CSE143—Computer Programming II
Programming Assignment #2
Due: see Canvas class site

This assignment will give you practice using set and map collections from JAVA API. You will read in the text of various books and use the provided lists of "positive" and "negative" words to classify each book as a "comedy" or a "tragedy." The practice of attempting to determine the overall tone of a block of text is known as "sentiment analysis." In practice, algorithms for sentiment analysis can be quite sophisticated. Ours is clearly quite simple. We'll see how accurate it is at classifying some books.

Part A: Create a BookAnalyzer Class

This class will have the following fields (posWords and negWords should both be SETS, bookWords should be an ARRAYLIST):

posWords
negWords
bookWords

Your class should have a CONSTRUCTOR that takes 3 INPUT PARAMETERS: the "positive" and "negative" text files, which are used to construct the posWords and negWords sets, and a book text file, which is used to construct the bookWords ArrayList object. Your class should handle bad input gracefully. One option might be to have "default" lists for positive and negative words, such as the lists shown below (which you would need to convert to sets):

```
String[] posList = {"good", "nice", "love", "excellent", "great", "awesome",
"wonderful", "fantastic", "fabulous", "like"};

String [] negList = {"bad", "awful", "hate", "terrible", "miserable", "sad",
"dislike", "atrocious", "depressed", "cry"};
```

Another option (and the one you should probably use for the book file name) is to keep querying the user until a valid file name is entered.

Your class should contain methods to analyze the book. You get to determine what methods you will need. Hint: You may want to have methods that create positive and negative sets of words for a given book.

Your BookAnalyzer class should determine whether the book being analyzed is a comedy (positive word count > negative word count), a tragedy (positive word count < negative word count), or a boring book (positive word count = negative word count).

Part B: Create a SentimentReport Class

This class will contain your client code. You should include the following methods:

`intro()` This method will provide the user with information and instructions.
`getReport()` This method takes a book file as a parameter and prints out a report on the book. Most of the code for the report should be in the Book Analyzer class. Hint: you may want to use a loop in this method to process the books in the book list one by one. The calls needed to time the analysis could also be included in the loop.

The report should include the following information:

- Which book file is being analyzed
- The total number of words in the book
- The total number of positive words in the book
- The percentage of the total words that are positive words
- The most commonly occurring positive word and the number of times it occurs
- The total number of negative words in the book
- The percentage of the total words that are negative words
- The most commonly occurring negative word and the number of times it occurs
- Whether the book analyzed was a comedy, a tragedy, or a boring book
- The time (in milliseconds) used in completing the book analysis

Part C: Test Your Code

Design your client class (the Sentiment Report class) to work in either of 2 modes. Set the mode using a global variable as shown:

```
public static final boolean TEST_MODE = true;
```

In test mode, your client class should analyze the following books (text files provided with this assignment):

Macbeth.txt

MuchAdoAboutNothing.txt

When not in test mode, your client class should interactively accept a list of books input by the user (request book file names one by one from the user and add to a list in your client code).

Go to the following website: https://www.gutenberg.org/ to download text files of at least 2 books to use to test your code.

Part D: What to Submit

Submit the following files:

- BookAnalyzer.java
- SentimentReport.java
- Your two book files (.txt files) (not for the books used in Part C!)
- A .doc or .pdf of your output when running in test mode on the book files provided and when running in interactive mode using the book files you selected.