Lecture 11 - February 11

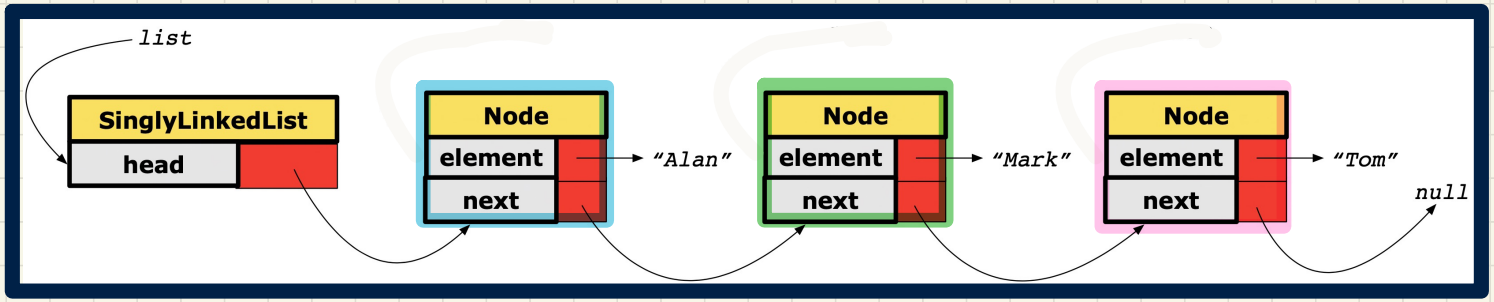
Arrays and Linked Lists

SLL: removeFirst, addLast
SLL: getNodeAt, insertAt, removeLast
Exercises: insertAfter, insertBefore

Announcements/Reminders

- ProgTest1 guide & example questions to be released
- *splitArrayHarder*: solution and tutorial video released
- Assignment 2 (on SLL) released
  + Required studies: Generics in Java (Slides 33 – 36)
  + Recommended studies: extra SLL problems
- Assignment 1 solution released
- Lecture notes template, Office Hours, TA Contact

SLL: Setting a List's Head to a Chain of Nodes



Approach 3

anonymous objects

Q. Given: SinglyLinkedList list = **new** SinglyLinkedList();
Write a **single line** of Java code to construct the above chain.

```
list.setHead(new Node("Alan", new Node("Mark", new Node("Tom", null))));
```

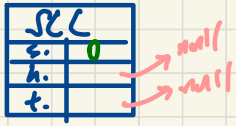
SLL Operation (sketch): Removing the First Node

void removeFirst()

Boundary Cases?

↳ size == 0 → Exception

↳ size == 1 → result: empty list



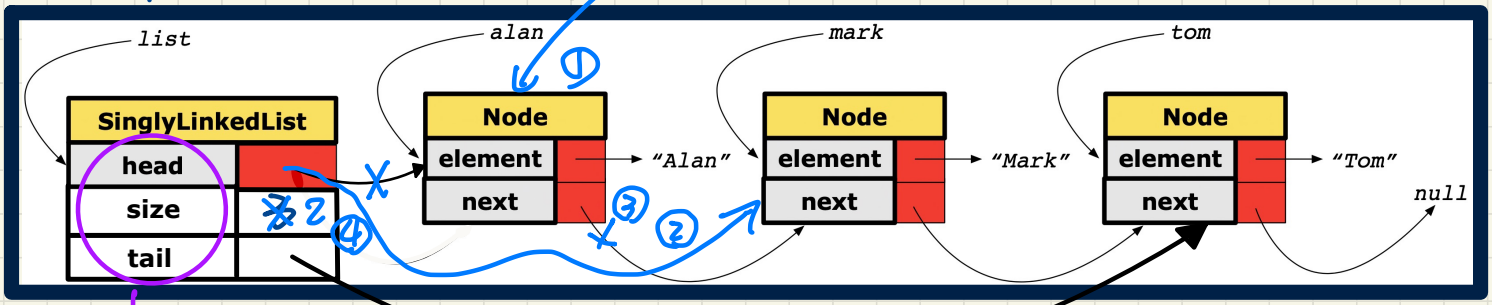
General Cases?

