Student Name: Kaamil Saib

Student Number: SBXKAA001

Overall OOP design

I've created 3 classes 'BST, 'ArrayOps' and 'KbLine'.

BST Class (BST Implementation):

Purpose: Manages the Knowledge Base using a Binary Search Tree structure.

Key Methods:

`loadToBST(String fileName):`

Populates the BST with contents from a text file.

`addOrUpdateTerm(String term, String statement, double cScore)`

Inserts a new term or updates an existing term in the BST.

`searchByTerm(String searchTerm):`

Searches for a term in the BST and returns the corresponding statement.

ArrayOps Class (Traditional Array Implementation):

Purpose: Manages the Knowledge Base using an Array structure.

Key Methods:

`loadToArray(String fileName):`

Populates an Array with contents from a text file.

`updateStatement(String term, String newStatement, String newCScore, String[] kbArray):`

Updates the statement of a term in the array.

`searchByTerm(String searchTerm):`

Searches for a term in the Array and returns the corresponding statement.

KbLine Class:

Purpose: Represents a line of information in the KB.

Key Methods:

`Getters and Setters:`

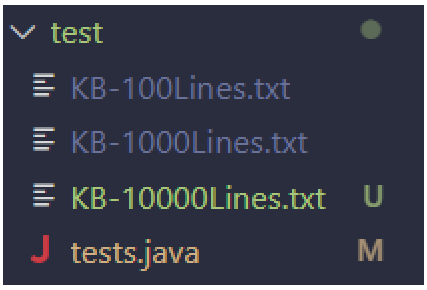
Kbline contains getters to retrieve the term, statement or confidence score and setters to modify terms, statements or confidence scores in the KB.

`compareTo(KbLine other):`

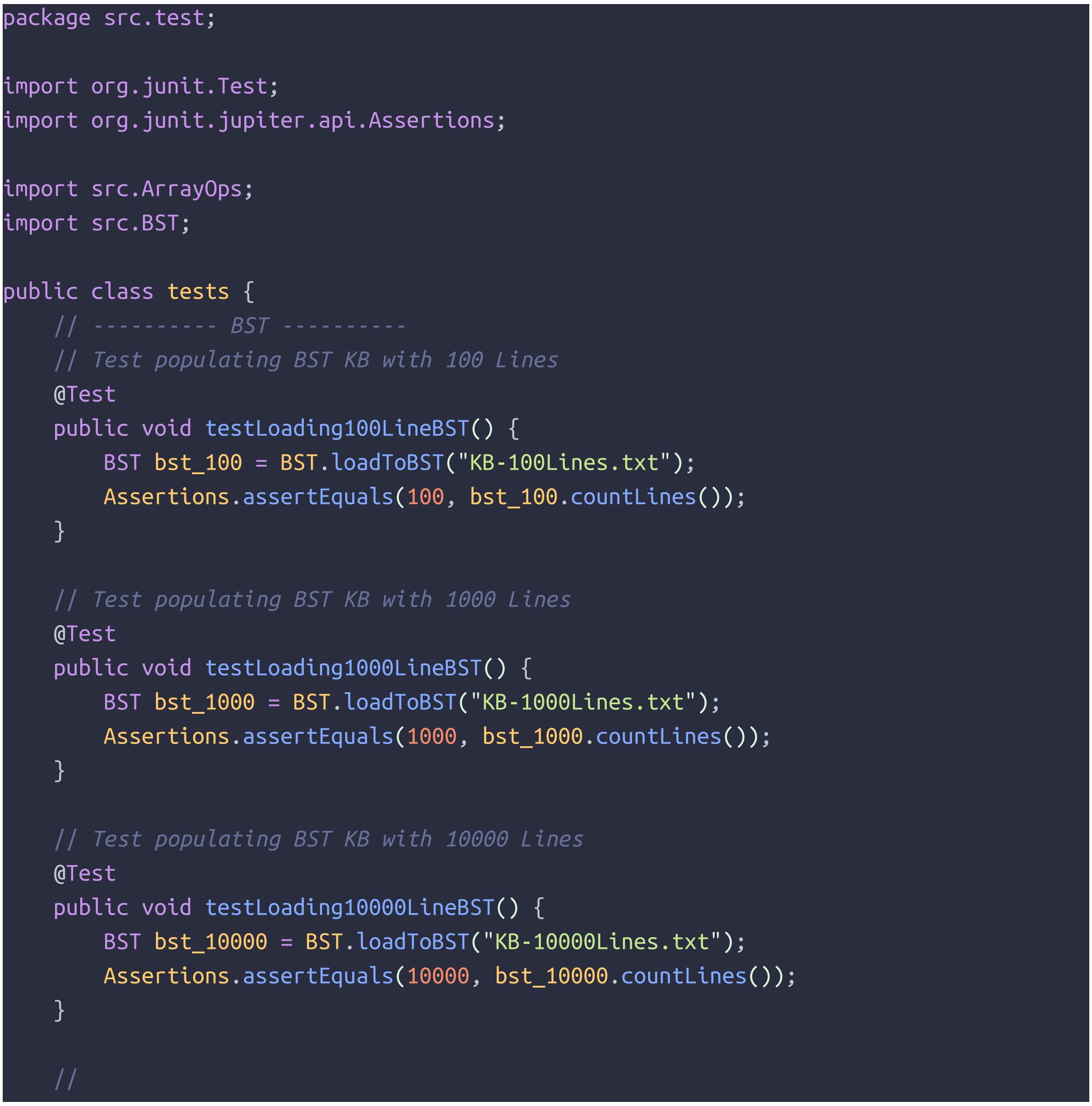
Compares two KbLine objects based on their terms.

