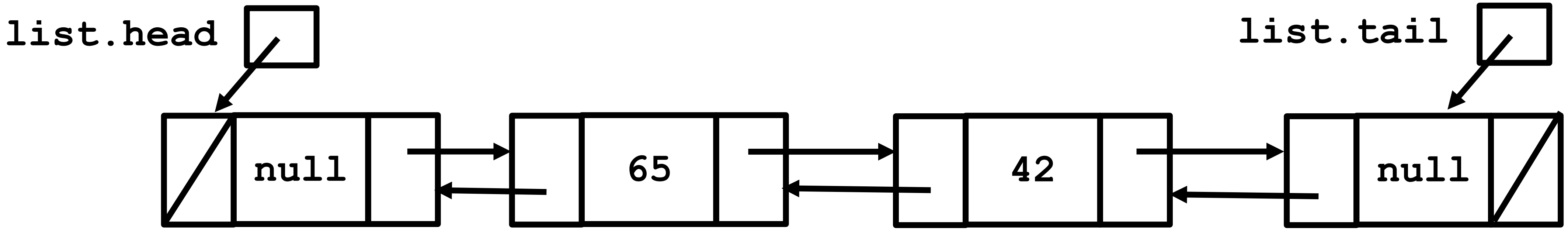
Testing in Practice: Linked Lists

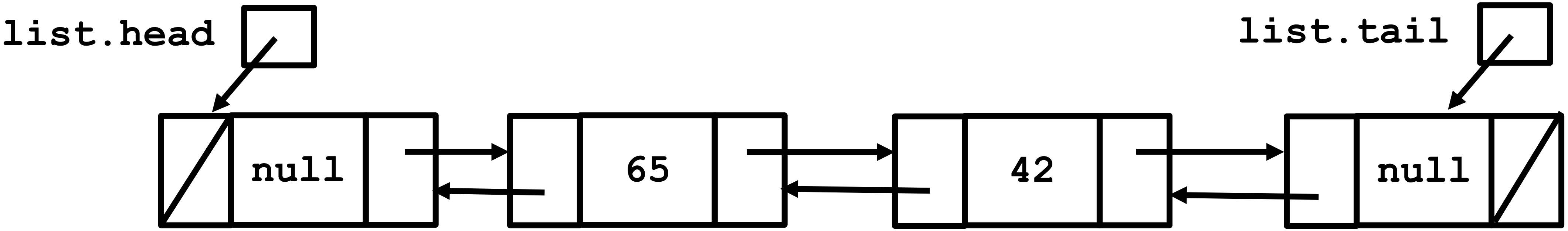
By the end of this video you will be able to…

- Write tests for a Linked List

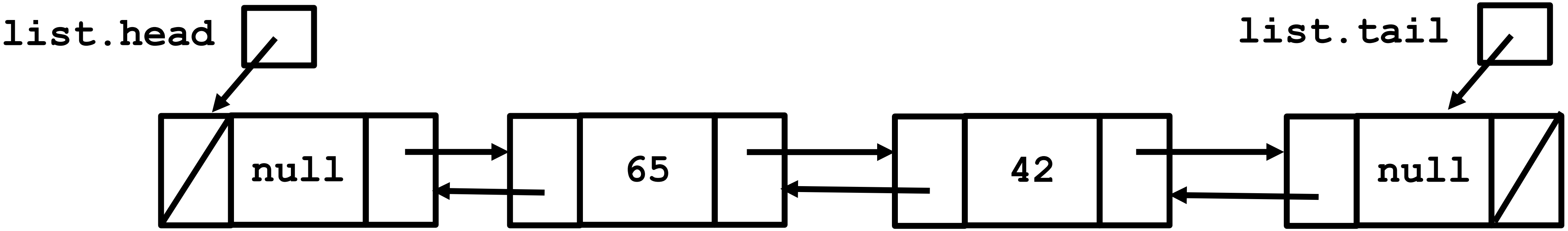
Doubly Linked List



Doubly Linked List



```
// retrieves the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```



```
// retrieves the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run (select all)? Try to avoid redundant tests.

