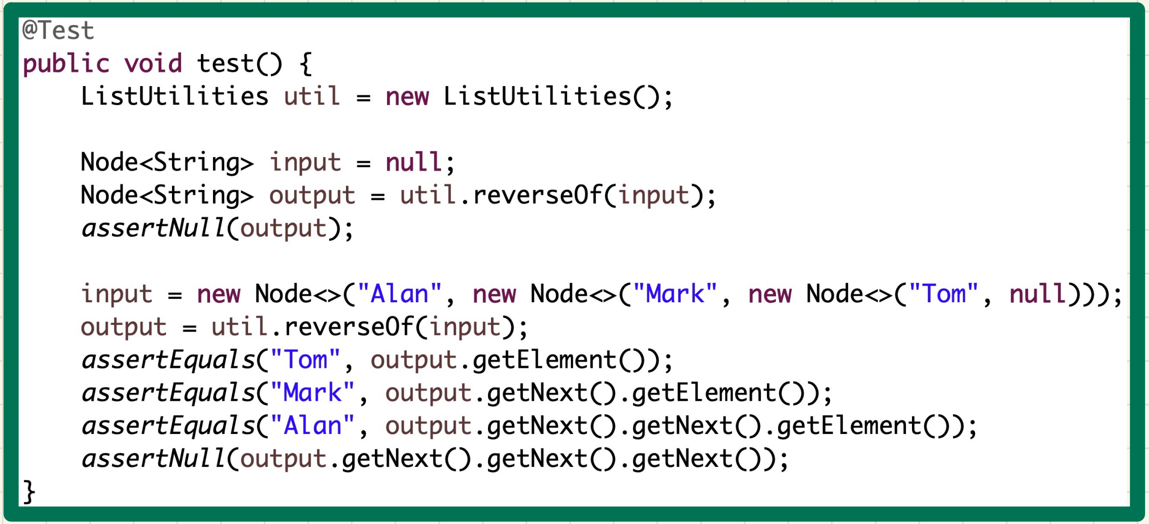
Problem on SLL: Reversing a Chain of Nodes

You are asked to program this method:

public Node<String> reverseOf(**Node<String>** input)

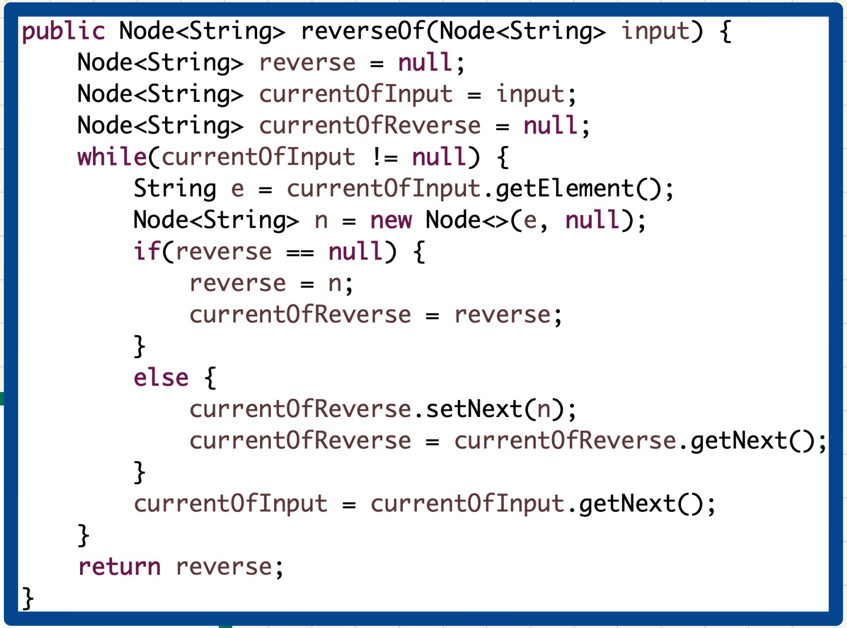
The returned node references the front of a separate chain of nodes representing the reverse of the input.

Requirement: The input node may or may not be null.

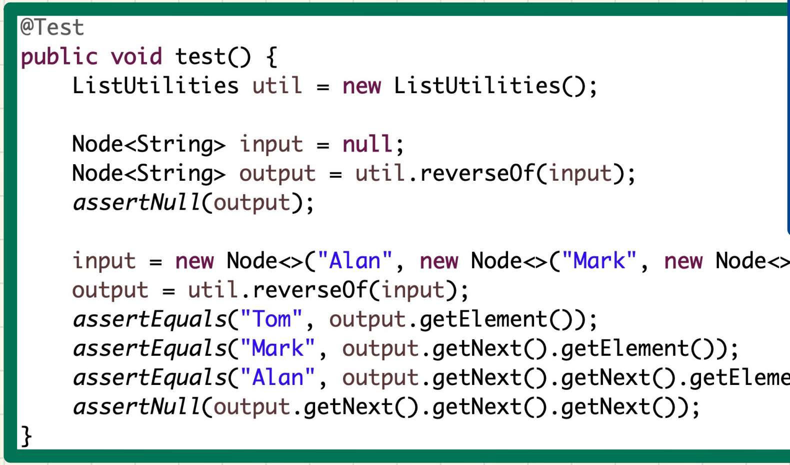


```
@Test
public void test() {
    ListUtilities util = new ListUtilities();

    Node<String> input = null;
    Node<String> output = util.reverseOf(input);
    assertNull(output);

    input = new Node<>("Alan", new Node<>("Mark", new Node<>("Tom", null)));
    output = util.reverseOf(input);
    assertEquals("Tom", output.getElement());
    assertEquals("Mark", output.getNext().getElement());
    assertEquals("Alan", output.getNext().getNext().getElement());
    assertNull(output.getNext().getNext().getNext());
}
```

Problem on SLL: Reversing a Chain of Nodes?



```java
public Node<String> reverseOf(Node<String> input) {
    Node<String> reverse = null;
    Node<String> currentOfInput = input;
    Node<String> currentOfReverse = null;
    while(currentOfInput != null) {
        String e = currentOfInput.getElement();
        Node<String> n = new Node<>(e, null);
        if(reverse == null) {
            reverse = n;
            currentOfReverse = reverse;
        }
        else {
            currentOfReverse.setNext(n);
            currentOfReverse = currentOfReverse.getNext();
        }
        currentOfInput = currentOfInput.getNext();
    }
    return reverse;
}
```



```java
@Test
public void test() {
    ListUtilities util = new ListUtilities();

    Node<String> input = null;
    Node<String> output = util.reverseOf(input);
    assertNull(output);

    input = new Node<>("Alan", new Node<>("Mark", new Node<>("Tom", null)));
    output = util.reverseOf(input);
    assertEquals("Tom", output.getElement());
    assertEquals("Mark", output.getNext().getElement());
    assertEquals("Alan", output.getNext().getNext().getElement());
    assertNull(output.getNext().getNext().getNext());
}
```

Is this the correct algorithm for reversing a chain of nodes?