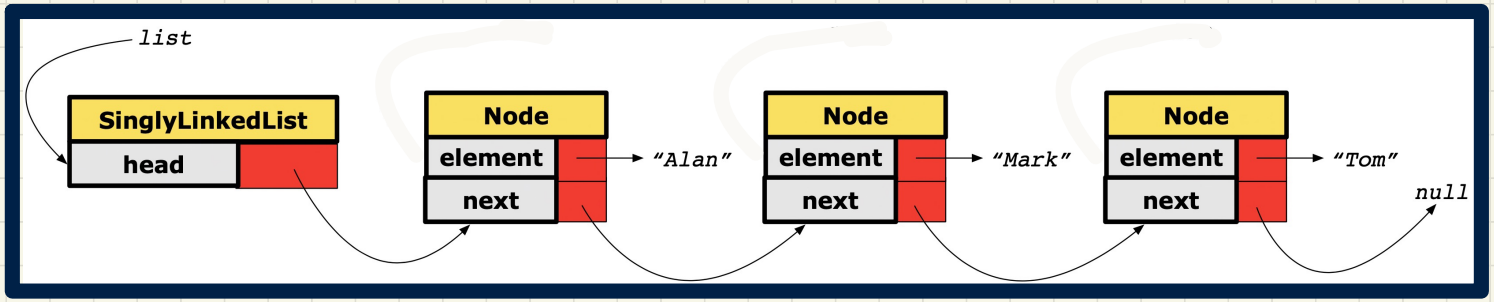
SLL: Setting a List's Head to a Chain of Nodes



Approach 3

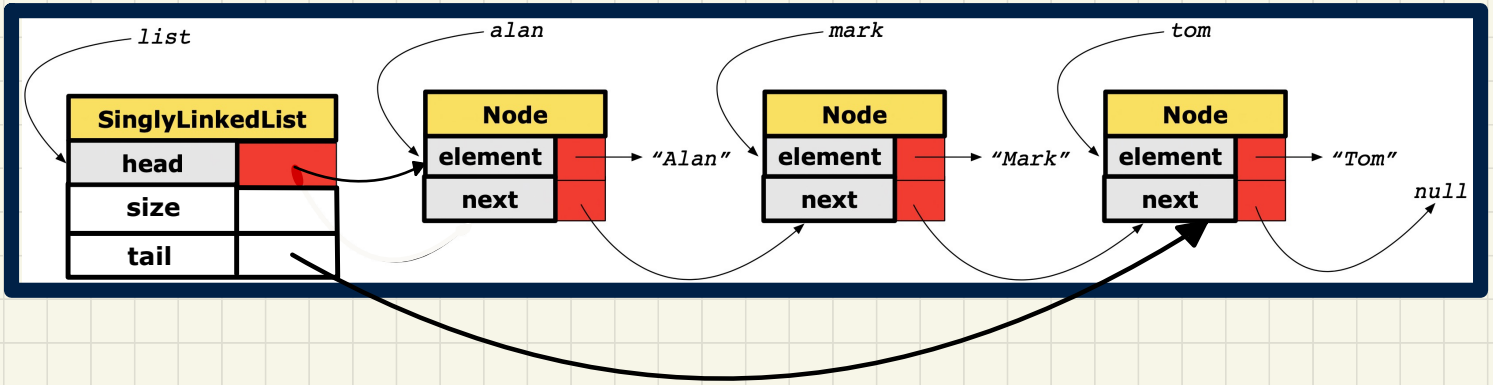
Q. Given: SinglyLinkedList list = **new** SinglyLinkedList();
Write a **single line** of Java code to construct the above chain.

SLL Operation (sketch): Removing the First Node

void removeFirst()

General Cases?

Boundary Cases?

