Daniel C. Landon Jr.

Program # 1

Read The Book

01.22.2020

***Abstract***

Given the file “Oliver.txt” we were required to process the file for specific information;

* Total number of words in document (this is alpha characters only, no special characters or numbers.)
* Total line count
* Average number of words per line
* Longest word found in text
* Time to execute

Created a single class to contain all code needed for processing the book. I started with a single-entry point as I saw no reason for interaction with the user once processing started other than echo results. I modified the main method to check args[0] for value. This gives the user the option to supply a text file for a different book, otherwise the app defaulted to “oliver.txt” for processing if nothing supplied.

The following algorithm was used for execution;

* Main will pass file to process.
* Enclose all of the following in a try/catch.
* Capture start time
* Open file
* Loop till EOF (End Of File)
  + Read in single line
  + Strip non-alpha characters from sentence
  + Parse sentence into separate words
  + Find the longest word (For this I went with the first word in the book that had the most characters. Any word that came after that had the same number of characters was ignored.)
  + Find wordcount (I had issues here with regEx. If I stripped everything and replaced it with a blank space some replaced characters would register as “blank” words. I am certain it has something to do with my lack of skill using regEX. Instead I opted for searching for “empty” words and skipping them. I will revisit this later to clean this up as I feel it is to much of a monkey with a hammer approach.)
  + Find line count
  + Echo out every 100,000th line.
* Close file
* Console log all necessary output to show results
* Calculate processing time and display
* PROGRAM TERMINATED – END OF LINE

***CONSOLE OUTPUT***

Starting Book Processing ...

START TIME: 2020-01-25T06:06:27.064Z

Every 100,000th line: the United Netherlands,' and his 'Life of John of Barneveld,' had

Number of lines read: 101168

Number of words read: 999541

Average number of words per line: 9

The FIRST word found with the most characters was 'POLYPHYSIOPHILOSOPHIQUES', it is 24 characters long.

END TIME: 2020-01-25T06:06:27.605Z

Time for completion (milliseconds): 541

SUCCESS: Book Processed.

***App.java***

package app;

/\*\*

\* <h1>App</h1>

\*

\* <h2>Notes:</h2>

\*

\* <h3>Entry point for application. If a command line argument is supplied then that file name will be used to process otherwise defaults to "oliver.txt" per class requirements.</h3>

\*

\* <p><strong>author:</strong> <em>Daniel C. Landon Jr.</em></p>

\* <p><strong>instructor:</strong> <em>Dr. Bob Walsh</em></p>

\* <p><strong>class:</strong> <em>CSCI 202 - Introduction to Software Systems</em></p>

\* <p><strong>date:</strong> <em>01.22.2020</em></p>

\*

\* @author Daniel C. Landon Jr.

\* @version 0.1

\*/

public class App {

/\*\*

\* <h1>main</h1>

\*

\* <p><strong><em>Notes:</em></strong> Nothing special here, entry point</p>

\*

\* @custom.precondition if a custom file is not supplied in the command line then the "oliver.txt" file must be available in the same directory as the application, this is the default text file.

\*

\* @custom.postcondition successful execution of program

\*

\* @param args argument list supplied through command prompt

\* @throws Exception any errors

\*/

public static void main(String[] args) throws Exception {

// variables

ProcessTheBook \_theBook = new ProcessTheBook();

String \_processMessage = "";

// check to see if args is empty

if(args.length == 0) {

// no command line arguments supplied so use default

if(\_theBook.StartProcessing("oliver.txt")) {

\_processMessage = "SUCCESS: Book Processed.";

} // end if

else {

\_processMessage = "ERROR: Book Not Processed!";

} // end else

} // end if

else {

// command line argument supplied, use the value

if(\_theBook.StartProcessing(args[0])) {

\_processMessage = "SUCCESS: Book Processed.";

} // end if

else {

\_processMessage = "ERROR: Book Not Processed!";

} // end else

} // end else

System.out.println("\n" + \_processMessage);

} // end main

} // end class

***ProcessTheBook.java***

package app;

import java.io.File;

import java.time.Duration;

import java.time.Instant;

import java.util.Scanner;

/\*\*

\* <h1>ProcessTheBook</h1>

\*

\* <p><strong><em>Notes:</em></strong> This class does not have a constructor. The class will take a text file and process it for word and line count and display relevant information once completed.</p>

\*

\* <p><strong>author:</strong> <em>Daniel C. Landon Jr.</em></p>

\* <p><strong>instructor:</strong> <em>Dr. Bob Walsh</em></p>

\* <p><strong>class:</strong> <em>CSCI 202 - Introduction to Software Systems</em></p>

\* <p><strong>date:</strong> <em>01.22.2020</em></p>

\*

\* @author Daniel C. Landon Jr.

\* @version 0.1

\*/

public class ProcessTheBook {

/\*\*

\* <h1>StartProcessing</h1>

\*

\* <p><strong><em>Notes:</em></strong> This is the start point for processing the book.</p>

\*

\* @custom.precondition text flie must be supplied for processing

\*

\* @custom.postcondition application will process book successfully

\*

\* @param \_bookToProcess string indicating the name of the file containg the book to process.

