Department of Electrical and Computer Engineering
ECSE 202 – Introduction to Software Development
Assignment 2
Stacks, B-Trees, Sorting Data

Due February 19, 2018 at 5:00 pm

Problem Description

In this assignment we will explore some of the data structures discussed in class, namely the Stack and B-Tree classes. You will use these in conjunction with a file reading program to sort a list of names in ascending and descending order by creating a B-Tree representation of the file and the performing an inorder traversal to sort the data. Finally, a stack will be used to reverse the order of the list for final display.

Here is an example run of the program you are to write, given a short file of names:

```
Assignment 2 - File Sorting Program
Enter name of file to sort: Names-short.txt


File in sort order:

Benes Dorethea
Britto Teisha
Freeze Clarisa
Galentine Dante
Woolery Ciera


File in reverse sort order:

Woolery Ciera
Galentine Dante
Freeze Clarisa
Britto Teisha
Benes Dorethea


Program terminated
```

To make this assignment tractable, you are provided with a listFile class that opens a text file for reading and displays it on the output device, line by line. The constructs used in this class are beyond where we are currently in the course, but you should have a sufficient knowledge of Java to figure out how to modify the code for Assignment 2.

The file containing the code, listFile.java, is available on myCourses and reproduced below:

```
package listFile;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

/**
 * This program reads a specified text file and echos it to standard
 * display.  You can use this as the basis of Assignment 2, adding the
 * necessary components to implement the program specifications.
 * @author ferrie
 *
 */

public class listFile {
    @SuppressWarnings("resource")
    public static void main(String args[]) {

// Prompt user for a file name.  If no name is entered, terminate
// the program, otherwise attempt to open the file. If file open
// is not successful, prompt again for a new name.  Keep doing this
// until successful open, or a blank line is entered.

      System.out.println("Simple File Listing Program");
      Scanner sc = new Scanner(System.in);
      BufferedReader rd = null;

      while(rd == null) {
           System.out.print("Enter name of file to list: ");
           String filename = sc.nextLine();
           if (filename.equals("")) {
                System.out.println("Program terminated");
                System.exit(0);                         // Exit
           }
// Try to open the specified file
           try {
                rd = new BufferedReader(new FileReader(filename));
           }
           catch (IOException ex) {
                System.out.println("Unable to open file, try again.");
```

```
            }
        }

    // Read the file a line at a time into a string.  Print as read
    // to the output display.  Modify the code below as necessary.

        System.out.println("");
        try {
            while (true) {
                String line = rd.readLine(); // Read a line of text
                if (line == null) break;     // Exit if at end of file
                System.out.println(line);    // Print (or do whatever)
            }                                // to current line
        }
        catch (IOException ex) {
            System.out.println("I/O Error - program terminated");
            System.exit(-1);
        }

        System.out.println("\n\nProgram terminated");

      }
    }
```

Pay particular attention to the last while loop in the program. Here you have access to each line of text in a file as a string. You should be able how to save each string in a B-Tree for further processing.

Strategy

You have two essential problems to solve: i) sort the file and ii) print the file sorted in both ascending and descending order. As we have seen in class, sorting can be done easily by creating a B-Tree and then performing an inorder traversal to visit each node in sort order. This means that you need to create a class, bTree, with addNode and traversal methods (following the example in the lecture slides). The bNode class, which holds the data, must now hold a String instead of an int for the payload. This complicates the addNode method somewhat, as you now need a method to compare two strings (hint: google java string compare method). It is also a good idea to use the version of this method that is *case insensitive*, i.e., where smith john is equal to SMITH JOHN or Smith John. From here it should not be too difficult to figure out how write an inorder traversal method which prints out the nodes (lines of text) in ascending order.

The second component, is to display the file again, but in reverse sort order. You are not allowed to use any of the Java array types to do this. Instead, create a Stack class where the listNode class is modified to hold a string instead of an integer. Since a stack is accessed in last-in, first-out order, you can obtain the desired effect by creating a version of your inorder traversal that substitutes a PUSH for printing the value at the current node. Later, once you have output the list in ascending order, you can perform another while loop that pops the stack, displaying each node in desscending order.

Structure

Your program should containing the following 5 classes (use the same names):

1. A2,java	The main program derived from listFile
2. bNode.java	The B-Tree node class
3. bTree.java	The B-Tree class containing the addNode method and traversal methods
4. listNode.java	The listNode class
5. Stack.java	The Stack class containing push and pop methods

Approach

This is a relatively straightforward program, but definitely not trivial (especially if this is your first course in computer programming). The easiest way to proceed is to develop and test each class independently before attempting to write the overall program.

Instructions

Test your program using the Names.txt file. When inputting the file name, you must specify the complete name including path, e.g., \Users\ferrie\assignments\A2\Names.txt, unless the file resides in the *default directory*. If your project lives at \Users\ferrie\workspace\A2, then if you copy the Names.txt file to that location, then it suffices to type *Names.txt* when prompted for the file name.

If your program works correctly, it should produce a display similar to that shown on Page 1, with the exception of printing all the entries in Names.txt. Save this output by copying and pasting into a text file, e.g. into Notepad if you are using Windows, for example. Save this file as A2.txt.

Upload A2.txt along with all of your source (.java) files to myCourses as indicated.

About Coding Assignments

We encourage students to work together and exchange ideas. However, when it comes to finally sitting down to write your code, this must be done *independently*. Detecting software plagiarism is pretty much automated these days with systems such as MOSS.

https://www.quora.com/How-does-MOSS-Measure-Of-Software-Similarity-Stanford-detect-plagiarism

Please make sure your work is your own. If you are having trouble, the Faculty provides a free tutoring service to help you along. You can also contact the course instructor or the tutor during office hours. There are also numerous online resources – Google is your friend. The point isn't simply to get the assignment out of the way, but to actually learn something in doing.

fpf/February 4, 2018