Lecture 11 - February 11

Arrays and Linked Lists

SLL: removeFirst, addLast
SLL: getNodeAt, insertAt

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 Operation: Inserting to the Front of the List

O(1)

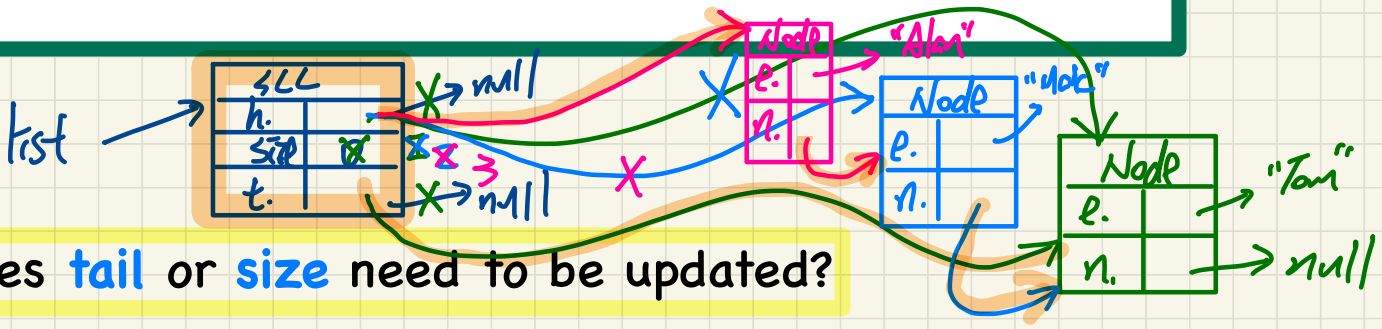
```java
@Test
public void testSLL_02() {
    SinglyLinkedList list = new SinglyLinkedList();
    assertTrue(list.getSize() == 0);
    assertTrue(list.getFirst() == null);

    list.addFirst("Tom");
    list.addFirst("Mark");
    list.addFirst("Alan");
    assertTrue(list.getSize() == 3);
    assertEquals("Alan", list.getFirst().getElement());
    assertEquals("Mark", list.getFirst().getNext().getElement());
    assertEquals("Tom", list.getFirst().getNext().getNext().getElement());
}
```

first.size==0

what if interleaving addFirst and addLast

```
1  void addFirst (String e) {
2    head = new Node(e, head);
3    if (size == 0) {
4      tail = head;
5    }
6    size ++;
7  }
```



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

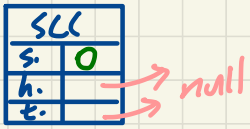
SLL Operation (sketch): Removing the First Node

void removeFirst()

Boundary Cases?

↳ size == 0 → Exception

↳ size == 1 → result: empty list



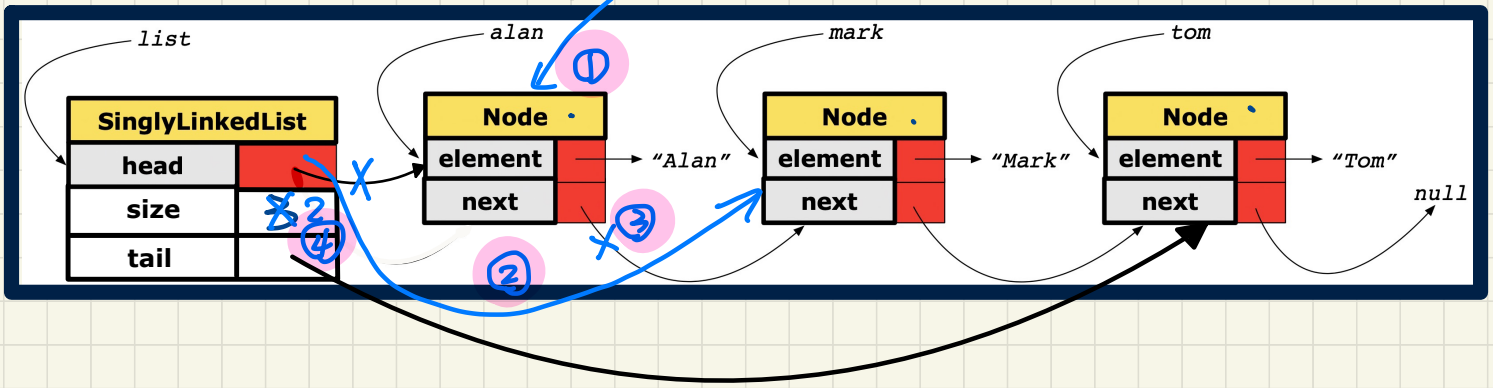
General Cases?

