

```
public interface StringList {
    /* Add an element at the beginning of the list */
    void prepend(String s);           // (new!)
    /* Add an element at the end of the list */
    void add(String s);
    /* Get the element at the given index */
    String get(int index);
    /* Get the number of elements in the list */
    int size();
    /* Add an element at the specified index */
    void insert(int index, String s);
    /* Remove the element at the specified index */
    void remove(int index);
}
```

```
import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class TestStringList {

    @Test
    public void testAddThenGet() {
        StringList slist = new      LinkedList();
        slist.  prepend ("banana");
        slist.  prepend ("apple");

        assertEquals(          , slist.get(0));

        assertEquals(          , slist.get(1));
    }

    @Test
    public void testAddThenSize() {
        StringList slist = new      LinkedList();
        slist.  prepend ("banana");
        slist.  prepend ("apple");

        assertEquals(          , slist.size());
    }
}
```

```
class Node {
    String value;
    Node next;
    public Node(String value, Node next) {
        this.value = value;
        this.next = next;
    }
}

public class      LinkedList      implements StringList {

    Node front;

    // How will we construct it?

    // How will we implement the methods?
    // Focus on .prepend(), .get(), and the diagram first

}
```

| | |
|----------------------|----------|
| @A.prepend("banana") | |
| this | @A |
| s | "banana" |
| returns: | |

| | |
|----------------|--|
| .prepend() | |
| this | |
| s | |
| returns: | |

| | | | |
|----|----------------------|--|--|
| @A | LinkedList front: | | |
| | | | |
| | | | |

```
public class TestStringList {
    ... all code from other side ...
    @Test
    public void testAddMany() {
        StringList slist = new LinkedList();
        slist.add("m");
        slist.add("n");
        slist.add("o");
        // memory diagram here!
        slist.add("p");

        assertEquals("p", slist.get(3));
    }
}
```

```
public class LinkedList {
    ... all code from other side ...
    // Now focus on .add() and .size()

}
```

| | | | |
|----|---------------------------------|----|----------------------------------|
| @A | LinkedList front: @B | @C | Node value: "m" next: @D |
| @B | Node value: null next: @C | @D | Node value: "n" next: @E |
| | | @E | Node value: "o" next: null |
| | | | |
| | | | |

| | |
|-------------|--|
| @A.add("p") | |
| this | |
| s | |
| | |
| | |
| | |
| returns: | |

Key Linked list idea: elements are stored in nodes that track a value and a reference to the node after them.