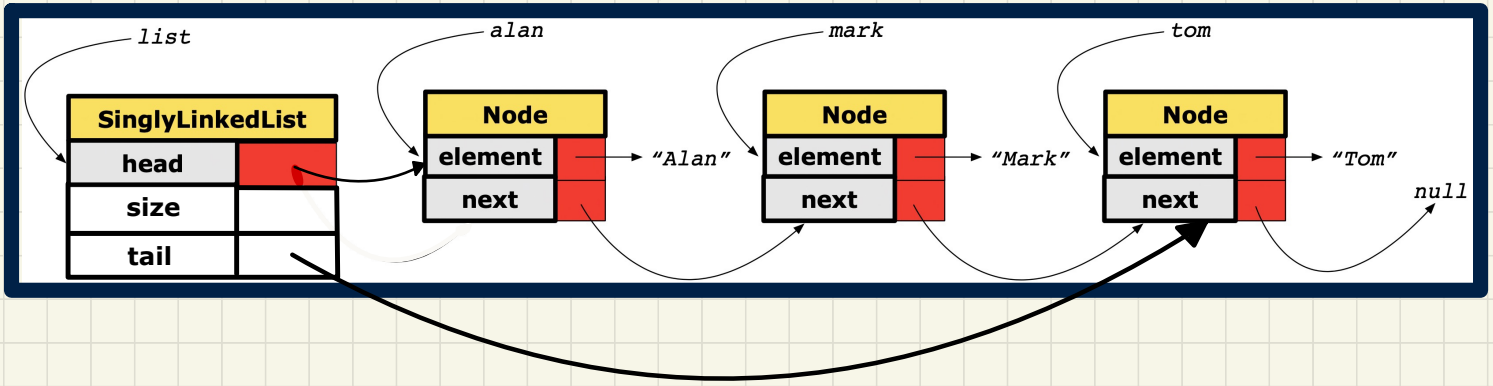


SLL Operation (sketch): Adding a Last Node

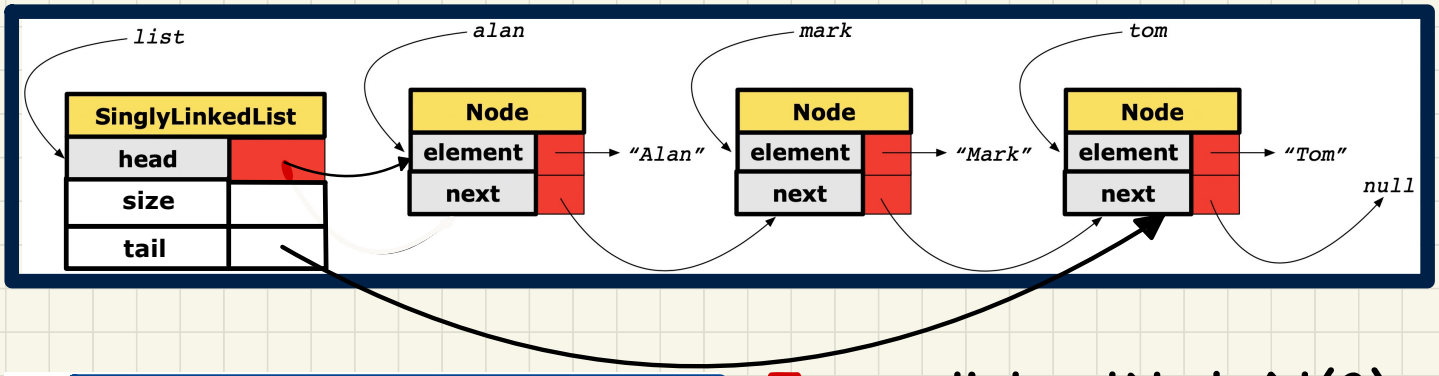
void addLast(String e)

General Cases?

Boundary Cases?



SLL Operation: Accessing the Middle of the List




```
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  }
```

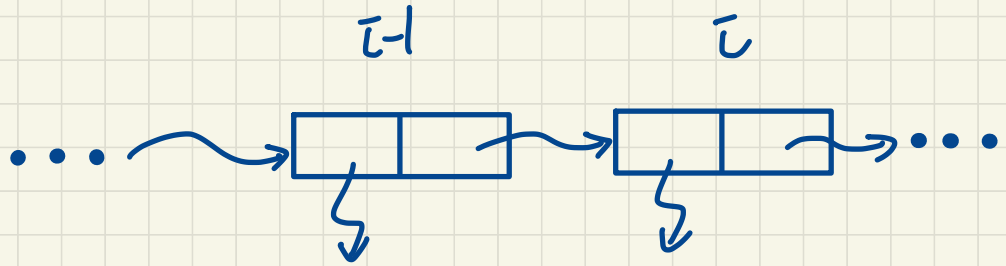
Trace: list.getNodeAt(2)