↳ size >= 2

① , ② , ③ , ④

∴ algo. does not depend on the size of the chain

O(1)

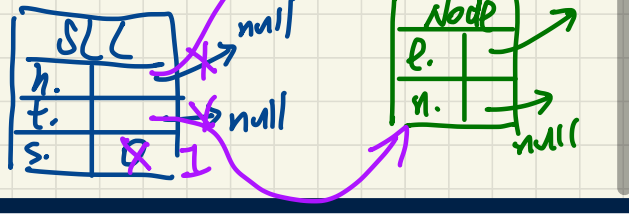


SLL Operation (sketch): Adding a Last Node

void addLast(String e)

Boundary Cases?

↳ size == 0

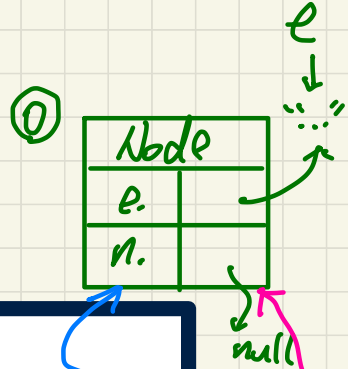


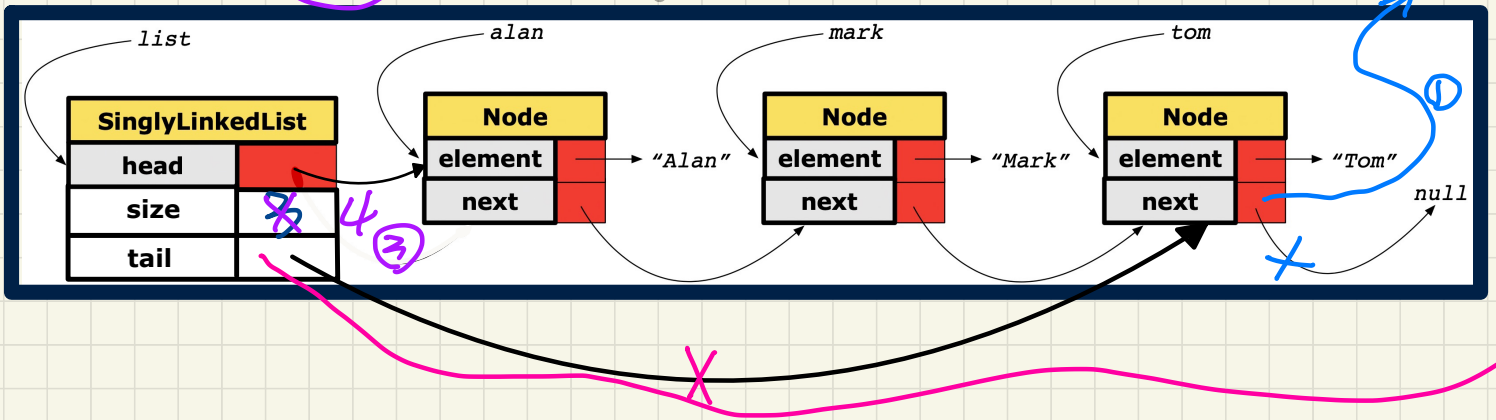
General Cases?

↳ size >= 1

O(1)

e ↓ ... @, ?, ≥, ≥.

