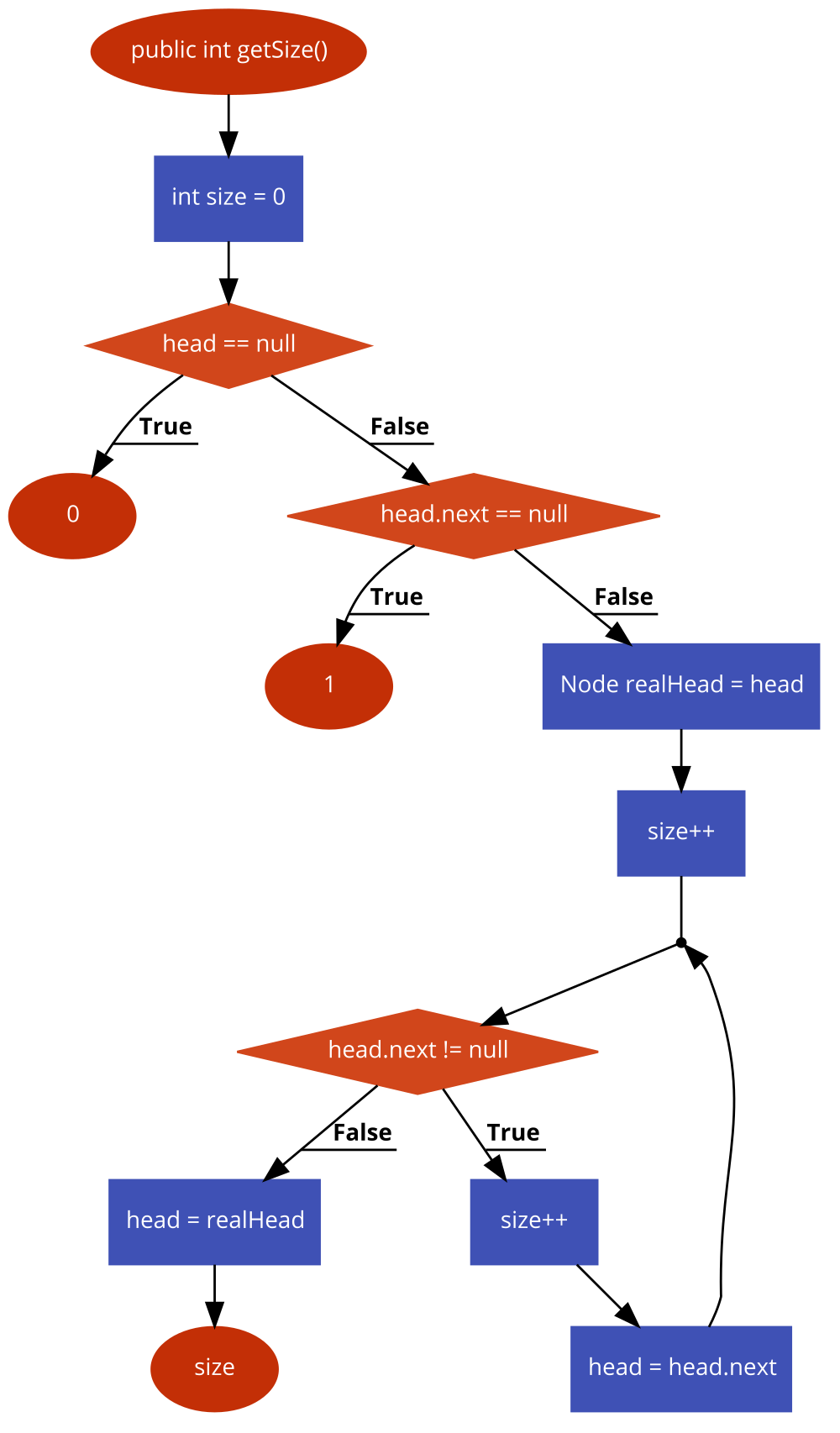
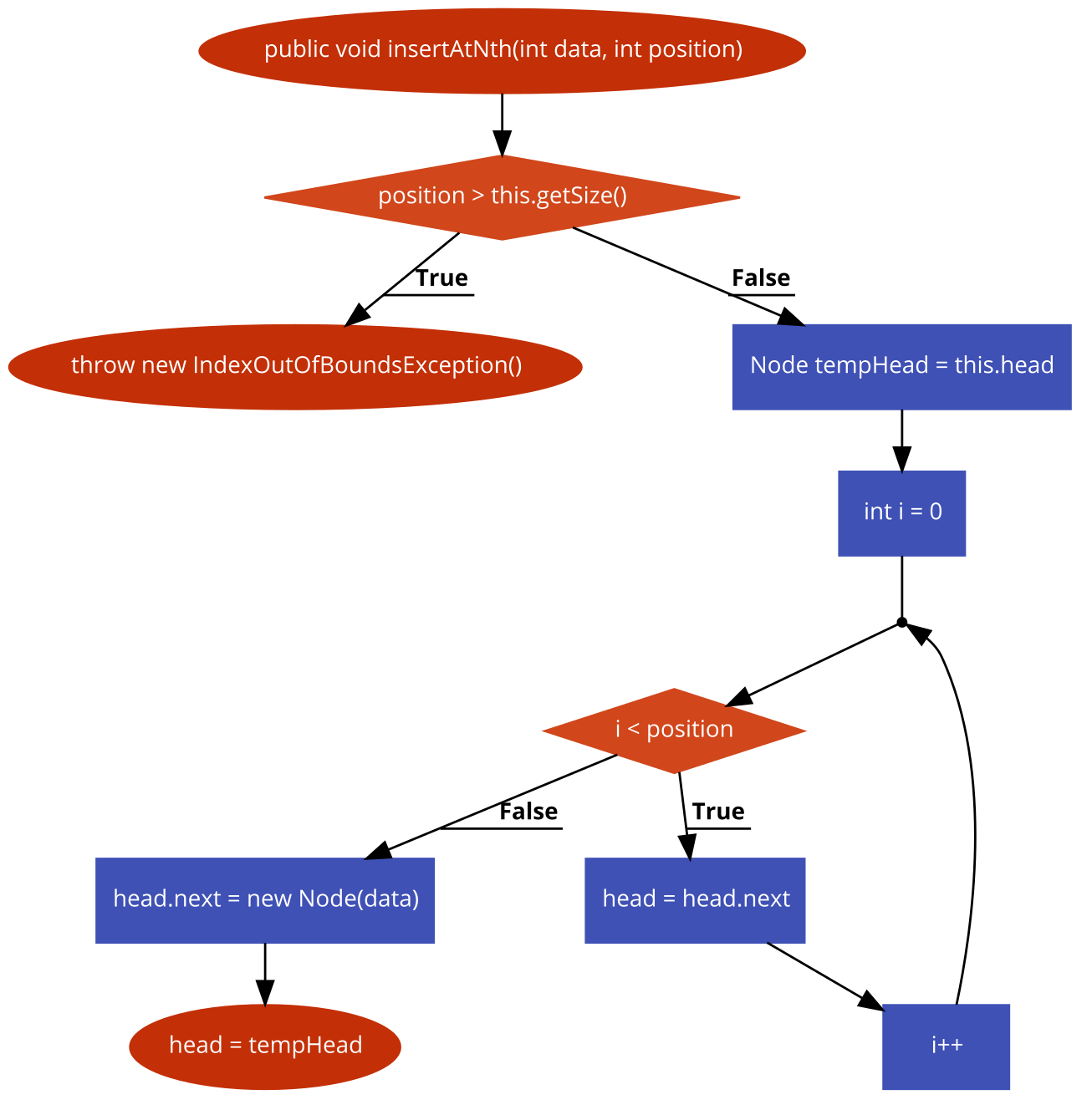
