Lecture 10 - Monday, February 13

Announcements

- Assignment 2 released
  - Required & Recommended Studies
  - Looking Ahead: Programming Test 1
- Assignment 1 solution released

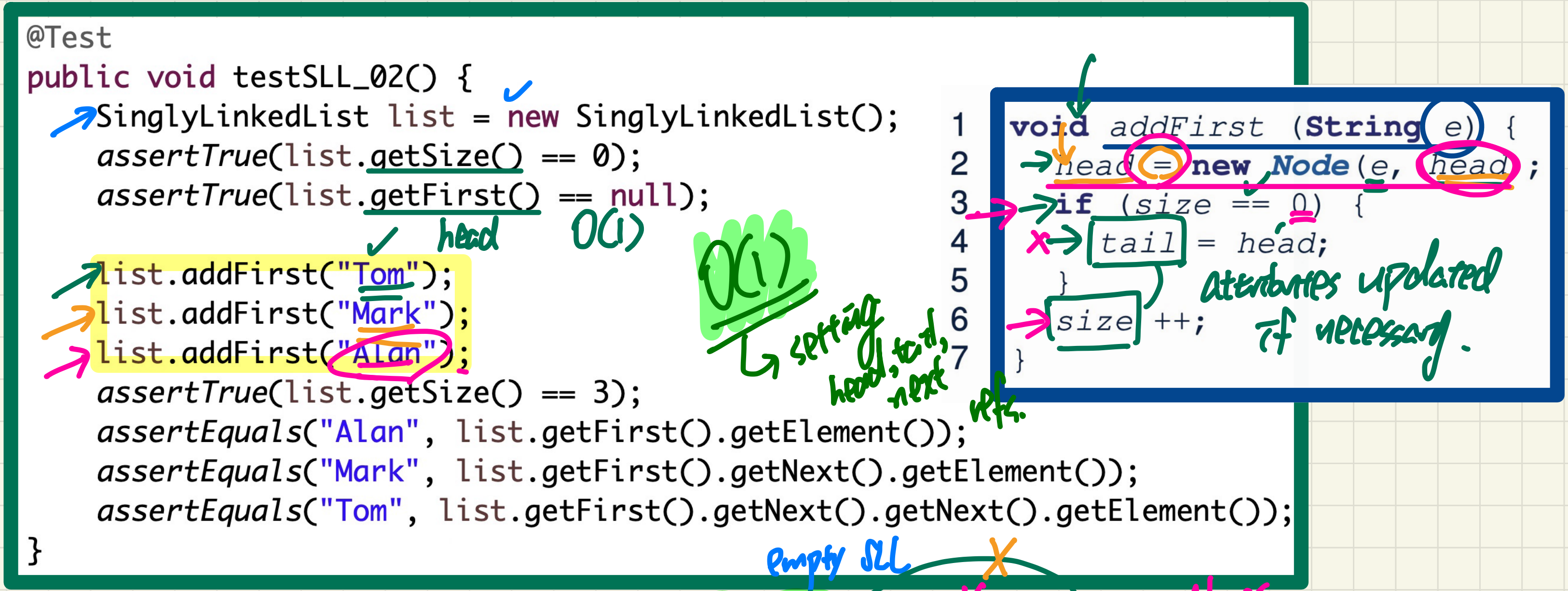
Assume

SLL class: head, tail, size

attributes → $O(1)$

Catch: for methods that might impact the head, tail, or size of a SLL, the body of the method should update these attributes accordingly.

SLL Operation: Inserting to the Front of the List



```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());
}
```

head

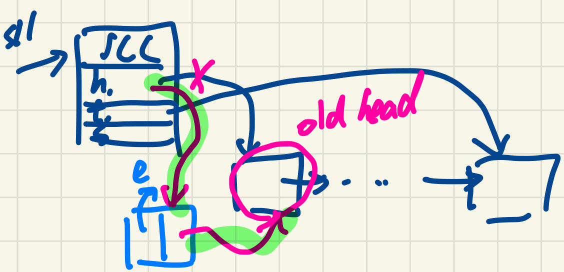
O(1)

