Jake Lepere - Scribe

Johnny Chi - Driver

11-06-2015

Lab 10

PartA

1. Done
2. Create a node

Loop through I = 0..n

If I == n return node.data

1. Driver
2. We get an error because the get method returns null
3. Driver
4. We pass the test because n is not greater than the amount of elements in the list

7.

@Test(expected=IndexOutOfBoundsException.**class**) **public** **void** outOfBoundsLess() {

sampleList.get(-1);

}

@Test(expected=IndexOutOfBoundsException.**class**) **public** **void** outOfBoundsMore() {

sampleList.get(6);

}

1. We pass both the test cases because our get method threw an exception if the counter never reached n, whether n < 0 or n > number of elements.
2. We know we are making progress because one of the tests passes
3. If the loop is fully executed and it we have not found the nth element than n is larger than the number of elements, so we will throw and exception.
4. Driver
5. We pass both test cases because we check for those scenarios.

PartB

1. Driver
2. Done
3. We cannot use the exact loop because we aren’t returning anything and we need to create a new Node for the element we are adding.

4.

**if** (n < 0) {

**throw** **new** IndexOutOfBoundsException();

}

**if** (n == 0) {

addFirst(x);

} **else** {

Node node = **new** Node();

node = first;

**int** i = 0;

**while** (node != **null**) {

**if** (i == n-1) {

Node previous = node;

Node newNode = **new** Node();

newNode.data = x;

newNode.next = previous.next;

previous.next = newNode;

}

node = node.next;

i ++;

}

}

1. Yes we passed
2. If n = 0

@Test

**public** **void** addZero()

{

sampleList.add(0, "Mrs.");

*assertEquals*("Mrs.", sampleList.get(0));

}

1. Yes it passed
2. We added a case to check if n == 0, and if so we used the addFirst method to add the element x to the beginning of the list. Otherwise we added normally.
3. Yes they all passed

PartC

1. We passed the removeInMiddle test, but not the removeHead or removeTail tests.
2. We are missing a case where we are removing at the head or at the tail, ie the first and the last element

The variable this has samplelist of the nodes: Mary, had, a, little, lamb

Iter has the variables isAfterNext = true, isAfterPrevious = false, and a position which is currently the node with data “Mary”, next the node with data “had”, and a previous of null.

1. First has an id=33, and iter has an id=34
2. Done
3. They have values the same as the node previous to there position. This makes sense because they have the id is referencing the same object.
4. Done
5. The second if statement gets executed, calling: positionToRemove.next.previous = positionToRemove.previous;

The third if stamement is also executed calling:

position = position.previous;

1. They differ because the previous variable for the node with data “had” points to a null previous.
2. Mary should have been removed but it has not been.
3. Driver
4. What bug? Our code passed.