Assignment 4

Due date

- noon on Nov ~~20th~~ ~~22nd~~ 23rd.

Submit your code as per the provided instructions. A signup sheet will be provided to you during class to setup an appointment with the TA to provide a demo of your project.

Updates

- Assignment posted: Tue Nov 8 09:40:28 EST 2016

Assignment Goal

"wc" command with Visitor, Prototype, and Observer

- You are required to use ANT for the following:

- Compiling the code
- running the code
- Generating a tarball for submission

- Your code should compile and run on *remote.cs.binghamton.edu* or *bingsuns* or the *debian-pods* in the Computer Science lab in the Engineering Building.

Project Description

Text Processing Visitors

- Design and develop the following 2 visitors:
  - A visitor (Populate Visitor) that reads an input file, input.txt, and populates a data structure with all the words in the file. Use a data structure that will maximize performance for the second visitor. Store the words in the data structure using the insert criteria for that data structure.
  - A second visitor (Word Count) that counts the number of words, number of distinct words, and number of characters (NOT distinct characters) stored by the data structure, and then stores it in a file named output.txt. Total number of characters should not take whitespaces into account. Also,
  - If the input has the following words: *the the the quick brown fox jumps jumps over the lazy dog* , then the number of distinct words is 8, as the rest are repeated words.
  - A third visitor (Clone-and-Observe-Visitor) that clones the data structure so that it can be used as a backup. You should setup the observer pattern's subject and listener relationship as part of the implementation of this visitor.
  - Use the observer pattern, so that whenever any word is modified in the original data structure, the corresponding element (or whatever is used to store the word) of the backup data structure is notified and updated. You will be graded on how fine grained you can keep the updates. So, do not update the entire data structure, each time.
  - The second visitor is supposed to be specialized in extracting information such as word count, etc. from the data structure. The first visitor should not do any work for the second visitor. So, you CANNOT use a separate data structure, outside the original data structure, to keep track of the word count. It is also NOT acceptable to keep a running count while populating the original data structure. So, your performance should depend on the choice of data structure, choice of data structure within each node/element to store information, and traversal through the data structure to calculate the total number of words, distinct words, and characters.
- In the driver code, call the two visitors and test the performance in the following manner:

```
long startTime = System.currentTimeMillis();

  Start of loop N times
     declare/instantiate the data structure and visitors
     code to visit with the PopulateVisitor
     code to visit with the WordCountVisitor.
  End of loop N times

// the file should be open closed within the code in the loop
// So, each iteration should overwrite the file

long finishTime = System.currentTimeMillis();
Calculate total_time as (finishTime-startTime)/NUM_ITERATIONS.
Write the total_time value to stdout
```

- In the driver, after the timing loop, call a method using a test class, to show that the 3rd visitor works correctly.
  NEW One way to do this it write another visitor that updates the integer value in all the nodes of the original tree. The updates should automatically notify the backup nodes via the observer pattern. Next, write to two different files the int values in the nodes using an inorder traversal of the trees. During the demo, do a "diff" of the two files to prove that your observer pattern worked.
- In the driver, after the timing loop, call a method from a test class, to show that the observer pattern for the backup works correctly.
- Use any whitespace as the criteria to delimit words.
- Assume that the input file will NOT contain any special characters.
- The following should be read from command line: input file name, output file name, and the value of NUM_ITERATIONS.
- Use the Logger class from any previous assignment along with your own debug scheme.

Sample Inputs

- First sample
  - Input and Output. [Thanks, Aamir]. [Received feedback that this input file has special characters.]
  - Input and Output. [Thanks, Hardik]. [Received feedback that this input file has special characters.]

- Input and Output. [Thanks, Saurabh].
- Another Input
  - Input and Output. [Thanks, Purva].
  - Input and Output. [Thanks, Parshant].

Design Requirements

- Same as previous assignments, except that javadoc is now optional.

Code Organization

- Your directory structure should be the following:

```
lastName_firstName_assign4
  ---wordCount
    ----- README.txt
    ----- build.xml
    ----- src
       -----wordCount
          ----------driver
                          ----------Driver

                                   (1) Validate that command line inputs for input and output file names.
                                   (2) Create the element(s)
                                   (3) Create the two visitor instances
                                   (4) Use the performance measurement loop given above.


          ----------util
                     ----------FileProcessor
                         [methods to read a line and write to a file]
          ----------dsForStrings
                     ----------...
                         [Classes for data structures, if you define your own]
          ----------visitors
                     ----------PopulateVisitor
                         [Visitor class that reads words from a file and populates a data structure]
                     ----------WordCountVisitor
                         [Visitor that determines the total number of words, total number of unique words, and characters stored in the data structu
                     ----------DSProcessingVisitorI
                         [visitor interface]
          ----------other packages that you need
```

Code Templates

- None provided for this assignment.

Submission

- Same as Assignment-1.

Late Submissions

- The policy for late submissions is that you will lose 10% of the grade for each day that your submission is delayed.

Grading Guidelines

Grading Guidelines.

mgovinda at cs dot binghamton dot edu
Back to Design Patterns