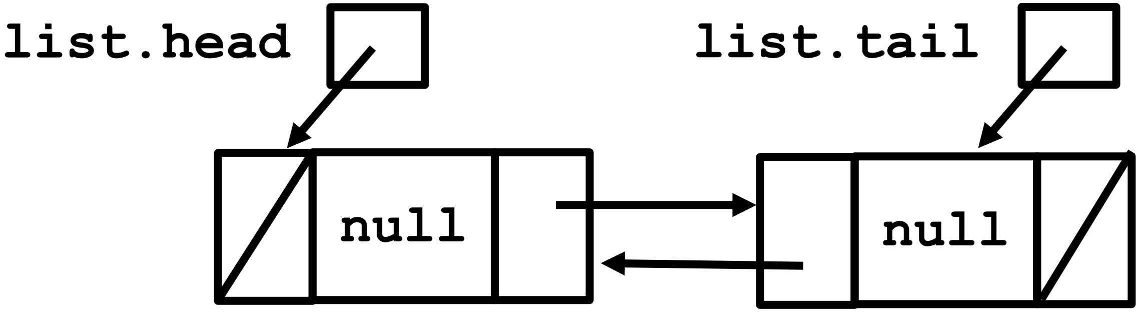
A. Test get(0) from an empty list

B. Test get(-1) from a list with 1 element

C. Test get(0) from a list with 1 element

D. Test get(1) from a list with 2 elements

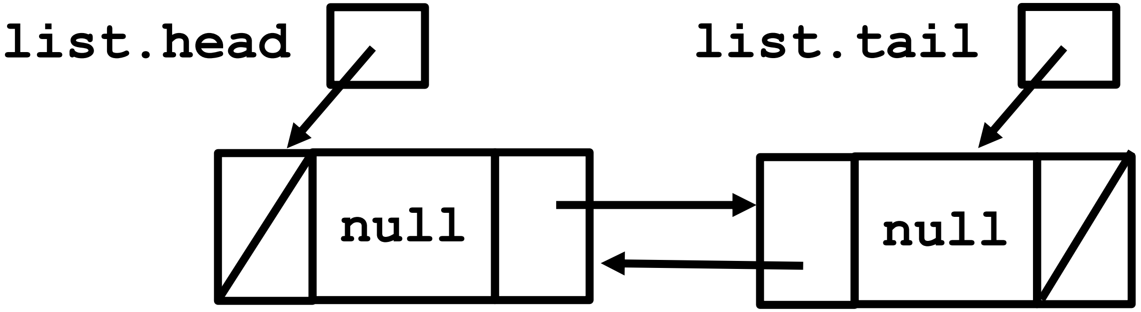
E. Test get(2) from a list with 3 elements



```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

A. Test get(0) from an empty list

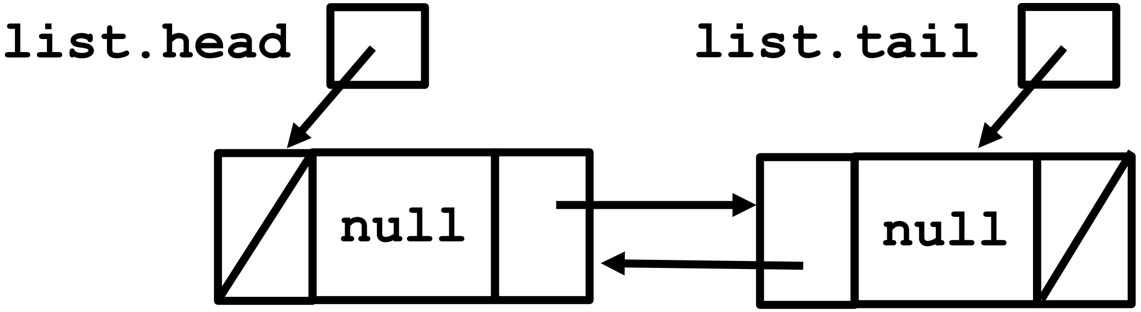


```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

A. Test get(0) from an empty list ✓

Tests corner case (empty list)