↳ size ≥ 2

O(1)

∵ none of the operations depends on the # nodes in the chain

① , ② , ③ , ④

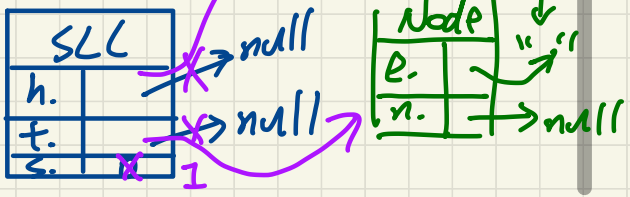


SLL Operation (sketch): Adding a Last Node

```
void addLast(String e)
```

Boundary Cases?

↳ size == 0



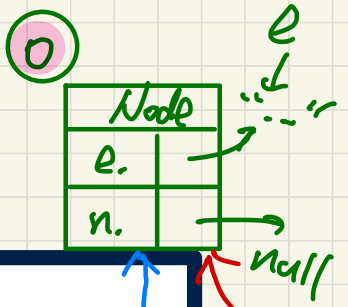
General Cases?

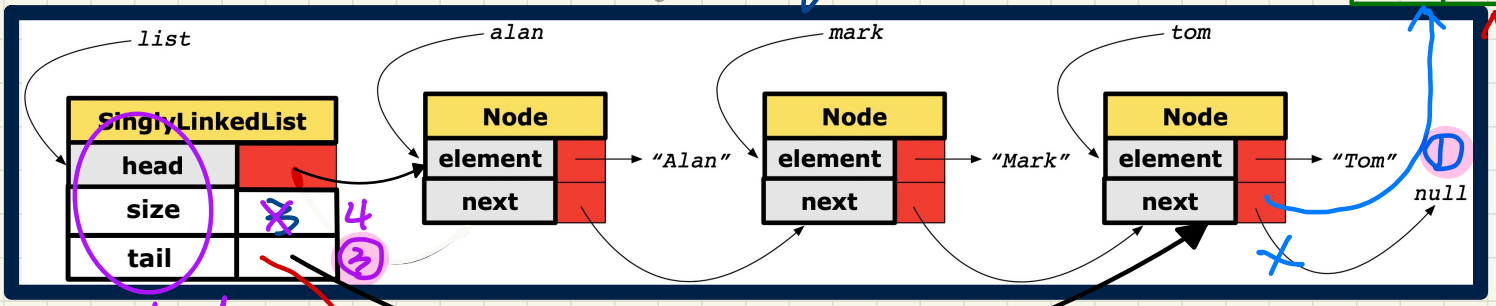
↳ size >= 1

O(1)

0, 1, 2, 3

all are variable assignments

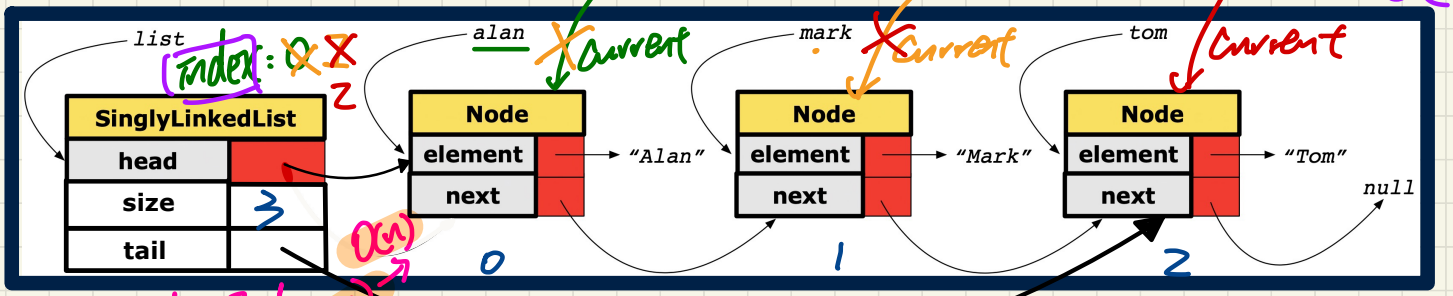




* current always references the node stored at index "index"

SLL Operation: Accessing the Middle of the List

O(i) = O(n-1) = O(n)



iterations = i

* to access the 2nd last node: getNodeAt (n-2)