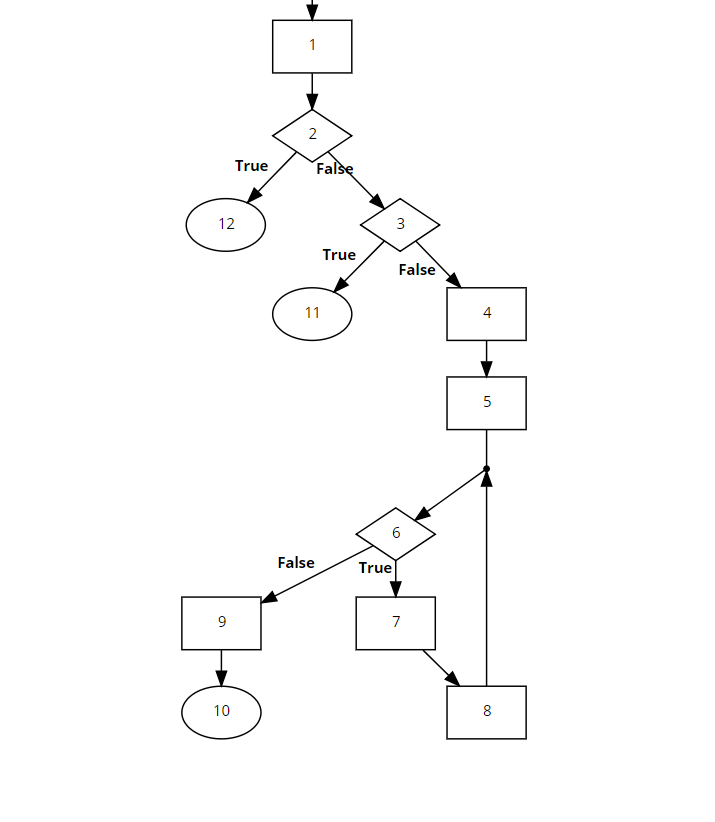
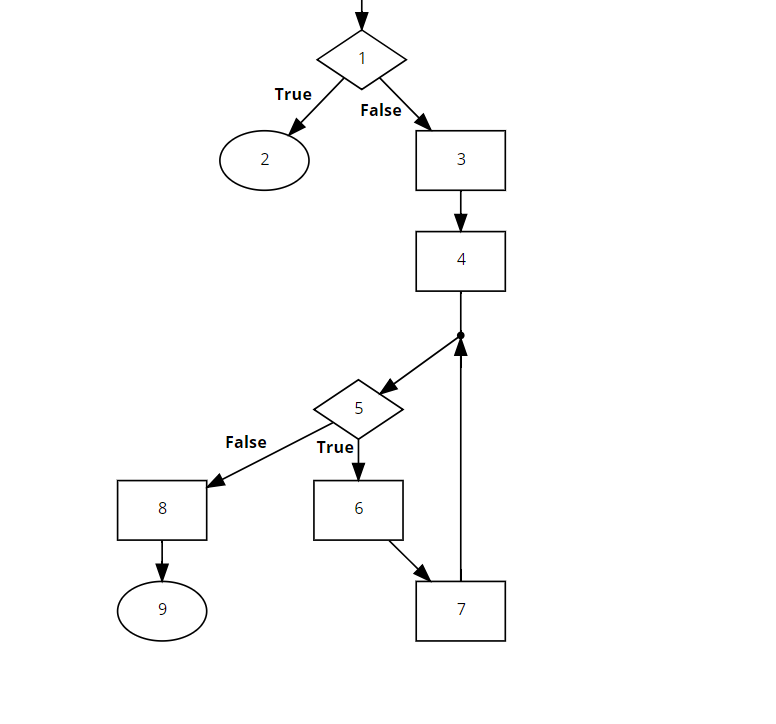
## Singly Linked List Testing:

