ITCS 2214-002 Program 0.1 (Review) Spring 2016

Due Saturday January 23rd, 2016 at 11:55PM

This assignment is primarily designed to introduce timing operations and review the following programming concepts: reading from a file and parsing Strings. Indirect goals include practicing standard code organization, getting a feel for lab sessions for this course, and simply warming up programming skills!

Overview

The primary goals of this program are as follows:

(1) Compute the frequency of character occurrence in the most-used words in the English language (given in 'words.txt')
(2) Create a Stopwatch class to get a basic idea of how to time sections of code

You will create three classes: **StopWatch**, **CharacterCount**, and **Driver**.

StopWatch will use 'System.nanoTime()' to implement the given '**Timer**' interface. Stopwatch must give functionality to start and stop the timer, and to return the duration of the most recent segment of time.

CharacterCount will use an Array to store the occurrence of each character of the English alphabet. This class must allow users to add a new word to the structure, parsing each given String into its component characters and updating the count for each one. It must also include a method to print the overall occurrences for each letter.

Driver will create instances of the StopWatch and CharacterCount classes, and will time how long it takes to read the given data file line by line, updating the character count with each successive word.

Read the following section carefully to ensure you address every requirement. See rubric for details.

Tasks

- Create a new project/package in the IDE of your choice and name it **program0.1**
- Download 'words.txt' from Moodle and **place it in your project's root directory**

Driver Setup:

- Create a new class named 'Driver' with a main method OR copy the skeleton driver given below
- Create a method to read the given data file. If you are comfortable with a different approach to reading files, feel free to ignore these steps.
    - Use BufferedReader and FileReader objects to open the file
    - Read one line at a time using BufferedReader's 'readline' method
    - For now, you can simply test this method by printing each line to the console
    - Make sure you close your reader object after your method finishes reading the data file!

StopWatch:

- Copy the 'Timer' interface given below into a new file in your project.
- Create a new class named 'StopWatch' which implements the 'Timer' interface
- StopWatch must use 'System.nanoTime()' to compute the start time and duration. This method returns **long** type values, so create **start, end,** and **duration** fields of this type. These fields should be private.
- Implement the methods described in the Timer interface.
- You can test your StopWatch class by creating an instance of it in your driver and calling all its methods from there.

CharacterCount:

- Create a new class named 'CharacterCount.'
- At minimum, this class should contain the following fields:
    - **An Array** to hold a count for each character of the English alphabet
    - **A count** of the number of words passed to it
- At minimum, this class should contain the following methods:
    - A constructor to initialize the class' fields
    - **addWord** to add a new word. This method should examine each character in the given word and update the appropriate indices of the character count array. It should also update the word count. Be careful about case sensitivity.
    - **printLetterCount** to print the final count for each letter
    - **printWordCount** to print the word count

Driver Details

Now that you've implemented the StopWatch and CharacterCount classes, create instances of both objects in your driver.

- In the method where you read from the data file, pass each word to the CharacterCount's **addWord** method
- Use the StopWatch to calculate the amount of time it takes to process the entire data file. Be careful where you place the 'start' and 'stop' method calls; you only want to time how long it takes to read all the words and send them to the CharacterCount object, you don't want to time variable initializations and other unrelated method calls.

Deliverables

- Your program should print the following values each time it runs**:**
    - The number of words processed by the CharacterCount class
    - The occurrences of each character in the CharacterCount class
    - The amount of time (in nanoseconds) taken to process the data
- Try timing your program with different numbers of words from the file, stopping at word counts of 100, 500, 1000, 5000, etc. How does the size of the data affect the running time of the program?
- **Create a graph** plotting the word count against the running time for at least 5 different counts. (100, 500, 1000, 5,000, 10,000). For each number of words, you will need to run the program a few times (at least 3) and find the average of the timing results, as they can differ a great deal. You can simply re-run the program enough times to obtain these results, or you can set up your program to compute the results for you.
- **Bonus points**: create a bar graph of character occurrences (from all 10,000 words)

‘Timer’ Interface

```
/*
 *        Interface 'Timer' describes the operations necessary to support a StopWatch class
 *        A StopWatch class should use System.nanoTime() to determine the amount of time between calls to 'start' and
'stop'
 *
 */
public interface Timer {

          /*
           * Start the timer. Set the start time using System.nanoTime()
           */
          public void start();

          /*
           * Stop the timer. Calculate the duration using the start time and System.nanoTime()
           */
          public void stop();

          /*
           * Get the elapsed time from the timer. This value will be given in nanoseconds.
           */
          public long getElapsedTime();
}
```

‘Driver’ Skeleton (optional)

```
/*
 * Skeleton Driver for Program 0.1
 *
 */
public class Driver {

          /*
           * Main method: point of entry
           */
          public static void main(String[] args) {

                    // StopWatch object
                    // CharacterCount object
                    // Other fields
                    // ..

          }

          /*
           * Read the given data file. Each line contains a word.
           * Pass each word to a CharacterCount object to increment the appropriate characters
           * Print out the relevant information
           */
          public static void readFromFile(  ) {

                    // variable declarations
                    // ..

                    try{
                              // open data file

                              // start timer
                              //        read each line from data file
                              //        pass each word to CharacterCount
                              // end timer

                    } catch (  ) {
                              // error!
                    } finally {
                              // close file reader
                    }

                    // print word count
                    // print character occurrences
                    // print elapsed time
          }
}
```

Scoring Rubric

StopWatch: 20 points

- Up to 5 points for appropriate documentation and comments
- Up to 15 points for correctly implementing the interface's methods
    - **start, stop, getElapsedTime**

CharacterCount: 20 points

- Up to 5 points for appropriate documentation and comments
- Up to 15 points for correctly implementing the methods
    - **addWord, printLetterCount, printWordCount**

Driver: 15 points

- Up to 5 points for appropriate documentation and comments
- Up to 5 points for correctly reading from the data file
- Up to 5 points for correctly utilizing the CharacterCount and StopWatch objects

Graph: 15 points

- Up to 10 points for graphing the timing results. Your graph must contain results from at least 5 different word counts (100, 500, 1000, 5,000, 10,000).
- Up to 5 BONUS points for creating a bar graph of the character occurrences from all 10,000 words in the data file

Total points available: 70

The final score will be calculated out of 60, so it is possible to miss a few points without penalty.

Late programs turned in prior to class on Tuesday (January 26th) will receive a 10 point penalty.

Programs turned in after class on Tuesday (January 26th) will receive at most 50% of the available points.