\* @return true if successful, false if there was a problem

\*/

public boolean StartProcessing(String \_bookToProcess) {

try {

System.out.println("\nStarting Book Processing ...");

Instant \_startTime = Instant.now();

System.out.println("\n START TIME: " + \_startTime);

// open file

File readFile = new File(\_bookToProcess);

// read object

Scanner \_dataInput = new Scanner(readFile);

// variables

int \_lineCounter = 0;

int \_wordCounter = 0;

String \_longestWord = "";

String \_cleanLine = "";

String[] \_parse;

while(\_dataInput.hasNext()){

// read the line

String \_line = \_dataInput.nextLine();

// strinp special characters

\_cleanLine = StripSpecialCharacters(\_line);

// parse the sentence into individual words

\_parse = \_cleanLine.split(" ");

// find the first longest word

\_longestWord = FindLongestWord(\_parse, \_longestWord);

// get the word count for words with length greater than zero

\_wordCounter+= FindWordCount(\_parse);

// number of lines processed

\_lineCounter++;

// echo out every 100,000th line

if(\_lineCounter % 100000 == 0) System.out.println("\nEvery 100,000th line: " + \_line);

} // end while

\_dataInput.close();

// output to console

ConsoleOutput(\_lineCounter, \_wordCounter, \_longestWord);

Instant \_endTime = Instant.now();

Duration \_timeElapsed = Duration.between(\_startTime, \_endTime);

System.out.println("\n END TIME: " + \_endTime);

System.out.println("\nTime for completion (milliseconds): " + \_timeElapsed.toMillis());

} // end try

catch (Exception e) {

// boom the nija strikes

System.out.println("Something Went Wrong");

System.out.println(e.getMessage());

return false; // short circuit and return

} // end catch

return true;

} // end StartProcessing

/\*\*

\* <h1>FindWordCount</h1>

\*

\* <p><strong><em>Notes:</em></strong> Takes and array of words and counts them, but it skips anything that is blank.</p>

\*

\* @custom.precondition an array of words must be supplied

\*

\* @custom.postcondition a count of how many words in the array

\*

\* @param \_wordArray array of words from parsed sentence

\* @return returns the numbers of words found

\*/

protected int FindWordCount(String[] \_wordArray) {

// variables

int \_tempCount = 0;

// loop the array

for(String \_item: \_wordArray){

// if the word length is greater than zero count it.

// i am doing this here. having trouble removing blank words using regex...if i do not do this my word count is off

if(\_item.length() > 0 ) \_tempCount++; // end if

} // end for

// return the count

return \_tempCount;

} // end FindWordCount

/\*\*

\* <h1>StripSpecialCharacters</h1>

\*

\* <p><strong><em>Notes:</em></strong> Takes a line of text and removes all non-alpha characters.</p>

\*

\* @custom.precondition must be supplied a line of text

\*

\* @custom.postcondition line will be modified to where only alpha characters will be available

\*

\* @param \_lineToProcess the line of text to process for special characters

\* @return the line after it has been processed

\*/

protected String StripSpecialCharacters(String \_lineToProcess) {

// remove all special characters from line

\_lineToProcess = \_lineToProcess.replaceAll("[^a-zA-Z]", " ");

// return the processed line

return \_lineToProcess;

} // end StripSpecialCharacters

/\*\*

\* <h1>FindLongestWord</h1>

\*

\* <p><strong><em>Notes:</em></strong> This will sort through the array of words supplied to determine which one is the largest. Once a word is found it will be kept. Any words that come after that are of the same size will be ignored and the first word found will be kept.</p>

\*

\* @custom.precondition an array of words, with no special characters must be supplied as well as a variable containing the current largest word

\*

\* @custom.postcondition a new variable will be returned showing the current largest word

\*

\* @param \_wordArray an array containg the words from the parsed line of text

\* @param \_currentWord the largest word found.

\* @return the largest word

\*/

protected String FindLongestWord(String[] \_wordArray, String \_currentWord) {

// loop the array comparing the words to find the largest one

for(String \_item: \_wordArray) {

// if the words are the same length then do nothing and keep the first word

if(\_item.length() == \_currentWord.length()) {

return \_currentWord; //short circuit and get out

} // end if

// new word is larger than current largest word so replace current largest word with new word

if(\_item.length() > \_currentWord.length()) \_currentWord = \_item;

} // end for

// return new word

return \_currentWord;

} // end FindLongestWord

/\*\*

\* <h1>ConsoleDisplay</h1>

\*

\* <p><strong><em>Notes:</em></strong> Displays results of the book being processed.</p>

\*

\* @custom.precondition variable containing line/word count as well as largest word

\*

\* @custom.postcondition simple output to console based on information supplied

\*

\* @param \_lineCount number of lines counted

\* @param \_wordCount number of words counted

\* @param \_largestWord largest word found

\*/

protected void ConsoleOutput(int \_lineCount, int \_wordCount, String \_largestWord) {

// give me some output

System.out.println("\nNumber of lines read: " + \_lineCount);

System.out.println("\nNumber of words read: " + \_wordCount);

System.out.println("\nAverage number of words per line: " + (\_wordCount / \_lineCount));

System.out.printf("\nThe FIRST word found with the most characters was '%s', it is %d characters long.\n", \_largestWord, \_largestWord.length());

} // ConsoleDisplay

} // end class