$0 \le i \le lst.size - 1$

```
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

i == 2.

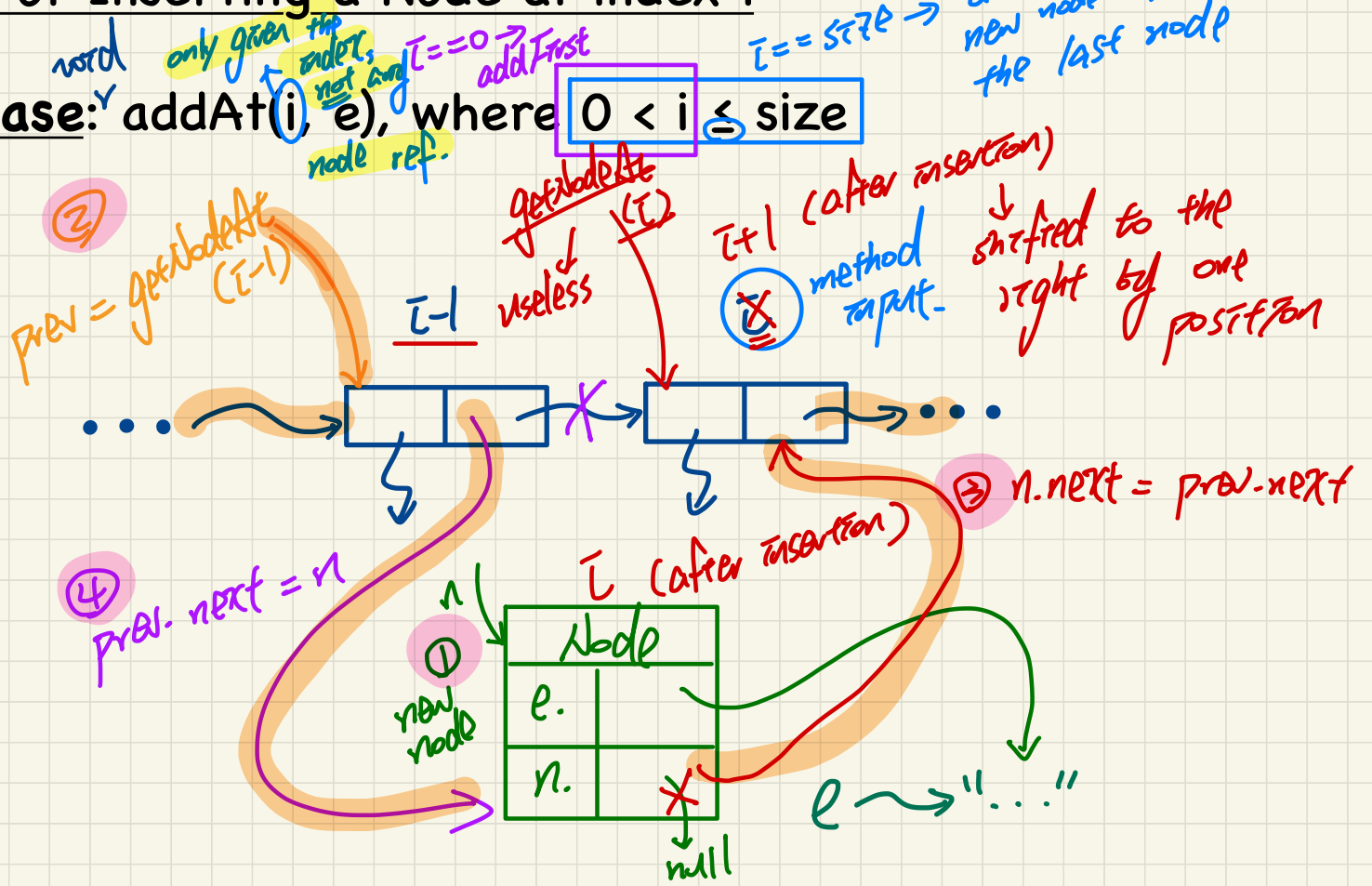
Trace: list.getNodeAt(2)

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

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

Idea of Inserting a Node at index i

Case: addAt(i, e), where $0 < i \leq size$

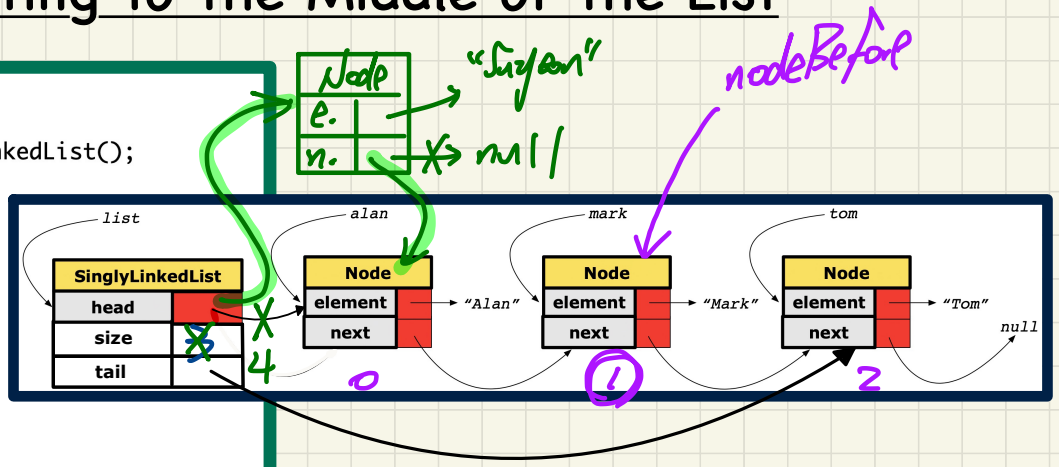


SLL Operation: Inserting to the Middle of the List

```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());
}
```



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

```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 }
```

O(n)