Section 3

Worksheet can be found on Kevin-Miao.com

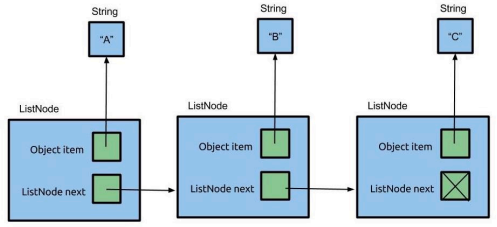
Today

+ Linked Lists
+ Quiz Review
+ Worksheet
+ Questions/Attendance

Linked Lists

Vanilla

['A', 'B', 'C']



- Enhancements
  1. Encapsulation : Abstraction
  2. Sentinel : No null Handling
  3. Doubly-linked : efficiency
  4. Generic Lists
- Destructive vs Non-Destructive
  - Destructive → Mutative (void)
  - Non-Destructive → Returns New List

Quiz Review

Q 1

```
public class Panda {
        static String food;
        int age; //any int
        boolean happy;

        public Panda(int age) {
                this.age = age;
        }
}

public class PandaList {
        public Panda item;
        public PandaList rest;

        public PandaList(Panda p) {
                this.item = p;
                this.rest = null;
        }
}
```

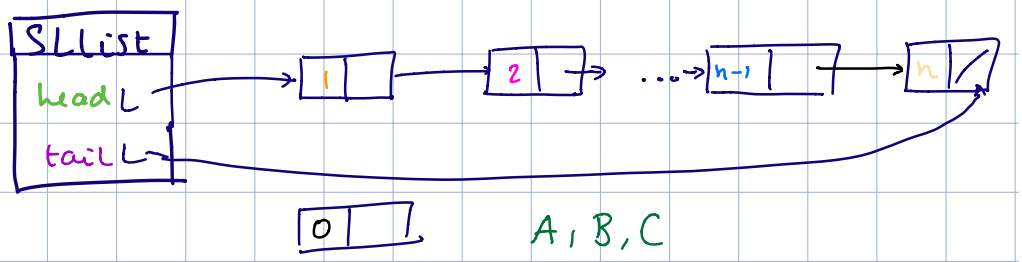
```
public static void main(String[] args) {
        Panda p = new Panda(1);
        PandaList lst = new PandaList(p);
        lst.rest = new PandaList(p);
        lst.rest.rest = lst;
        p = new Panda(-10);
        PandaList temp = lst;
        lst = new PandaList(p);
        lst.rest = temp;
        lst.rest.rest.rest.item.food = "Bamboo";
        temp.rest.rest.item.happy = true;
        temp.item.food = "shoots";
        lst.rest.rest.rest = lst;
        lst.rest.rest.rest.item.happy = false;
        lst.rest.rest.item.happy = true;
        System.out.println(temp.rest.item.age);            // Print statement 1
        System.out.println(temp.rest.rest.rest.item.food); // Print statement 2
        System.out.println(lst.rest.item.happy);           // Print statement 3
        System.out.println(temp.rest.rest.item.happy);     // Print statement 4
        System.out.println(lst.rest.rest.item.age);        // Print statement 5
}
```

visualizer link on website

Q 2 Which functions are supported in constant (O(1)) time?

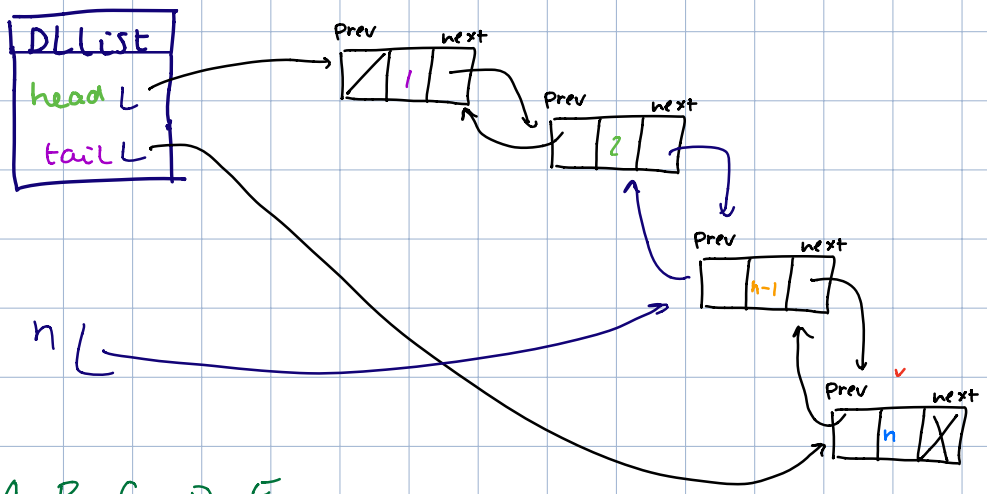
A) addFirst (int x)
B) addLast (int x)
C) removeFirst (int x)
D) removeLast (int x)
E) remove (Node n)

1) Encapsulated Singly linked list



A, B, C

2) Encapsulated Doubly Linked List



A, B, C, D, E

remove (Node n) is constant!