Description of the testing protocol

- Implemented unit tests for each key functionality in both ArrayOps and BST classes.
- Tested populating, updating, and searching on 3 different sized KBs.



```
test
  KB-100Lines.txt
  KB-1000Lines.txt
  KB-10000Lines.txt    U
  tests.java           M
```



```java
package src.test;

import org.junit.Test;
import org.junit.jupiter.api.Assertions;

import src.ArrayOps;
import src.BST;

public class tests {
    // ---------- BST ----------
    // Test populating BST KB with 100 Lines
    @Test
    public void testLoading100LineBST() {
        BST bst_100 = BST.loadToBST("KB-100Lines.txt");
        Assertions.assertEquals(100, bst_100.countLines());
    }

    // Test populating BST KB with 1000 Lines
    @Test
    public void testLoading1000LineBST() {
        BST bst_1000 = BST.loadToBST("KB-1000Lines.txt");
        Assertions.assertEquals(1000, bst_1000.countLines());
    }

    // Test populating BST KB with 10000 Lines
    @Test
    public void testLoading10000LineBST() {
        BST bst_10000 = BST.loadToBST("KB-10000Lines.txt");
        Assertions.assertEquals(10000, bst_10000.countLines());
    }

    //
    //
    //
    // ---------- Array ----------
    // Test populating Array KB with 100 Lines
    @Test
    public void testLoading100LineArray() {
        String[] arr_100 = ArrayOps.loadToArray("KB-100Lines.txt");
        Assertions.assertEquals(100, arr_100);
    }

    // -- Test update 100 Lined Array KB
    @Test
    public void testUpdate100LineArray() {
        String[] arr_100 = ArrayOps.loadToArray("KB-100Lines.txt");
        ArrayOps.updateStatment("maple syrup", "Used as a salad dressing", "1.0", arr_100);
        String[] updatedArr = ArrayOps.loadToArray("KB-100Lines.txt");
        Assertions.assertFalse(ArrayOps.searchByTermSen("maple syrup", "Used as a salad dressing",
                updatedArr) == "Term or statement not found.\n");
    }

    // -- Test searching 100 Lined Array KB
    @Test
    public void testSearch100LineArray() {
        // closer Closers are magazines. 1.0
        String[] arr_100 = ArrayOps.loadToArray("KB-100Lines.txt");
        String searchTerm = "closer";
        String result = ArrayOps.searchByTerm(searchTerm, arr_100);
        String expected = "Statement found: Closers are magazines. (Confidence interval: 1.0)";
        Assertions.assertEquals(expected, result);
    }

    // Test populating Array KB with 1000 Lines
    @Test
    public void testLoading1000LineArray() {
        String[] arr_1000 = ArrayOps.loadToArray("KB-1000Lines.txt");
        Assertions.assertEquals(1000, arr_1000.length);
    }

    @Test
    public void testUpdate1000LineArray() {
        String[] arr_1000 = ArrayOps.loadToArray("KB-1000Lines.txt");
