COMP163 Movie Review Evaluation

The purpose of this assignment is to be able to input the text of a movie review and determine if it is a positive or negative review, that is, does the reviewer like or dislike the movie. Sentiment Analysis is a Big Data problem which seeks to determine the general attitude of a writer given some text they have written. For instance, we would like to have a program that could look at the text "The film was a breath of fresh air" and realize that it was a positive statement while "It made me want to poke out my eye balls" is negative.[1]

One simple *(but, regretfully, not very effective)* algorithm that we can use for this is to assign a numeric value to words based on how positive or negative that word is and then score the statement based on the average value of the words. To create numerical values for words, the program will read a file of over 8,000 movie reviews that have a rating number from 0 to 4 along with the text of a review. For each word in the reviews, the program will keep the sum of the ratings and the number of times the word occurred. The average "goodness" of the word can be calculated by dividing the sum of the ratings where that word appears by the number of times the word appeared.

Information about a word should be kept in the class `WordValue`. The `WordValue` class needs to keep two class instance variables to hold:

- The number of times the word has been used.
- The sum of the review ratings where the word was used

A constructor method should be written to set the use count to one and initialize the sum to a parameter. Two additional methods are needed:

```
public void addRating(int rating)
```

which increments the use count by one and adds the rating to the sum.

```
public double avgRating()
```

which returns the average rating (sum divided by the use count).

In a separate class (and therefore a separate file) the main program has two major parts. The first reads the file `moviereviews.txt` and builds a dictionary of `WordValue` objects. The second part reads a review from the keyboard and determines if it is a positive or negative review.

The dictionary of words is best kept in a hash table. The hash table can be created in Java by:

```
java.util.Hashtable<String, WordValue> dict =
                                        new java.util.Hashtable<>(18000);
```

A hash table works like a dictionary where each object has a key or name. When you put an object in the dictionary, you give it both the `WordValue` object to store in the dictionary and a key String to find it. You can then retrieve the object by giving it the key String. The hash table class has two methods of interest to this program, `put` and `get`. The `put` method has two parameters, the key String and the `WordValue` object to be saved in the dictionary. In this program the key is a word from the review. The `get` method of a hash table retrieves the previously stored object with this key String. If no object was previously stored with this key String, the method returns null.

[1] The idea for this program comes from Dr. Eric D. Manley and Dr. Timothy M. Urness of Drake University

hash table example:

```
WordValue cat = new WordValue( dog );
String aardvark = "something";
dict.put( aardvark, cat);
cat = dict.get( aardvark );   // may return null if aardvark is not in the dictionary
```

The file `moviereviews.txt` contains over 8000 reviews. Each review starts with a number 0 through 4 with the following meaning:

0 : negative
1 : somewhat negative
2 : neutral
3 : somewhat positive
4 : positive

After the integer is a series of words separated by white space. The end of each review is indicated by a single period preceded by a space. The punctuation has been separated from the words, so reading the file with the Scanner method `next()` will return each English word. For consistency, your program should convert all input to lower case so the dictionary will find them regardless of case. If the first character of the word is not a letter, the word should be ignored. The Java expression `Character.isLetter(inWord.charAt(0))` will be true if the first character of inWord is a letter.

A general outline for the program is:

```
// create dictionary
declare entry as WordValue reference
while more data in the file
        read rating level number
        read next word and convert to lower case
        while word not equal to "."
                if word starts with a letter
                        entry = dict.get( word )
                        if entry is null
                                entry = new WordValue( rating level )
                                dict.put( word, entry )
                        else
                                entry.addRating( rating level )
                read next word
// evaluate new movie review
read word and convert to lower case
while word not equal to "."
        entry = dict.get( word )
        if entry not null
                add entry.avgRating() to sum for review
                increment count of words in review
        read word and convert to lower case
review average is sum / count of words
if review average is greater than 2.0 display positive else display negative
```

Example output – (Note lone period at the end of the line .)

```
Enter a review >The film was a breath of fresh air .
The average of this rating is 2.156184387867476
positive review

Enter a review >It made me want to poke out my eye balls .
The average of this rating is 1.9824086446499938
negative review
```