! remove (int n) for n := index is O(n)!!

b/c you have to find the node.

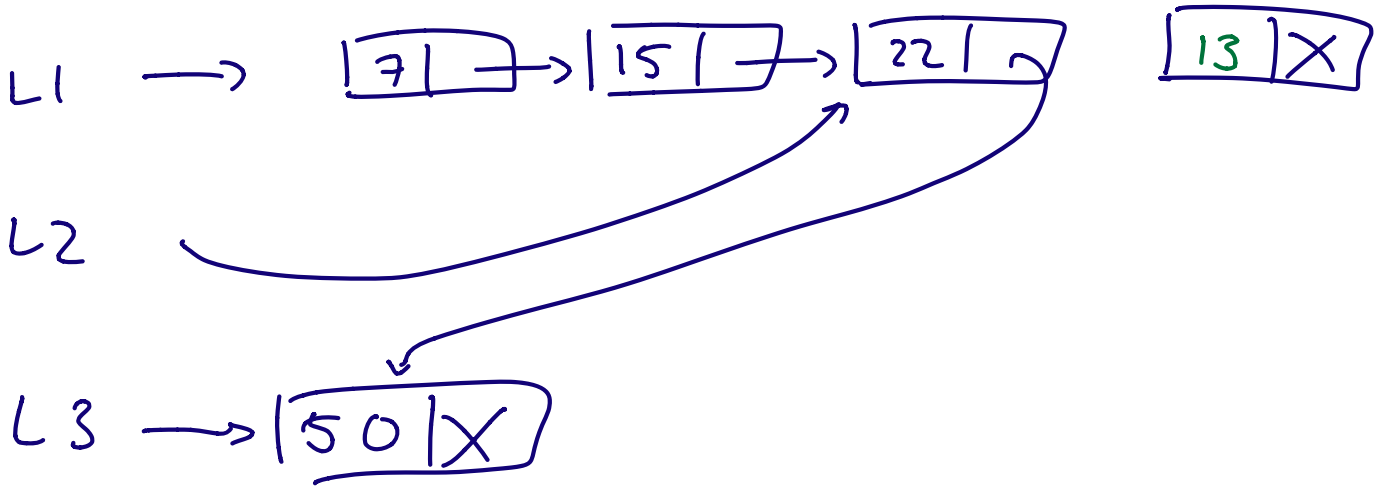
CS 61B Summer 2020 | Small Group Tutoring Section 3: Linked Lists | Worksheet 3

1 Pointer Practice (Assume Vanilla Implementation)

Draw the resulting box and pointer diagram for the IntLists after the following code is executed:

IntLists

```
IntList L1 = IntList.of(7,15,22,31);
IntList L2 = L1.next.next;
L2.next.item = 13;
L1.next.next.next = L2;
IntList L3 = new IntList(50);
L2.next.next = L3;
```



2 Skip Me

Write a function that takes in an IntList L, which must contain at least one element, and returns an IntList with every odd indexed element removed, starting at index 0. For example, if $L = \{1,2,3,4\}$, the function should return an IntList with elements $\{1,3\}$.

① Observation: DESTRUCTIVE

② Two Base Cases: {1,2,3,4,null} or {1,2,3,null}

1. **Destructive**: input IntList, L, should be modified

```
public static void skipDestructive (IntList L) {
    if (L.next == null || L == null) {
        return;
    }
    L.next = L.next.next;
    skipDestructive(L.next);
}
```

2. **Nondestructive**: input IntList, L, should not be modified

observation:
① we need 2 pointers one for our new list and old list.
② we need head pointer b/c otherwise lost reference to new list.

```
public static IntList skipNondestructive (IntList L) {
    IntList pointer = L;
    IntList retPtr = new IntList(pointer.item);
    IntList retHead = retPtr;
    while (pointer.next != null && Pointer.next.next != null) {
        retPtr.next = new IntList(pointer.next.next.item);
        pointer = pointer.next.next;
        retPtr = retPtr.next;
    }
    return retHead;
}
```

3 Benefits of Enhancements

1. List one advantage of having a sentinel node. no null Handling ERRORS

2. Suppose we implement a doubly linked list with a sentinel. In order to write the addFirst method, which pointers will we change?

* sentinel.next
  sentinel.prev
* sentinel.next.prev
  sentinel.next.next