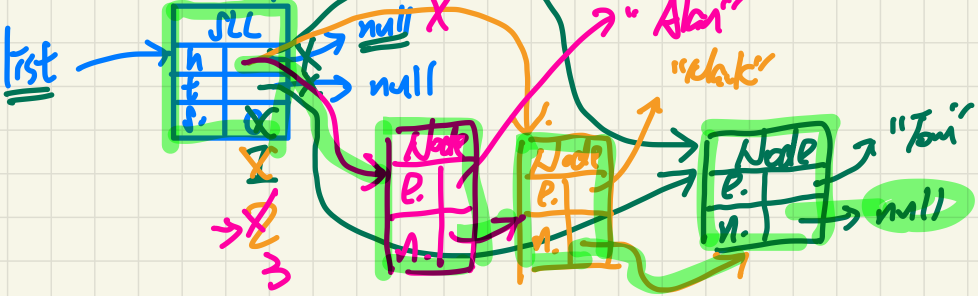
O(1)

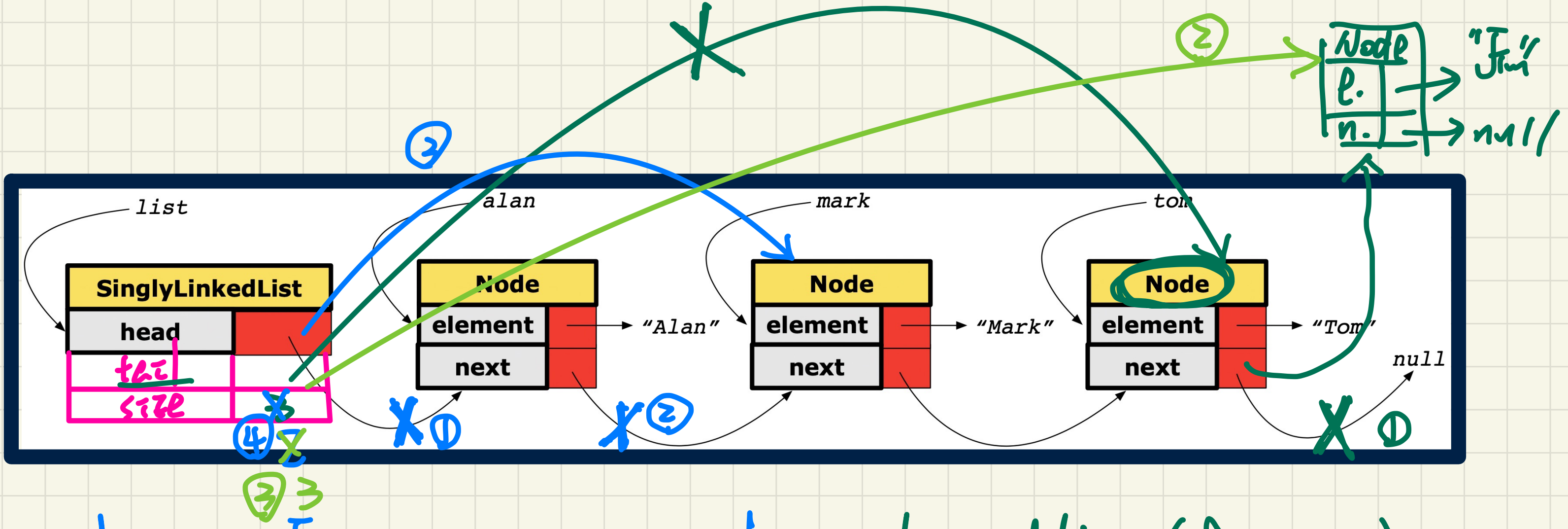
↳ setting head, tail, next refs.

```java
void addFirst (String e) {
  head = new Node(e, head);
  if (size == 0) {
    tail = head;
  }
  size ++;
}
```

attributes updated if necessary.







SLL

void removeFirst( )

① ② ③ ④

If size == 1

↳ after removal, list becomes empty

↳ tail = null

If size == 0

↳ throw some exception.

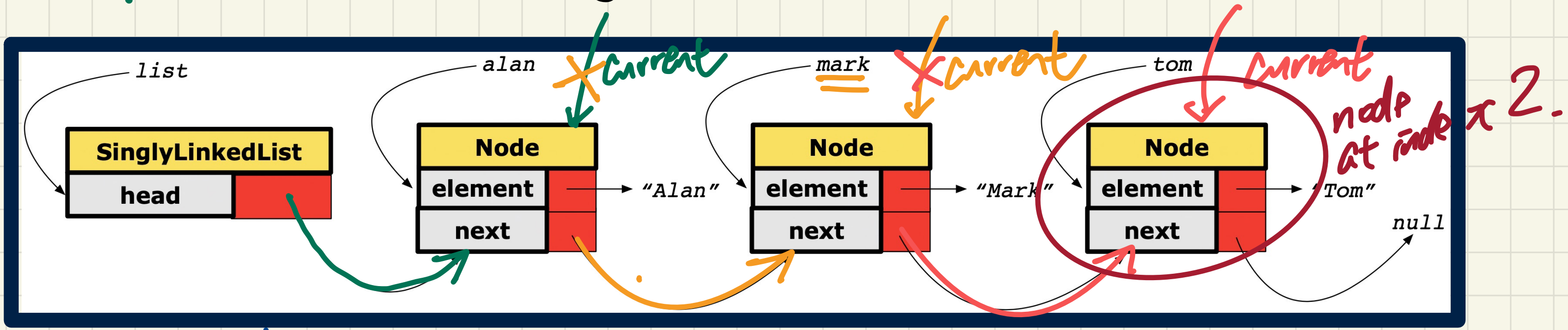
void addLast(String e)