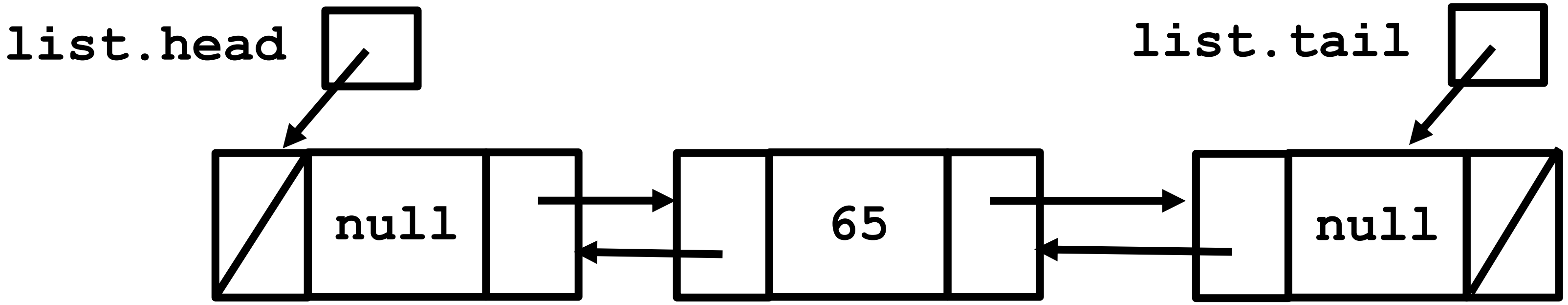


```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

A. Test get(0) from an empty list ✓

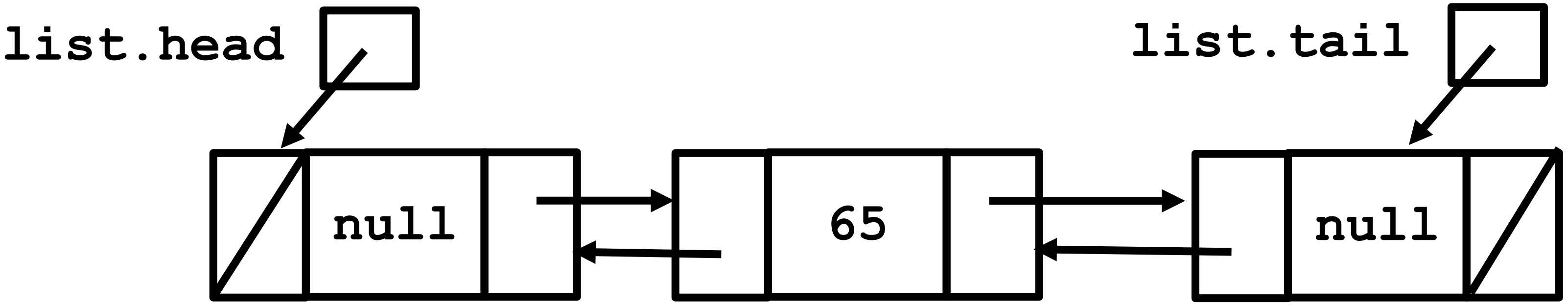
```
// in testGet (in JUnit @Test)
try {
  emptyList.get(0);
  fail("Check out of bounds");
}
catch (IndexOutOfBoundsException e) {
}
```