        ArrayOps.updateStatment("maple syrup", "Used as a salad dressing", "1.0", arr_1000);
        String[] updatedArr = ArrayOps.loadToArray("KB-1000Lines.txt");
        Assertions.assertFalse(ArrayOps.searchByTermSen("maple syrup", "Used as a salad dressing",
                updatedArr).equals("Term or statement not found.\n"));
    }

    // -- Test searching 1000 Lined Array KB
    @Test
    public void testSearch1000LineArray() {
        String[] arr_1000 = ArrayOps.loadToArray("KB-1000Lines.txt");
        String searchTerm = "closer";
        String result = ArrayOps.searchByTerm(searchTerm, arr_1000);
        String expected = "Statement found: Closers are magazines. (Confidence
interval: 1.0)";
        Assertions.assertEquals(expected, result);
    }

    // Test populating Array KB with 10000 Lines
    @Test
    public void testLoading10000LineArray() {
        String[] arr_10000 = ArrayOps.loadToArray("KB-10000Lines.txt");
        Assertions.assertEquals(10000, arr_10000.length);
    }

    // -- Test update 10000 Lined Array KB
    @Test
    public void testUpdate10000LineArray() {
        String[] arr_10000 = ArrayOps.loadToArray("KB-10000Lines.txt");
        ArrayOps.updateStatment("maple syrup", "Used as a salad dressing", "1.0",
arr_10000);
        String[] updatedArr = ArrayOps.loadToArray("KB-10000Lines.txt");
        Assertions.assertFalse(ArrayOps.searchByTermSen("maple syrup", "Used as a
salad dressing",
                updatedArr).equals("Term or statement not found.\n"));
    }

    // -- Test searching 10000 Lined Array KB
    @Test
    public void testSearch10000LineArray() {
        String[] arr_10000 = ArrayOps.loadToArray("KB-10000Lines.txt");
        System.out.println(arr_10000);

        String searchTerm = "closer";
        String result = ArrayOps.searchByTerm(searchTerm, arr_10000);
        String expected = "Statement found: Closers are magazines. (Confidence
interval: 1.0)";
        Assertions.assertEquals(expected, result);
    }
}
```

Git usage log

```
$ git log | (ln=0; while read l; do echo $ln\: $l; ln=$((ln+1)); done) | (head
-10; echo ...; tail -10)
0: commit 76dc683e95a52180723e76324c38c7d5e9ccdffd
1: Author: Kaamil-Saib <kaamilsaib01@gmail.com>
2: Date: Tue Mar 5 16:07:09 2024 +0200
3:
4: created javadoc and moved .java files into src
5:
6: commit 60c0b8cb0f53d9eaf979390465a4a381e5bc0d18
7: Author: Kaamil-Saib <kaamilsaib01@gmail.com>
8: Date: Tue Mar 5 14:31:28 2024 +0200
9:
...
32: Date: Mon Mar 4 18:49:46 2024 +0200
33:
34: Modified Kbline.java - added compareTo
35: Added BST
36:
37: commit 961612728fa42c987f2ee1264185d3b3f2f7d329
38: Author: Kaamil-Saib <kaamilsaib01@gmail.com>
39: Date: Mon Mar 4 16:37:21 2024 +0200
40:
41: First Commit
```