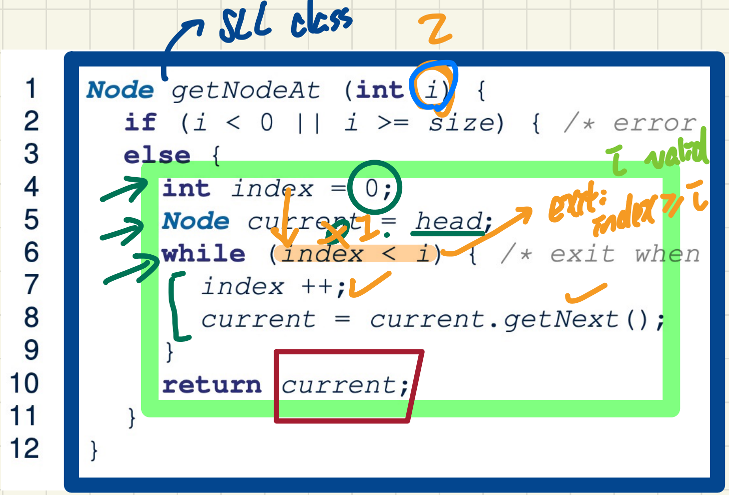
list.addLast("Jim")

①, ②, ③

if size == 0
addFirst.

SLL Operation: Accessing the Middle of the List





```
Node getNodeAt (int i) {
   if (i < 0 || i >= size) { /* error
   else {
      int index = 0;
      Node current = head;
      while (index < i) { /* exit when
         index ++;
         current = current.getNext();
      }
      return current;
   }
}
```

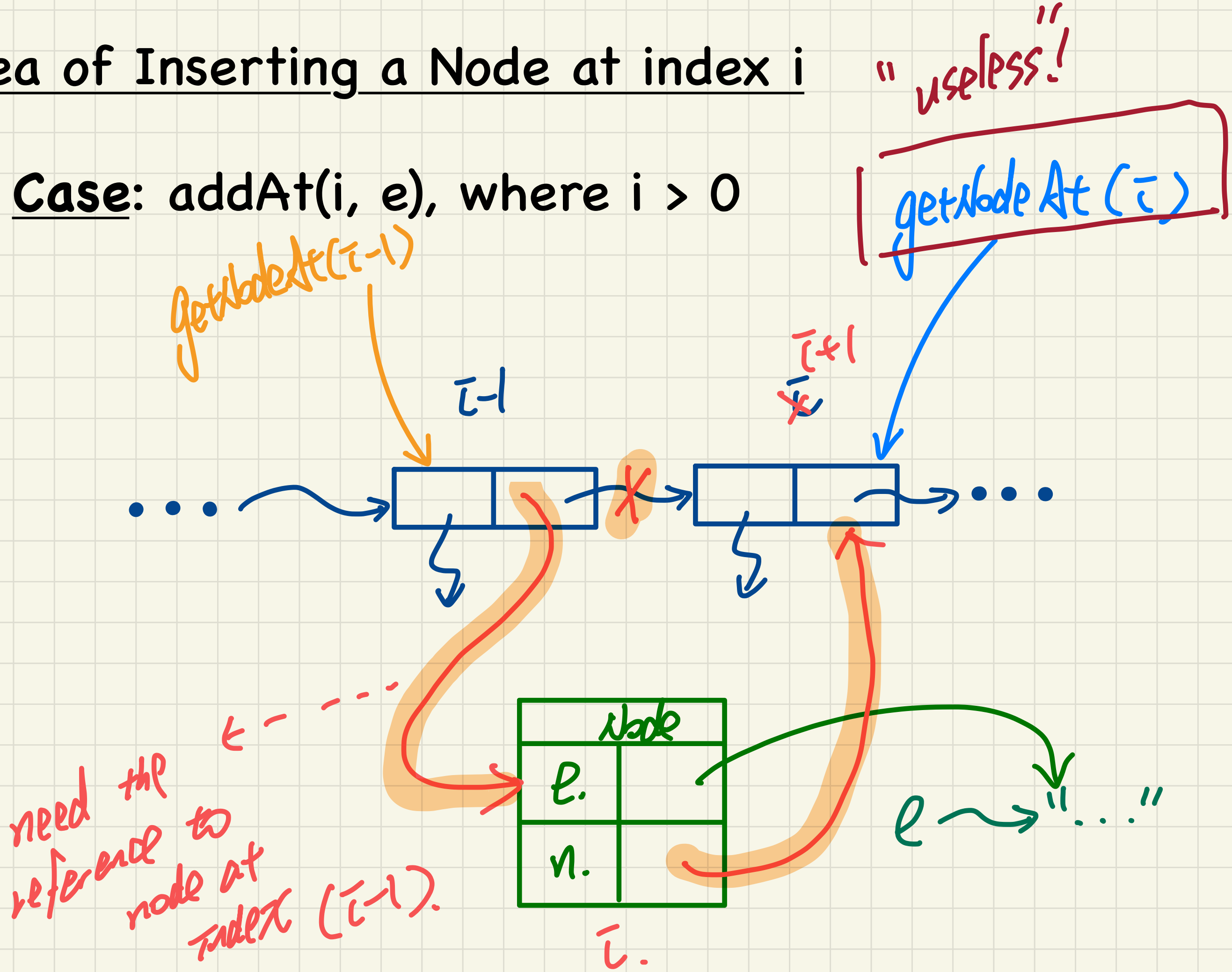
Trace: list.getNodeAt(2)