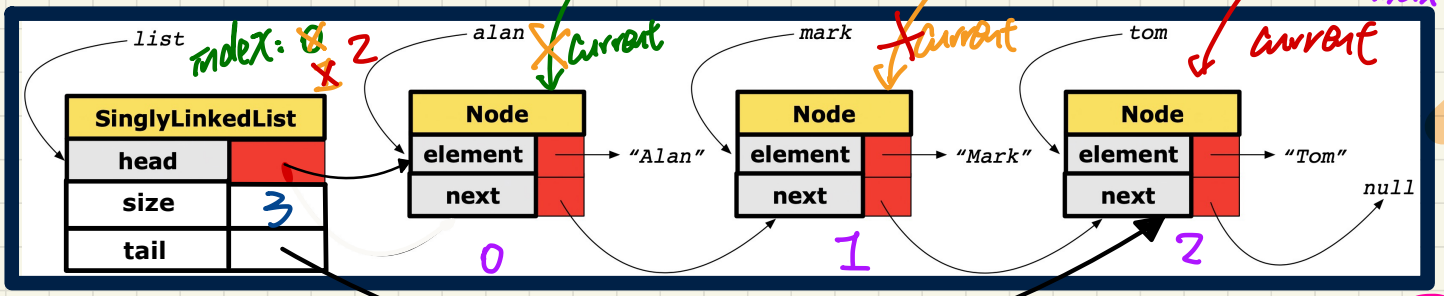




* when i = n-1 ⇒ list.tail . O(1) ; when i = n-2 ⇒ O(n-2) = O(n)

O((n-1) · 1) → O(n)

SLL Operation: Accessing the Middle of the List



0 ≤ i ≤ list.size-1

```
1  Node getNodeAt (int i) {
2    if (i < 0 || i >= size) { /* error
3    else {
4      int index = 0;
5      Node current = head;
6      while (index < i) { /* exit when
7        index ++;
8        current = current.getNext();
9      }
10     return current;
11   }
12 }
```

exit when index == i

Trace: list.getNodeAt(2)

current	index	index < 2	Start of Iteration
alan	0	0 < 2	1st it.: index 0 → 1, current alan →
mark	1	1 < 2	2nd it.: index 1 → 2, current mark → tom
tom	2	2 < 2	

exit.

