Programming Exercise: Telling a Random Story

Assignment 1: Most Frequent Word

You will write a program to determine the word that occurs the most often in a file. If more than one word occurs as the most often, then return the first such word found. You should make all words lowercase before counting them. Thus, “This” and “this” will both be counted as the lowercase version of “this.” You should not consider punctuation, so “end” and “end,” will be considered different words. Use the **WordFrequencies** program in the lesson as a starting point.

Specifically, you should do the following:

- Create a new project in BlueJ and then create a new class called **WordFrequencies**. Put all the following items in this class.
- Create two private variables. One is called **myWords** and should be an ArrayList of type String to store unique words from a file, and one is called **myFreqs** and should be an ArrayList of type Integer. The kth position in **myFreqs** should represent the number of times the kth word in **myWords** occurs in the file.
- Write a constructor **WordFrequencies**, and initialize the private variables.
- Write a void method **findUnique** that has no parameters. This method should first clear both **myWords** and **myFreqs**, using the .clear() method. Then it selects a file and then iterates over every word in the file, putting the unique words found into **myWords**. For each word in the kth position of **myWords**, it puts the count of how many times that word occurs from the selected file into the kth position of **myFreqs**, as was demonstrated in the lesson.
- Write a void **tester** method that has no parameters. This method should call **findUnique**. Then print out the number of unique words, and for each unique word, print the frequency of each word and the word, as was demonstrated in the lesson.

- Write the method **findIndexOfMax** that has no parameters. This method returns an int that is the index location of the largest value in **myFreqs**. If there is a tie, then return the first such value.
- Add code to the **tester** method to determine and print the word that occurs the most often in a selected file and how many times it occurs. You should find it helpful to call **findIndexOfMax**.

For example, if the file were **testwordfreqs.txt**:

```
This is a test. Yes a test of a test. Test.
```

Then the output would be:

```
Number of unique words: 7
1    this
1    is
3    a
3    test.
1    yes
1    test
1    of
The word that occurs most often and its count are: a 3
```

We are ignoring punctuation, so note that “test.” and “test” are different, as the first one has a period with it. Also note that there is a tie—two words are counted three times; you should return the first such word found which is “a”.

Assignment 2: Character Names

Write a program to determine the characters in one of Shakespeare's plays that have the most speaking parts. Consider the play "The Tragedy of Macbeth" in the file **macbeth.txt**. Here are a few lines from the file put into a much smaller file called **macbethSmall.txt:**

```
MACBETH. My dearest love,
    Duncan comes here tonight.
LADY MACBETH. And when goes hence?
MACBETH. Tomorrow, as he purposes.
LADY MACBETH. O, never
    Shall sun that morrow see!
    Your face, my Thane, is as a book where men
    May read strange matters. To beguile the time,
    Look like the time; bear welcome in your eye,
    Your hand, your tongue; look like the innocent flower,
    But be the serpent under it. He that's coming
    Must be provided for; and you shall put
    This night's great business into my dispatch,
    Which shall to all our nights and days to come
    Give solely sovereign sway and masterdom.
MACBETH. We will speak further.
```

Note that each speaking part is at the beginning of the line (there may be some blanks before it) and has a period immediately following it. Shakespeare used this format in many of his plays. Sometimes the name of the person to speak was all capitalized and sometimes it was not. Write a program to print out the main characters in one of Shakespeare's plays, those with the most speaking parts. You should identify a speaking part by reading the file line-by-line and finding the location of the first period on the line. Then you will assume that everything up to the first period is the name of a character and count how many times that occurs in the file. You will only print those characters that appear more often than others. Notice our method is somewhat

error prone. For example, a period is also used to indicate the end of a sentence. By printing out only those characters that appear a lot, we will get rid of most of the errors. Periods that indicate the end of a sentence will likely be a unique phrase so you won't print that as it would just occur once or maybe twice.

For the file **macbethSmall.txt**, if we process it and print ALL the possible speaker characters and counts found, we would get the following output:

```
MACBETH  3
  Duncan comes here tonight   1
LADY MACBETH  2
  May read strange matters    1
  But be the serpent under it 1
  Give solely sovereign sway and masterdom    1
```

If we only print those with a count greater than 1, then our output is:

```
MACBETH  3
LADY MACBETH  2
```

In processing the complete play in **macbeth.txt** you should not print out every count—you would have too much output, so instead print every count that is greater than or equal to some number (you decide what that number is).

Specifically, you should do the following:

- Create a class named **CharactersInPlay**. Put all the following items below in this class.
- You will need to create two private ArrayLists. One to store the the names of the characters you find and one to store the corresponding counts for each character.
- Write a void method named **update** that has one String parameter named **person**. This method should update the two ArrayLists, adding the character's name if it is not already there, and counting this line as one speaking part for this person.

- Write a void method called **findAllCharacters** that opens a file, and reads the file line-by-line. For each line, if there is a period on the line, extract the possible name of the speaking part, and call **update** to count it as an occurrence for this person. Make sure you clear the appropriate instance variables before each new file.
- