P09 – DICTIONARY USING BST

Posted on March 22, 2019 by Mouna AYARI BEN HADJ KACEM

- **SUBMIT your completed assignment by 9:59PM on Wednesday, April 17th 2019.** We will accept and grade submissions made through 9:59PM on Thursday, April 18th 2019 (HARD DEADLINE). But, NO CREDIT will be granted for any work that is submitted even one second after this hard deadline.
- **Pair Programming is allowed but not required for this assignment. Register your partnership no later than Monday, April 15th to work with a partner. If you have problems accessing this form, try following the advice here.**

OVERVIEW

This assignment involves the implementation of a dictionary using Binary Search Trees (BST). The dictionary stores a collection of Dictionary words or definitions (words and their meanings). The user can add one or more word definitions to this dictionary. He can also use this dictionary to retrieve the meaning of any word already added. You are going to learn how BSTs can be used to store keys (words in this assignment) to facilitate insertion and retrieval operations to and from a collection of ordered elements, in an efficient way.

OBJECTIVES AND GRADING CRITERIA

The goals of this assignment include:

- Implement common Binary Search Tree (BST) operations,
- Gain more Practice in recursive problem-solving,
- Gain more experience with developing unit tests.

20	Online Tests: these automated grading test results are visible upon uploading your submission. You are allowed multiple opportunities to correct the organization and functionality of your code (if necessary).
20 points	Offline Tests: these automated grading tests are run after the assignment's deadline has passed. They check for similar functionality and organizational correctness as the Online Tests. Since you will not have opportunities to make corrections after seeing the feedback from these tests, you should consider and test the correctness of your own code as thoroughly as possible.
10 points	Code Readability: human graders will review the commenting, style, and organization of your final submission. They will be checking whether it conforms to the requirements of the CS300 Course Style Guide. Since you will not have opportunities to make

corrections after seeing the feedback from these graders, you should consider and review the readability of your own code as thoroughly as possible.

SUBMISSION

You will be submitting a total of five java source files through gradescope.com for this assignment: Dictionary.java, DictionaryWord.java, DictionaryBST.java, DictionaryDriver.java, and DictionaryTests.java. You can make as many submissions as you would like prior to the deadline for this assignment, and the submission marked as "active" is the one that will be used for additional grading after the deadline.

STEP1. GETTING STARTED

Start by creating a new Java Project in eclipse. You have to ensure that your new project uses Java 8, by setting the "Use an execution environment JRE:" drop down setting to "JavaSE-1.8" within the new Java Project dialog box. Then, add a new class called **DictionaryTests** to this project with a public static void main(String[] args) method stub. This class should contain all your test methods for this application. You have to define and implement **at least** FIVE unit test methods. Each of these test methods must be public, static, and return a boolean evaluated to true if an implementation passes the test and false otherwise. You are responsible for designing and writing these tests to help convince yourself that each of your classes is correctly implemented while you are working through this assignment.

STEP2. CREATE DICTIONARY INTERFACE

First, add a new interface called **Dictionary** to this project. This interface models and abstract data type for a dictionary and MUST define exactly the following methods:

```
1  // checks whether the dictionary is empty or not
2  public boolean isEmpty();
3
4  // adds this word definition (word and the provided meaning) to the dictionary
5  // Returns true if the word was successfully added to this dictionary
6  // Returns false if the word was already in the dictionary
7  // Throws IllegalArgumentException if either word or meaning is null or an empty
8  // String
9  public boolean addWord(String word, String meaning);
10
11 // Returns the meaning of the word s if it is present in this dictionary
12 // Throws a NoSuchElementException if the word s was not found in this dictionary
13 public String lookup(String s);
14
15 // Returns the number of words stored in this dictionary
16 public int size();
```