① # it. = i

Q. Does **tail** or **size** need to be updated? No!

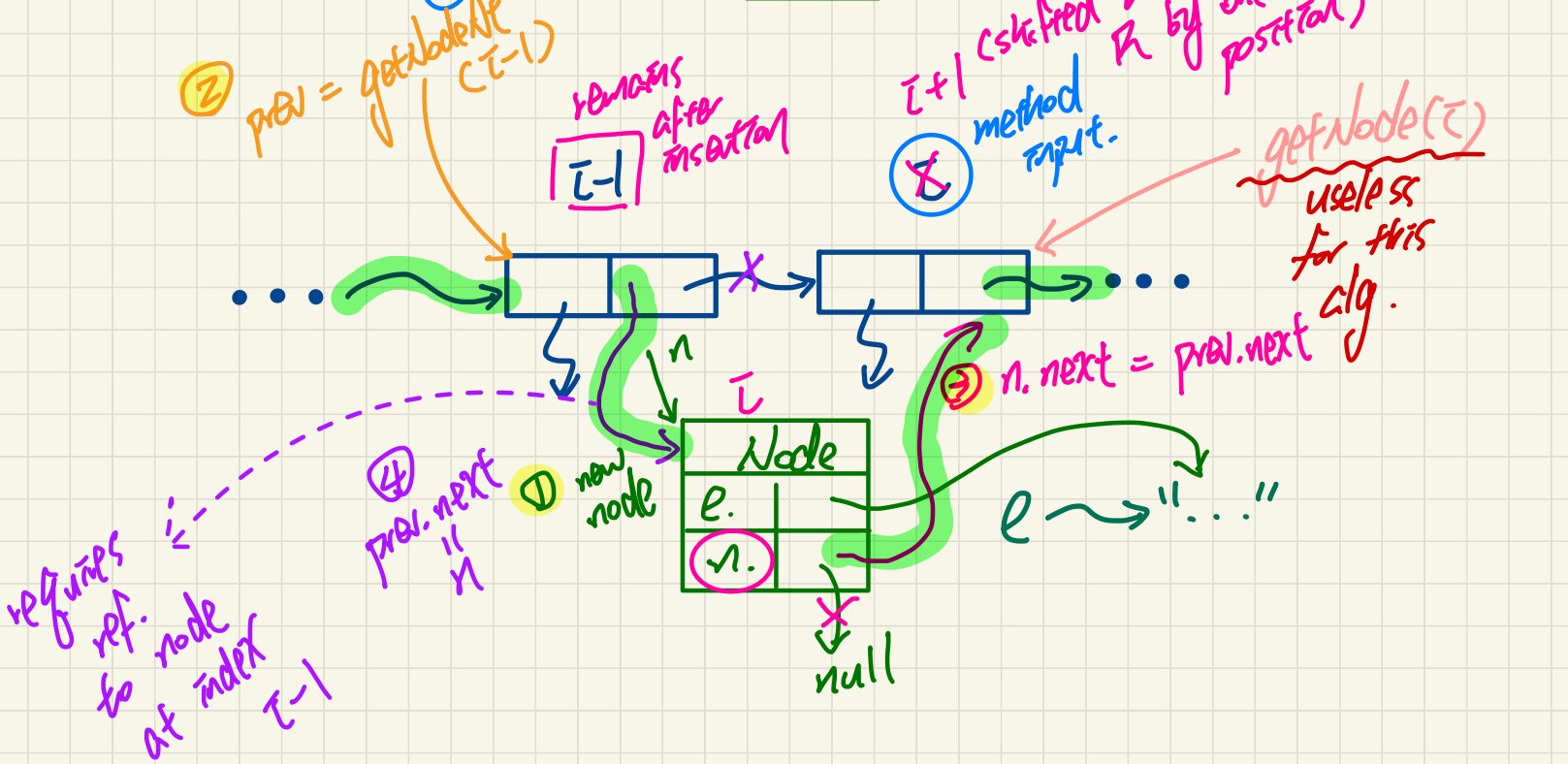
* ref to node at index i-1 not given

Idea of Inserting a Node at index i

* only an index is given

case i = 0 considered separately (i.e., addFirst(e))

Case: addAt(i, e), where 0 < i ≤ size



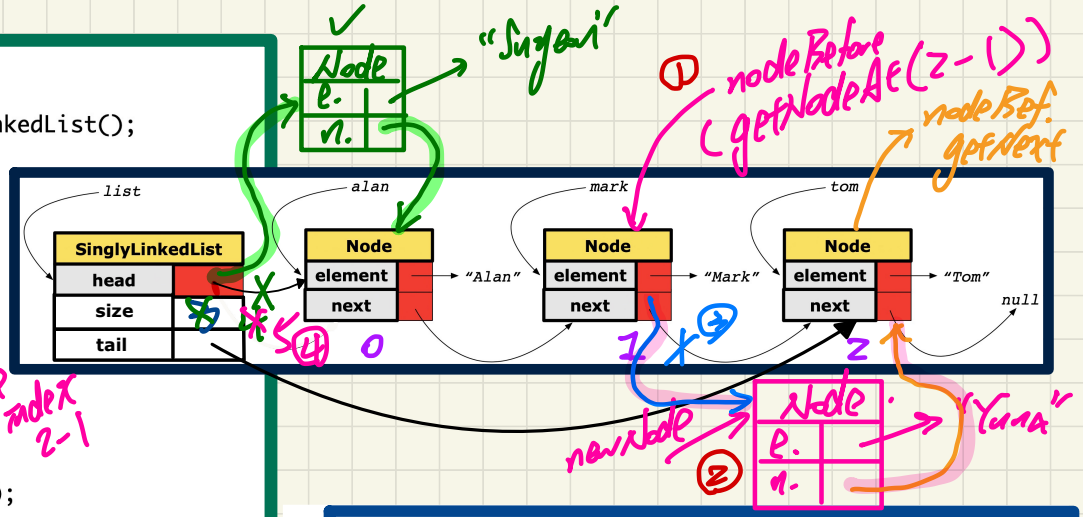
SLL Operation: Inserting to the Middle of the List

```java
@Test
public void testSLL_addAt() {
    SinglyLinkedList list = new SinglyLinkedList();
    assertTrue(list.getSize() == 0);
    assertTrue(list.getFirst() == null);

    list.addFirst("Tom");
    list.addFirst("Mark");
    list.addFirst("Alan");
    assertTrue(list.getSize() == 3);

    list.addAt(0, "Suyeon");
    list.addAt(2, "Yuna");
    assertTrue(list.getSize() == 5);
    list.addAt(list.getSize(), "Heeyeon");
    assertTrue(list.getSize() == 6);
    assertEquals("Suyeon", list.getNodeAt(0).getElement());
    assertEquals("Alan", list.getNodeAt(1).getElement());
    assertEquals("Yuna", list.getNodeAt(2).getElement());
    assertEquals("Mark", list.getNodeAt(3).getElement());
    assertEquals("Tom", list.getNodeAt(4).getElement());
    assertEquals("Heeyeon", list.getNodeAt(5).getElement());
}
```

need ref. to node at index 2-1




```java
1  void addAt (int i, String e) {
2    if (i < 0 || i > size) {
3      throw new IllegalArgumentException("Invalid Index.");
4    }
5    else {
6      if (i == 0) {
7        addFirst(e);
8      }
9      else {
10       Node nodeBefore = getNodeAt(i - 1);
11       Node newNode = new Node(e, nodeBefore.getNext());
12       nodeBefore.setNext(newNode);
13       size ++;
14     }
15   }
16 }
```

Q. Does **tail** or **size** need to be updated?

SLL Operation: Removing the End of the List

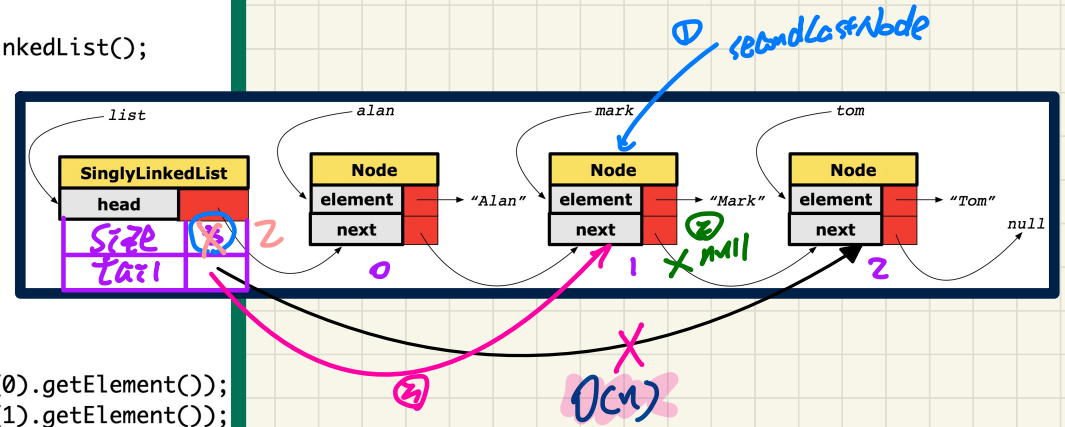
```java
@Test
public void testSLL_removeLast() {
    SinglyLinkedList list = new SinglyLinkedList();
    assertTrue(list.getSize() == 0);
    assertTrue(list.getFirst() == null);

    list.addFirst("Tom");
    list.addFirst("Mark");
    list.addFirst("Alan");
    assertTrue(list.getSize() == 3);

    list.removeLast();
    assertTrue(list.getSize() == 2);
    assertEquals("Alan", list.getNodeAt(0).getElement());
    assertEquals("Mark", list.getNodeAt(1).getElement());

    list.removeLast();
    assertTrue(list.getSize() == 1);
    assertEquals("Alan", list.getNodeAt(0).getElement());

    list.removeLast();
    assertTrue(list.getSize() == 0);
    assertNull(list.getFirst());
}
```




```java
1  void removeLast () {
2    if (size == 0) {
3    throw new IllegalArgumentException("Empty List.");
4    }
5    else if (size == 1) {
6    removeFirst();
7    }
8    else {
9    Node secondLastNode = getNodeAt(size - 2);
10   secondLastNode.setNext(null);
11   tail = secondLastNode;
12   size --;
13   }
14 }
```

