Chaehyeon Kim

CSDS 132 HW4 Test Report

Methods in DoubleLinkedList Class

1. **equals**: This method was tested to see if they work with lists that are equal with the same length, equal in length but have different elements, lists of different lengths, and comparing null methods.
2. **append**: This method was tested to see if they work to append different types of lists, such as those containing multiple elements, a single element, or no element. All 6 possible combinations of those lists were tested to see if they successfully append.
3. **Iterator**: This method was tested to check that it iterates through the list correctly. In order to do this, for-loops were used to check that it iterates through the list correctly without falsely going beyond the list.

DoubleLinkedListIterator

1. **hasNext & hasPrevious**: The method hasNext and hasPrevious were tested in the same test method *testHasNextAndHasPrevious* to check if they work correctly for lists containing zero, one, and many elements.
    a. The test for hasNext works correctly (with the tests for hasPrevious commented out), but hasPrevious has errors that cause them to falsely return false when tested to see if hasPrevious returns true (*side note*: it seemed like the code worked correctly when I tested it outside the testing class; might be an error in the testing class that's causing this).
2. **next**: This method was tested to see if the method works for lists containing zero, one, and many elements. For lists containing many and one element, they were also tested for both calling the next method continuously and calling next and previous methods back and forth. By checking beyond where the list has next, it was also tested to see if it correctly returns appropriate exceptions.
3. **previous**: This method was tested to see if the method works for lists containing zero, one, and many elements. For lists containing many and one element, they were also

tested for both calling the *previous* method continuously and calling next and previous methods back and forth. By checking beyond where the list has next, it was also tested to see if it correctly returns appropriate exceptions.

4. **add**: This method was tested on different types of lists (containing 0, 1, and many elements) at different points of lists.
    a. As shown in the test output, the code for this method contains errors and does not add correctly when used at the very end of the lists. It also doesn't work when adding to a list containing only one element. At other points of the lists in different types of lists, it works fine.
5. **set**: This method was tested to see if it works for a list containing 0, 1, and many lists with either next or previous as the last methods called. To check that it correctly returns an exception when trying to set a value to a null list, try-catch statements were used.

Methods in DNA

1. **toString**: This method was tested to see if it is correctly overridden to convert a String containing 0, 1, and many bases to a String.
2. **string2DNA**: This method was tested by inputting 0, 1, and many letters that have corresponding DNA bases (A, G, C, T). It was also tested to see if inputting letters without corresponding DNA bases, such as B and D, correctly throws an IllegalArgumentException.
3. **splice**: This method was tested to see if it works after removing 0, 1, and many bases from the input DNA. It was also tested to see if nothing is appended when the input DNA has fewer than "numbases" number of bases.
4. **overlaps**: This method was tested to see if it works correctly for cases when there is 0, 1, and many overlaps between two input DNA's.
5. **compareTo**: This method was tested to see if it correctly compares this DNA and the input DNA by length or by alphabetical order if both DNA's are the same in length. To test this, this DNA that is shorter than, equal in length, and longer than the input DNA was compared, and each DNA that was tested was also tested with DNA comparing 0, 1, and many bases (except for comparing the DNA's with the same length; DNA's with 0

bases were not compared since they are technically same in value). Also, two DNAs that have the same bases were compared to check if it correctly equals 0.