Input Space Partitioning for DoublyLinkedList

Testing by Connor14

To run, make sure that DoublyLinkedList.java and DoublyLinkedListTests.java are in the same Eclipse project and package (default). Run DoublyLinkedListTests.java under JUnit 4. Results will be shown on the left.

All of the tests below used:

- Interface Based Approach
- Each Choice Coverage

Summary:

- Total tests: 7
- Passed tests: 3
- Failed tests: 4
    - All failed with NullPointerException
    - Failures are a result of improper checking of null elements before using them

Notes:

- Custom getSize, getHead, and getTail methods were created for DoublyLinkedList.java so assist with the validation of the tests.

Observations/Conclusions:

- NullPointerExceptions are thrown within the DoublyLinkedList class. These are not intentional exceptions.
- The methods have descriptions of parameters and returns. However, some, like deleteHead, do not accurately describe return behavior
- The user will need to make his own assumptions with regards to the order in which to run the methods.
    - For example, one would assume that running insertTail on an empty list would yield a list with a single item that is both the head AND the tail, like insertHead yields. However, the result is a NullPointerException.

Function: insertHead

Parameters: int x

	Characteristic	b1	b2	b3

c1	Relation of x to 0	Greater than 0	Less than 0	Equal to 0
c2	Size of list	Equals 0	Equals 1	Greater than 1
c3	x exists in list	true	false	

Test Values

TestID	Relation of x to 0	Size of list	x exists in list
1	1	0	false
2	-2	1	true
3	0	2	true

Narrative:
This is to test setting the head of the list (the beginning)

Expected Results (based on method description/interface):
The item will be inserted into the beginning of the list. Existing items will be offset. If the inserted item is the only item, expect it to be the head and the tail.

Results:
PASS

Function: insertTail

Parameters: int x

	Characteristic	b1	b2	b3
c1	Relation of x to 0	Greater than 0	Less than 0	Equal to 0
c2	Size of list	Equals 0	Equals 1	Greater than 1
c3	x exists in list	true	false	

Test Values

TestID	Relation of x to 0	Size of list	x exists in list
1	3	0	false
2	-10	1	true

3	0	2	true

Narrative:
This is to test setting the tail of the list (the end)

Expected Results (based on method description/interface):
The item will be added to the end of the list. Existing items will come before it. If the inserted item is the only item, expect it to be the head and the tail.

Results:
FAIL.
Reason: NullPointerException on TestID 1

Notes:
- TestIDs 2 and 3 pass when TestID 1 is commented out.

Function: insertOrdered

Parameters: int x

	Characteristic	b1	b2	b3
c1	Relation of x to 0	Greater than 0	Less than 0	Equal to 0
c2	Size of list	Equals 0	Equals 1	Greater than 1
c3	x exists in list	true	false	

Test Values

TestID	Relation of x to 0	Size of list	x exists in list
1	8	0	false
2	-19	1	true
3	0	2	true

Narrative:
This is to test inserting an item into the list so that the list is in ascending order.

Expected Results (based on method description/interface):

The item will be inserted into the list so that the list remains in ascending order. Existing items will be offset accordingly. If the inserted item is the only item, expect it to be the head and the tail.

Results:
PASS

Function: deleteHead

Parameters: n/a

	Characteristic	b1	b2	b3
c1	Size of list	Equals 0	Equals 1	Greater than 1
c2	Head exists in list	true	false	

Test Values

TestID	Size of list	Head exists in list
1	0	false
2	1	true
3	2	true

Narrative:
This is to test deleting the beginning of the list.

Expected Results (based on method description/interface):
The item will be removed from the beginning of the list. Existing items be shifted up and the **new** head will be returned.

Results:
FAIL
Reason: NullPointerException on TestID 1

Notes:

- TestID 2 gives NullPointerException when TestID 1 is commented out.
- TestID 3 runs, but has an assertion error when TestIDs 1 and 2 are commented out.
    - Expected the new head, which is 1. Returns deleted head, which is 2

Function: deleteTail

Parameters: n/a

	Characteristic	b1	b2	b3
c1	Size of list	Equals 0	Equals 1	Greater than 1
c2	Tail exists in list	true	false	

Test Values

TestID	Size of list	Tail exists in list
1	0	false
2	1	true
3	2	true

Narrative:
This is to test deleting the end of the list.

Expected Results (based on method description/interface):
The item will be removed from the end of the list. Existing items will not be affected and the **new** tail will be returned.

Results:
FAIL
Reason: NullPointerException on TestID 1

Notes:

- TestID 2 gives NullPointerException when TestID 1 is commented out.
- TestID 3 runs, but has an assertion error when TestIDs 1 and 2 are commented out.
    - Expected the new tail, which is 1. Returns deleted tail, which is 2

Function: delete

Parameters: int x

	Characteristic	b1	b2	b3
c1	Relation of x to 0	Greater than 0	Less than 0	Equal to 0
c2	Size of list	Equals 0	Equals 1	Greater than 1

c3	x exists in list	true	false	

Test Values

TestID	Relation of x to 0	Size of list	x exists in list
1	10	0	false
2	-18	1	true
3	0	2	true

Narrative:
This is to test deleting an item from the list.

Expected Results (based on method description/interface):
The item will be removed from the list. Existing items will adjusted accordingly. The deleted item will be returned.

Results:
FAIL
Reason: NullPointerException on TestID 1

Notes:
- TestID 2 gives NullPointerException when TestID 1 is commented out.
- TestID 3 passes when TestIDs 1 and 2 are commented out.

Function: isEmpty

Parameters: n/a

	Characteristic	b1	b2	b3
c1	Size of list	Equals 0	Equals 1	Greater than 1

Test Values

TestID	Size of list
1	0
2	1
3	2

Narrative:
This is to test whether the isEmpty function accurately determined if the list is empty.

Expected Results (based on method description/interface):
The method will return true if the list is empty

Results:
PASS