current	index	index < 2	Start of Iteration
alan	0	0 < 2	1: index 0 → 1 current → mark
mark	1	1 < 2	2: index: 1 → 2 current → tom
tom	2	2 < 2 F	

RT: worst is when i = list.size() - 1 O(n)

Idea of Inserting a Node at index i

Case: addAt(i, e), where i > 0

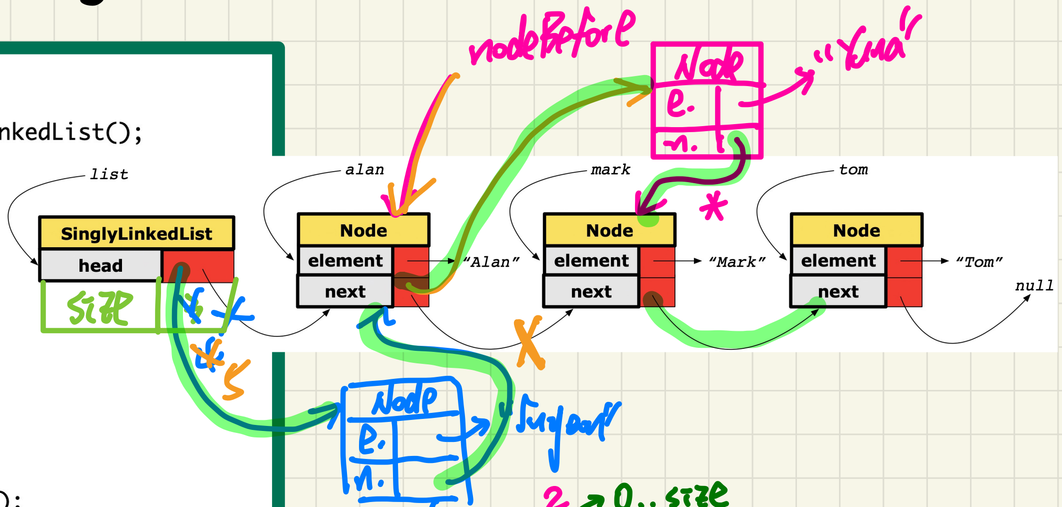


SLL Operation: Inserting to the Middle of the List

```
@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());
}
```



2 → 0..size

```
void addAt (int i, String e) {
  if (i < 0 || i > size) {
   throw new IllegalArgumentException("Invalid Index.");
  }
  else {
    if (i == 0) {
      addFirst(e);
    }
    else {
      Node nodeBefore = getNodeAt(i - 1);
      Node newNode = new Node(e, nodeBefore.getNext());
      nodeBefore.setNext(newNode);
      size ++;
    }
  }
}
```

getNodeAt(1)

O(n)

O(n)
↳ dominated by finding node at index (i-1)