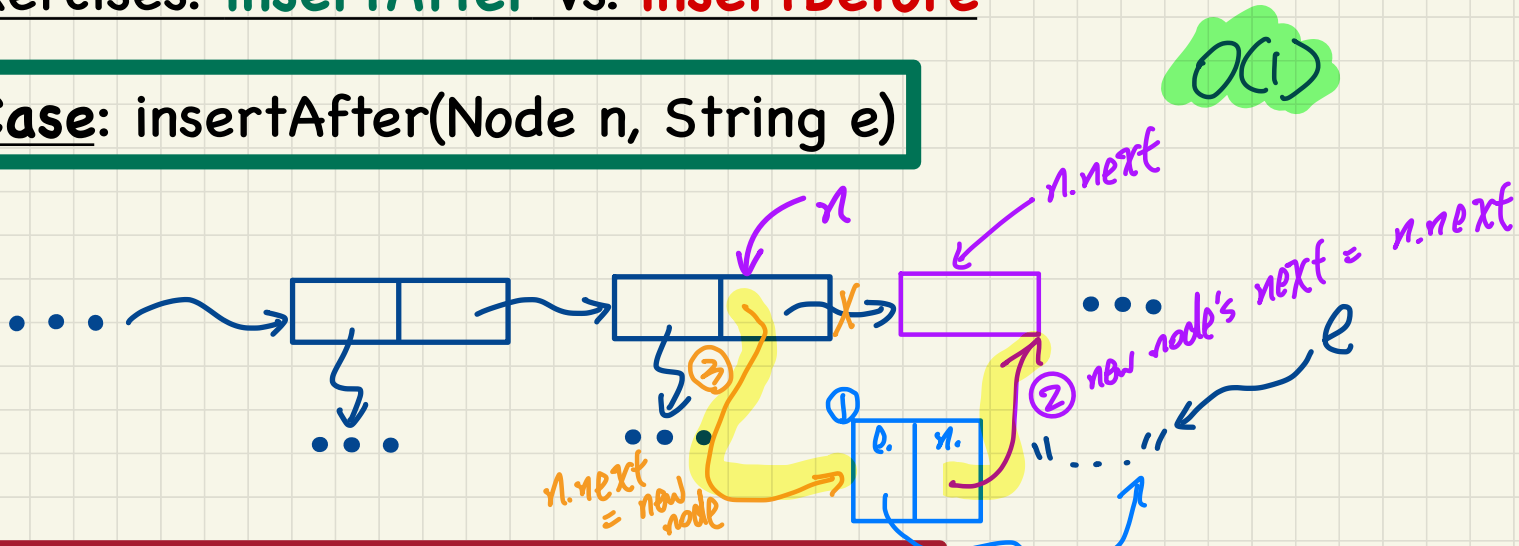
O(n-2) = O(n)

n-2=1

Q. Does tail or size need to be updated?

Exercises: insertAfter vs. insertBefore

Case: insertAfter(Node n, String e)



Case: insertBefore(Node n, String e)