current	index	index < 2	Start of Iteration

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

Idea of Inserting a Node at index i

Case: addAt(i, e), where 0 < i ≤ size



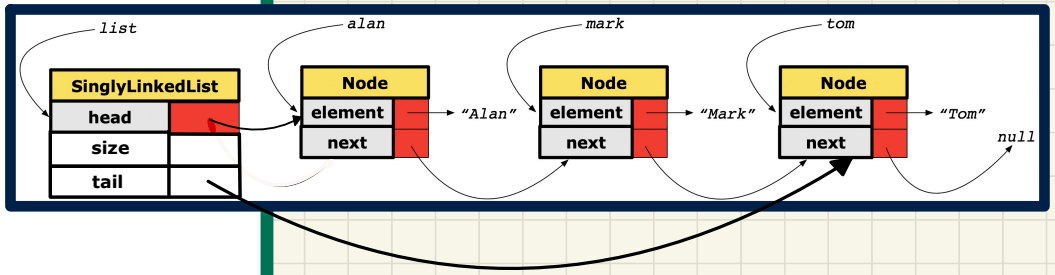
e → "..."

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
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 ++;
    }
  }
}
```

SLL Operation: Removing the End of the List

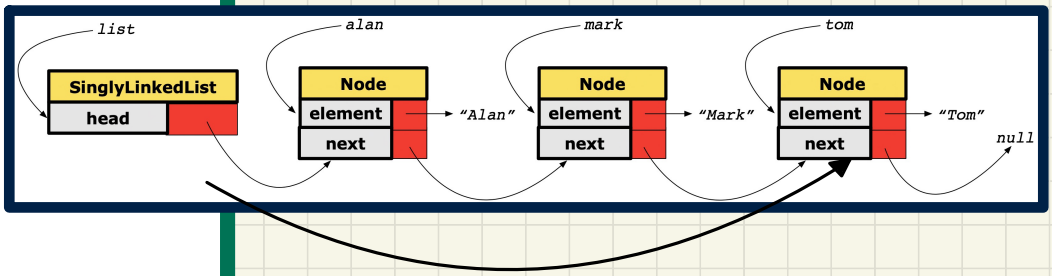
```java
@Test
public void testSLL_removeLast() {
    SinglyLinkedList list = new SinglyLinkedList();
    assertTrue(list.getSize() == 0);
    assertTrue(list.getFirst() == null);

    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());
}
```

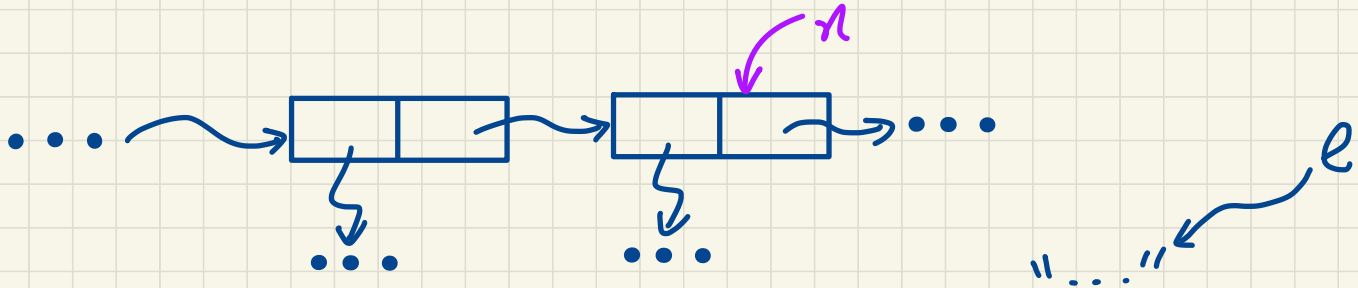


Q. Does **tail** or **size** need to be updated?

```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 }
```

Exercises: insertAfter vs. insertBefore

Case: insertAfter(Node n, String e)



Case: insertBefore(Node n, String e)