*Your Name Here* ITI-2450 Elementary Data Structures and Algorithms

**1:(place the Part #1 application code with output below this line):**

package javafiles;

import java.io.File;

import java.util.Scanner;

/\*\*

\* PsalmsReaderMain class - Template for Assignment-4

\* @author TMyatt

\* @version 1.0

\*/

public class PsalmsReaderMain {

static String words[] = new String[3000];

static int count[] = new int[3000];

static int lastIndex = 0;

/\*\*

\* addWordToList - add word to list or increment counter

\* @param inWord - the word to add to the list

\* @return number of words in the list

\*/

static int addWordToList(String inWord) {

int index = -1; ; // as index is given -1 for comparison as it cannot be negative

for (int currentIndex = 0; currentIndex < lastIndex; ++currentIndex) {

if (words[currentIndex].equals(inWord)) {

index = currentIndex; // give value to index in case word is repeated

break;

}

}

if (index == -1) { // In case word is new in file, add to array

words[lastIndex] = inWord; // assign word to array of words

count[lastIndex] = 1; // assign count as 1, first time counted

lastIndex++; // update last visited index

} else {

count[index]++; // increment in case word again come

}

return lastIndex; // return last index to know how many words are there and their counts

}

/\*\*

\* printWordList - dump the list if count > inMinimum

\*/

static void printWordList(int inMinimum) {

int iteration; // initialized iteration variable to be used in for loop

System.out.println("Most Repeated Words with Count greater than 1000 are given below:");

for (iteration = 0; iteration < lastIndex; iteration++) {

// iteration will be index for count array, if any word has count greater than

// inMinimum, it will be printed

if (words[iteration] != null && count[iteration] > inMinimum) {

System.out.println(words[iteration] + ": " + count[iteration]);

}

}

}

/\*\*

\* getVerse - Parse and return only the Psalms verse

\* @param inLine - the line to parse on tab delimiter

\* @return right half of tab delimited string

\*/

static String getVerse(String inLine) {

return inLine.split("\t")[1]; // split by tab as first part will be Psalms 147:20

// and second one will be verse, so concern is with verse

}

/\*\*

\* PsalmsReaderMain main() - Template for Assignment-4

\* @param args

\*/

public static void main(String[] args) {

try {

// Open the required text file for sequential read

Scanner inputFile =

new Scanner (new File("bible-psalms.txt"));

// Check for EOF, read the next line, and display it

while (inputFile.hasNextLine()) {

String inLine, verse;

String[] verseParsed;

inLine = inputFile.nextLine();

verse = getVerse(inLine);

verseParsed = verse.split("[ :;,.'!?()-]+");

for (String s: verseParsed) {

addWordToList(s.toLowerCase());

}

}

// Close the required file when EOF is reached

inputFile.close();

printWordList(1000);

} // END try

catch (Exception e) {

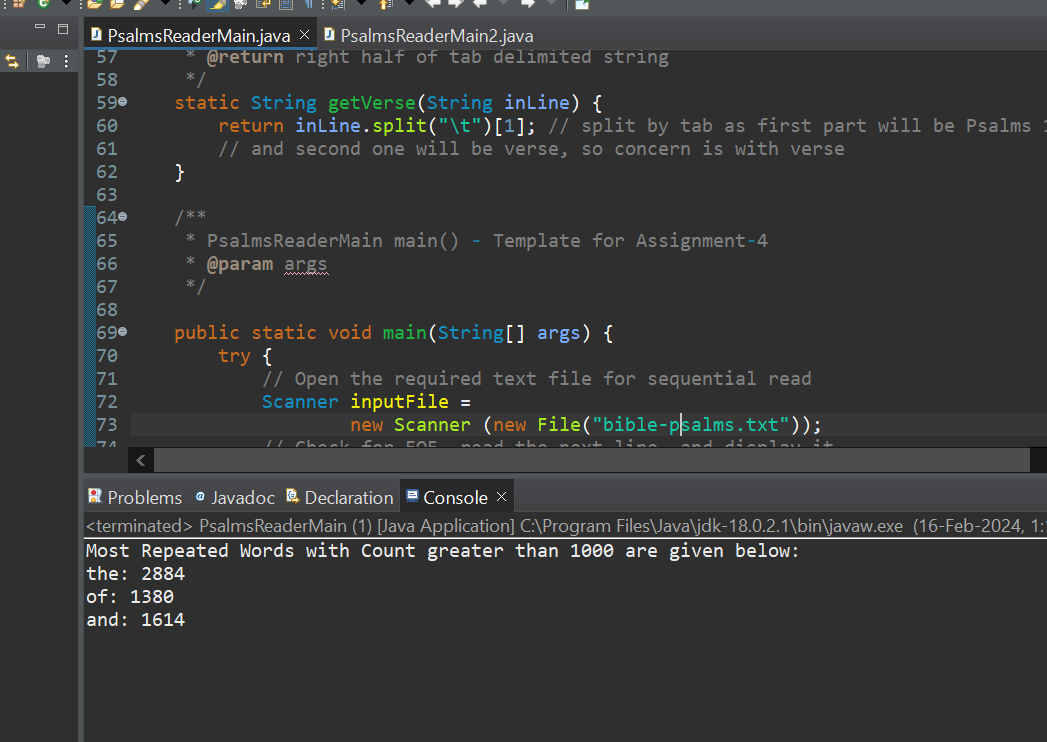
// All Exceptions come here for graceful termination

System.out.println("PsalmsReaderMain Error: " + e);

} // END catch

} // END main

} // END class



**2:(place a description of your changes with output below this line):**

In part 2, array length for words[] and count[] is increased up to 15000 to accommodate more words as there will be more unique words in bible-complete text file. Hence, there could be a possibility of index out of bound error. These changes are made in part 2 for having words with more than count of 10000 in bible-complete text file.

Changes:

1

static String *words*[] = new String[15000];

static int *count*[] = new int[15000];

2

Scanner inputFile = new Scanner(new File("bible-complete.txt"));

3

*printWordList*(10000);