Authored by Michael Gentile





The other graphs are singular nodes with no decision points. Therefore, they would simply be a singular node. All four graphs are shown to narrate test paths appropriately.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method Being Tested | Criteria (Prime Path) | Test Input | Narrative | Author | Result |
| insertAtNth() | [1,3,4,5,6,7], [6,7,5,8,9], [5,6,7,5], [7,5,6,7], [6,7,5,6] | A list of size five is made. An element is then added after the second index. | This is to test adding to the middle of a list. This encompasses all of the paths for longer than one sized lists and adding to the middle of it. | Michael Gentile | Fail |
| insertAtNth() | [1,3,4,5,8,9] | A list of size five had an element added at index 0. | This tests for adding to the beginning of the list. | Michael Gentile | Fail |
| insertAtNth() | [1,2] | A list of size of 5 attempts to have an element added to the 10th slot. | This checks for an index out of bands when things are attempted to be added when above the size of the list. | Michael Gentile | Pass |
| getSize() | [1,2,3,4,5,6,9,10] | A list of size two is made. | This covers the prime path of a list with two elements. This path is different than other multielement lists. | Michael Gentile | Pass |
| getSize() | [1,2,3,4,5,6,7,8], [7,8,6,9,10],[7,8,6,7], [8,6,7,8], [6,7,8,6] | A three element list is made. | This covers all of the prime paths for a multiple element (>2) in the case of this code. | Michael Gentile | Pass |
| getSize() | [1,2,3,11] | A list of size one is made. | This checks the list of size one. | Michael Gentile | Pass |
| getSize() | [1,2,12] | No input. | This checks the list of size zero. | Michael Gentile | Pass |
| insertHead() | [1] | Insert a 1. | This checks to make sure the only node is workable. Nothing changes based on other outside factors. | Michael Gentile | Pass |
| deleteHead() | [1] | Insert a 1 and remove the value. | This checks to make sure that a head can be deleted. | Michael Gentile | Pass |

Resources used: Code2Flow.com and cs.gmu.edu:8443/offutt/coverage/GraphCoverage

### Failure Narratives:

Both failures originate from insertAtNth. This method simply takes a node and places it into the list at a specified position. This fails in two manners. 1. Every value subsequent to the insertion value is removed from the list. 2. The value is inserted at the wrong location. This can be observed from the failure output and what is shown. This could only be seen by observing the entire list in the first case. The second case is immediately observable.

### Running the Tests

These tests are run in a generic JUnit test environment. A viable way of running these tests would be to open the project in eclipse. If you do not have eclipse, please acquire it at <https://www.eclipse.org/downloads/>. From this, you may open or import the project that is included with the source code. To run the tests, simply click on the testing file known as “SinglyLinkedListTest.java” and open the file (within eclipse). You then should run the tests by either clicking the green arrow (the run button) or you may run this with code coverage (green arrow with a red and green bar). The tests should then run and the results should be shown on the left hand side of your screen.