```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

B. Test get(-1) from a list with 1 element

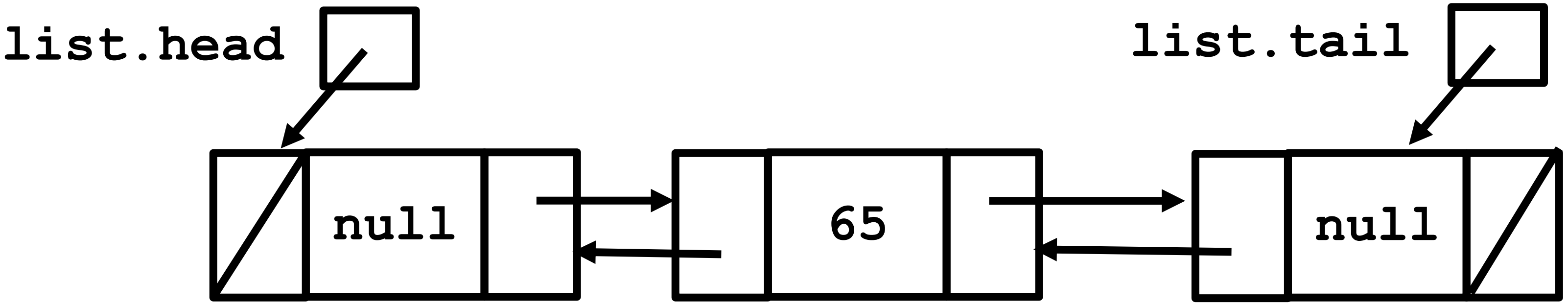


```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

B. Test get(-1) from a list with 1 element ✓

Tests corner case (negative index)

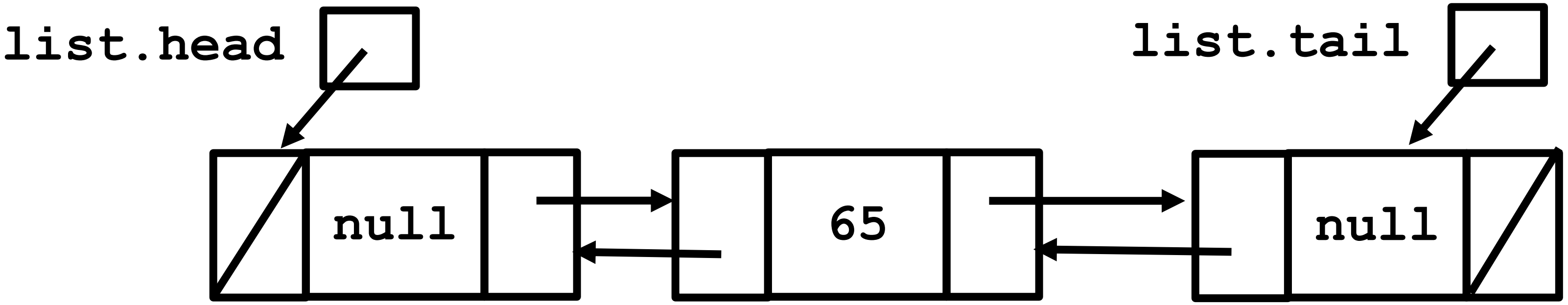


```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

B. Test get(-1) from a list with 1 element ✓

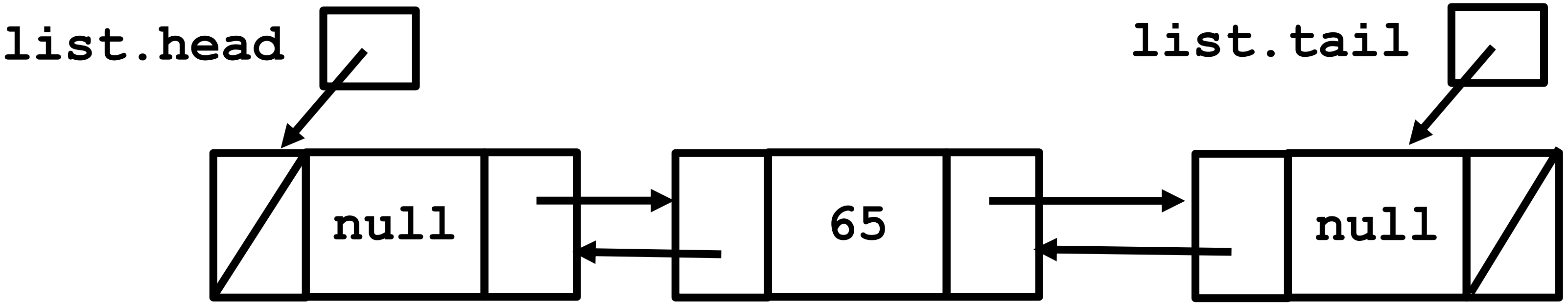
```
// in testGet (in JUnit @Test)
try {
  shortList.get(-1);
  fail("Check out of bounds");
}
catch (IndexOutOfBoundsException e) {
}
```



```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

C. Test get(0) from a list with 1 element

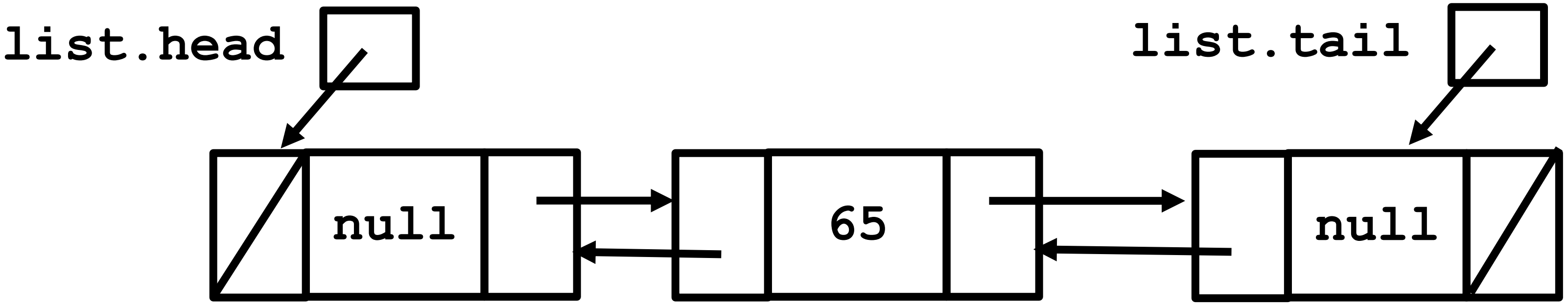