STEP3. CREATE DICTIONARYWORD CLASS

Now, create a new class called **DictionaryWord**. This class is NOT a generic class and it models a binary node of a Binary Search Tree. Each binary node (aka instance of DictionaryWord) defines its data consisting of a pair of word and its meaning, and a link to each child (right and left) of the node. This class must contain the private fields and publicly accessible constructor/methods with the exact names and method signatures as shown in the following. We note that NO additional instance or static fields can be added to this class. Besides, NO additional public or private helper methods should be added to this class.

```
private final String word; // word that represents the search key for this dictionary word
private final String meaning;   // The meaning of the word that this dictionary node defines
private DictionaryWord leftChild;  // The leftChild of the the current WebPageNode
private DictionaryWord rightChild; // The rightChild of the the current WebPageNode

// The following should be the only constructor for this class
// Creates a new dictionary word with the provided word and its meaning pair
// Throws IllegalArgumentException when the word or meaning are either references to an empty
// string or null references. The thrown exception should include a significant error message
// describing which of these problems was encountred.
public DictionaryWord(String word, String meaning) { }


// Getter for the left child of this dictionary word
public DictionaryWord getLeftChild() { }

// Setter for the left child of this dictionary word
  public void setLeftChild(DictionaryWord leftChild) { }

// Getter for the right child of this dictionary word
public DictionaryWord getRightChild() { }

// Getter for the right child of this dictionary word
public void setRightChild(DictionaryWord rightChild) { }

// Getter for the word of this dictionary word
public String getWord() { }

// Getter for the meaning of the word of this dictionary word
public String getMeaning() { }

// Returns a String representation of this DictionaryWord.
// This String should be formatted as follows. "<word>: <meaning>"
// For instance, for a dictionaryWord that has the String "Awesome"
// for the instance field word and the String "adj. Inspiring awe; dreaded."
// as value for meaning field, the String representing that dictionaryWord is
// "Awesome: adj. Inspiring awe; dreaded."
public String toString() { }
```

STEP4. CREATE DICTIONARYBST CLASS

Now, add a new class called **DictionaryBST** to your project. This class is not generic and it MUST implement the interface Dictionary defined earlier in this assignment, as a Binary Search Tree of DictionaryWord nodes. This class SHOULD contain only one private instance field called **root** of type DictionaryWord and it represents the root of this BST. No additional fields either instance or static can be added to this class.

In addition to the four instance public methods defined in the Dictionary interface, this class should define and implement the following instance public and private static methods with exactly the following signatures.

- Note that **duplicate** words are NOT allowed. addWord() method defined in the Dictionary interface MUST return false if is called with word input parameter that has a match in the stored dictionary words.
- We note also that the private **helper** methods defined in the following MUST be recursive. Each public method should make call to the recursive helper method with the corresponding name to operate. No additional fields or public methods (either instance or static) should be added to this class.
- It is also worth to highlight that all the comparisons that may be made in this assignments to compare Strings (either words or their corresponding meanings) should be CASE INSENSITIVE.
- The height() method returns the height of the dictionaryBST counting the number of nodes and NOT the number of edges from the root to the deepest leaf.

```
  // This should be the only constructor of this class.
  // Creates an empty dictionaryBST.
```

```
3   public DictionaryBST() { }
4
5   // Methods defined in the Dictionary interface
6   public boolean isEmpty() { }
7   public boolean addWord(String word, String meaning) { }
8   public String lookup(String s) { }
9   public int size(){ }
10
11  // Public methods not defined in the Dictionary interface
12   /**
13   * Computes and returns the height of this dictionaryBST, as the number of nodes
14   * from root to the deepest leaf DictionaryWord node.
15   *
16   * @return the height of this Binary Search Tree counting the number of DictionaryWord nodes
17   */
18  public int height() { }
19
20  /**
21   * Returns all the words within this dictionary sorted from A to Z
22   *
23   * @return an ArrayList that contains all the words within this dictionary sorted in
24   *         the ascendant order
25   */
26  public ArrayList<String> getAllWords() { }
27
28  // Recursive private helper methods
29  // Each public method should make call to the recursive helper method with the
30  // corresponding name
31
32  /**
33   * Recursive helper method to add newWord in the subtree rooted at node
34   *
35   * @param newWordNode a new DictionaryWord to be added to this dictionaryBST
36   * @param current the current DictionaryWord that is the root of the subtree where
37   *        newWord will be inserted
38   * @return true if the newWordNode is successfully added to this dictionary, false otherwise
39   */
40  private static boolean addWordHelper(DictionaryWord newWordNode, DictionaryWord current) { }
41
42
43  /**
44   * Recursive helper method to lookup a word s in the subtree rooted at current
45   *
46   * @param s String that represents a word
47   * @param current pointer to the current DictionaryWord within this dictionary
48   * @return the meaning of the word s if it is present in this dictionary
49   * @throws NoSuchElementException if s is not found in this dictionary
50   */
51  private static String lookupHelper(String s, DictionaryWord current) { }
52
53
54  /**
55   * Recursive helper method that returns the number of dictionary words stored in
56   * the subtree rooted at current
57   *
58   * @param current current DictionaryWord within this dictionaryBST
59   * @return the size of the subtree rooted at current
60   */
61  private static int sizeHelper(DictionaryWord current) { }
62
63
64  /**
65   * Recursive helper method that computes the height of the subtree rooted at current
66   *
67   * @param current pointer to the current DictionaryWord within this DictionaryBST
68   * @return height of the subtree rooted at current counting the number of
69   * DictionaryWord nodes from the current node to the deepest leaf in the subtree
70   * rooted at current
71   */
72  private static int heightHelper(DictionaryWord current) { }
73
74
75
76  /**
77   * Recursive Helper method that returns a list of all the words stored in
78   * the subtree rooted at current sorted alphabetically from A to Z
79   *
80   * @param current pointer to the current DictionaryWord within this dictionaryBST
81   * @return an ArrayList of all the words stored in the subtree rooted at current
82   */
83  private static ArrayList<String> getAllWordsHelper(DictionaryWord current) { }
```

