newNode; // references
    return;
}
```



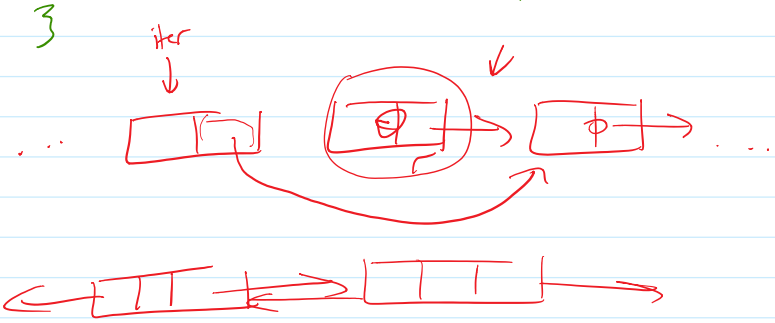



```
Node<E> iter = head;
for (int i=0; i < index-1; i++) {
    iter = iter.next;
}
Node<E> newNode = new Node<E>();
newNode.element = element;
newNode.next = iter.next;
iter.next = newNode;
size++;
}

public void append(E element) {
    insert(size, element);
}

public E get(int index) {
    // Exception (>= size)
    Node<E> iter = head;
    for (int i=0; i < index; i++) {
        iter = iter.next;
    }
    // iter has index 'index'
    return iter.element;
}
```




```
public E remove(int index) {
    Node<E> iter = head;
    for (int i=0; i < index-1; i++) {
        iter = iter.next;
```

```
}
E toReturn = iter.next.element;
iter.next = iter.next.next;
}
```




```
Doubly Linked Node <E>
    E element
    DL Node <E> next;
    DL Node <E> prev;
```