```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

C. Test get(0) from a list with 1 element ✓

Tests standard use

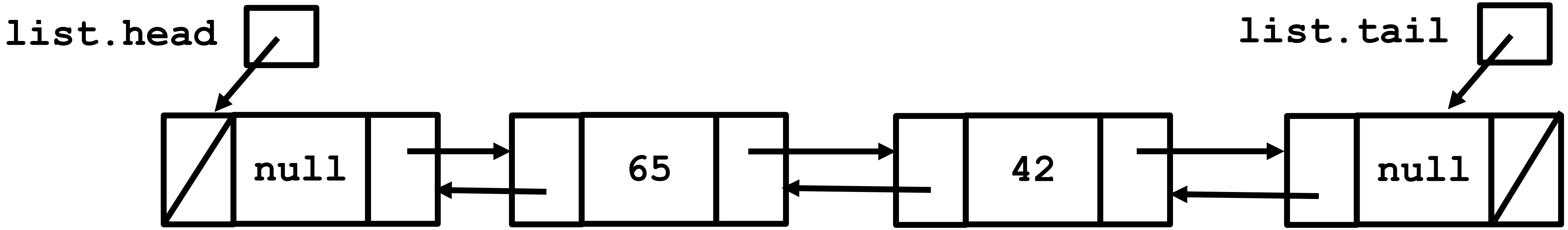


```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

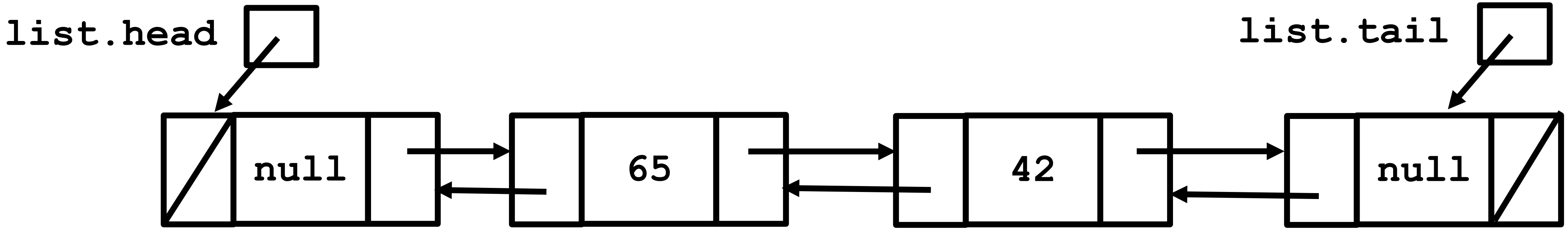
C. Test get(0) from a list with 1 element ✓

```
// in testGet (in JUnit @Test)
assertEquals("Check first", (Integer)65,
             shortList.get(0));
```