You are responsible for testing thoroughly your implementation of the DictionaryBST behaviors before moving on to the next step. Recall that you have to define and implement **at least** FIVE unit test methods in your DictionaryTests class. Each of these test methods must be public, static, and return a boolean evaluated to true if an implementation passes the test and false otherwise. Among other you have to implement tests methods to assess the correctness of each method defined in the Dictionary interface and implemented in your DictionaryBST class.

STEP5. DRIVER APPLICATION

The final step in this assignment is to develop a driver application to make use of your Dictionary implemented using BST. To do so, create a new class called **DictionaryDriver** with a public static void main(String[] args) method stub and add it to your project source folder. Then, implement the main() method of that new class, that serves as a driver of this application. Organize this functionality into whatever custom private static methods you see fit. But, make sure that running the main method within your DictionaryDriver class results in an interaction section comparable to the sample shown at the end of this section. Any new variables that you create for this driver must be local within some static method. No instance or static fields should be defined in the DictionaryDriver class. We provide you in the following with some specific requirements for how the interactive session of the driver application should proceed:

- This driver should continuously prompt the user to enter one command at a time, and process each of them until the user enters a 'Q' command to quit the program. For each of the command, you basically need to call the corresponding method on a single DictionaryBST object.
- The commands should be case INSENSITIVE and are as follows:

Command	Corresponding Method	Description
A <word> <meaning>	addWord(word, meaning)	"A" adds a dictionaryWord in the dictionary. It prints *"WARNING: failed to add duplicate word: <word>."* for duplicate entry
L <word>	lookup(word)	"L" searches the definition of the word in the dictionary. It prints *"No definition found for the word <word>"* when a word doesn't exist in the dictionary or *"There are no definitions in this empty dictionary."* when the dictionary is empty.
G	getAllWords()	"G" gets all the words in the dictionary in sorted order, and then prints those words on a single line separated by spaces. Prints "*Dictionary is empty."* when the dictionary has no words.
S	size()	"S" gets the count of words stored in the dictionary.
H	height()	"H" gets the height of the binary search tree dictionary counting number of nodes from the root of the BST to the deepest leaf.
Q	–	"Q" quits this program

- Your program should support commands in both lowercase and uppercase.
- All the arguments within a user command line are of type String.
- If the user enters a recognized command, but with a wrong number of followed arguments, a warning message starting by "*WARNING: Syntax Error*" must be displayed, and then re-prompt the user to enter another command.
- An 'A' (addWord) command option should not be followed by less than two arguments. The first argument will be the **word** (a single word). The second argument (the **meaning**) will begin after the first argument and will continue until the end of the input line.
- When the user enters a command option not listed above, the following warning message should be displayed: *"WARNING: Unrecognized command."*, and then re-prompt the user to enter another command.
- Whenever you show the user a warning message, you should not perform the corresponding operation on the dictionary, nor should you exit the program. Instead, you should re-prompt the user, and continue the program until the user enters a 'q'.
- DO NOT call System.exit() method to quit the program.
- Here is an interactive log that demonstrates a use of the complete application.

```
========================== Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
==================================================================

Please enter your command: g
Dictionary is empty.

========================== Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
==================================================================

Please enter your command: s
0

========================== Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
==================================================================

Please enter your command: h
0

========================== Dictionary ============================
Enter one of the following options:
```

```
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
========================================================================

Please enter your command: a exceptional adj. forming an exception, unusual, outstanding

============================ Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
========================================================================

Please enter your command: A Gigantic adj. Huge, giant-like.

============================ Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
========================================================================

Please enter your command: a Allocate  v. assign or devote to (a purpose, person, or pla

============================ Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
========================================================================

Please enter your command: a Sofa  n. Long upholstered seat with a back and arms.

============================ Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
========================================================================

Please enter your command: L allocate
allocate: v. assign or devote to (a purpose, person, or place).

============================ Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
```

```
[Q] to quit the program
========================================================================

Please enter your command: l famous
No definition found for the word famous

============================ Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
========================================================================

Please enter your command: a famous
WARNING: Syntax Error for [A <word> <meaning>] command line.

============================ Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
========================================================================

Please enter your command: a famous adj. having a widespread reputation, usually of a fa

============================ Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
========================================================================

Please enter your command: g
Allocate, exceptional, famous, Gigantic, Sofa

============================ Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
========================================================================

Please enter your command: s
5

============================ Dictionary ============================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
========================================================================
```

```
Please enter your command: h
3

========================== Dictionary ===========================
Enter one of the following options:
[A <word> <meaning>] to add a new word and its definition in the dictionary
[L <word>] to search a word in the dictionary and display its definition
[G] to print all the words in the dictionary in sorted order
[S] to get the count of all words in the dictionary
[H] to get the height of this dictionary implemented as a binary search tree
[Q] to quit the program
=================================================================

Please enter your command: q
============================== END ==================================
```

The following methods may be useful while completing this assignment:

- **String.compareTo(String anotherString):** Compares two strings lexicographically.
- **String.compareToIgnoreCase(String str):** Compares two strings lexicographically, ignoring case differences.
- **String.equals(Object anObject):** Compares this string to the specified object.
- **String.equalsIgnoreCase(String anotherString):** Compares this String to another String, ignoring case considerations.
- **String.split(String regex, int limit):** Splits this string around matches of the given regular expression.
- **ArrayList.addAll(Collection<? extends E> c):** Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator.

SUBMISSION

Congratulations on finishing this CS300 assignment! After verifying that your work is correct, and written clearly in a style that is consistent with the course style guide, you should submit your final work through gradescope.com. The FIVE files that you must submit include: Dictionary.java, DictionaryWord.java, DictionaryBST.java, DictionaryDriver.java, and DictionaryTests.java. Your score for this assignment will be based on your "active" submission made prior to the hard deadline of **9:59PM on Thursday, April 18th**.The second portion of your grade for this assignment will be determined by running that same submission against additional offline automated grading tests after the submission deadline. Finally, the third portion of your grade for your P09 submission will be determined by humans looking for organization, clarity, commenting, and adherence to the course style guide.

EXTRA CHALLENGES

Here are some suggestions for interesting ways to extend this simulation, after you have completed, backed up, and submitted the graded portion of this assignment. **No extra credit will be awarded for implementing these features**, but they should provide you with some valuable practice and experience. DO NOT submit such extensions using gradescope.

- Try to add “Load” option <load filename.txt> to load and add a set of dictionary words (words and their meanings) to your Dictionary from a textfile. Oxford_English_Dictionary.txt is an example of file containing a set of word definitions from Oxford_English_Dictionary. Each line started with a single word followed by its definition.
- Try to add “Save” option <save filename.txt> to save the collection of words and their meanings in a textfile according to the specific format you used to load the file.
- Try to add an option of your choice to get the set of words that start with a given prefix, and then print them in a single line separated by single spaces.
- Try to implement the Dictionary interface using a Singly linked list class, that you can name for instance DictionaryLL. Then, compare the running time complexity of the different methods defined in this interface while implemented using a BST versus using a Linked List.