```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

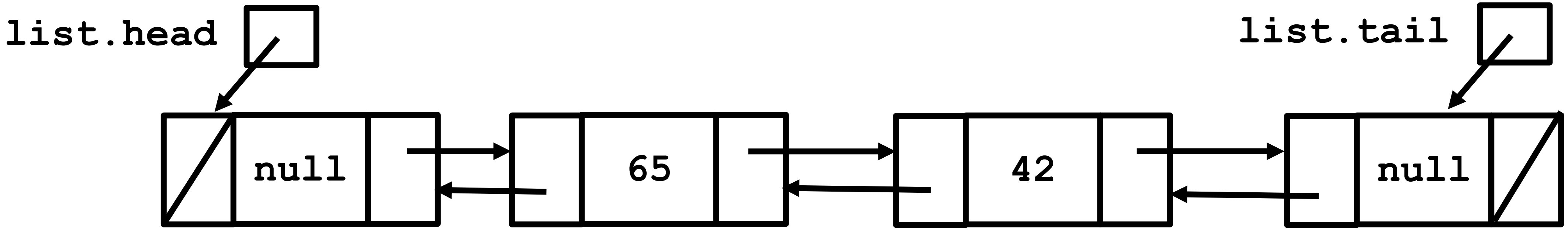
Which of the following tests should I run?
D. Test get(1) from a list with 2 elements



```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?
D. Test get(1) from a list with 2 elements ✓

Ensures we can get more than just the 1st element



list.head

null

65

42

null

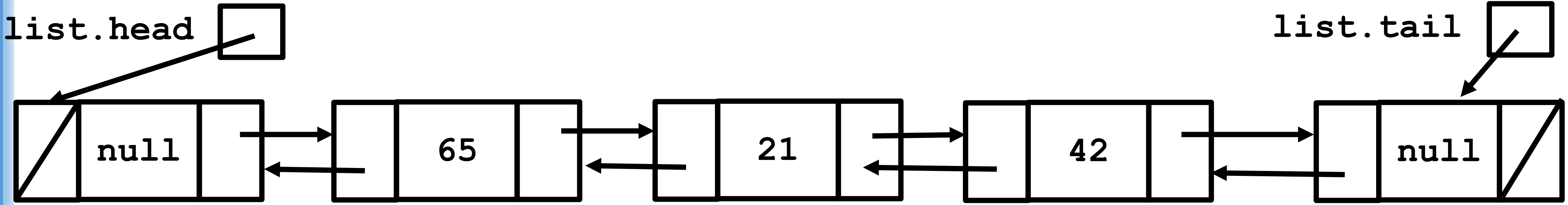
list.tail

```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

D. Test get(1) from a list with 2 elements ✓

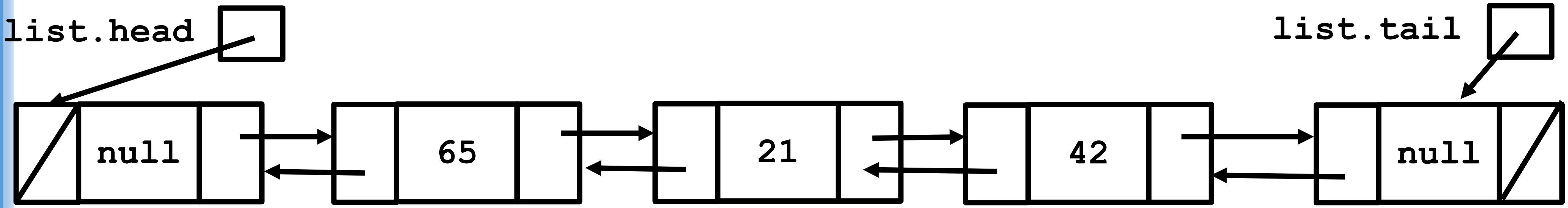
```
// in testGet (in JUnit @Test)
assertEquals("Check second", (Integer)42,
             shortList2.get(1));
```



```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

E. Test get(2) from a list with 3 elements



```
// removes the element at the index indicated or, if the index
// is invalid, throw and IndexOutOfBoundsException
public E get(int index)
```

Which of the following tests should I run?

E. Test get(2) from a list with 3 elements ✗

Redundant, what is new here?

Summary

- Consider corner cases when testing
- Test common case use
- Remember testing has costs, particularly tests which are run repeatedly (like unit tests).

Better to err on the side of caution, but be careful.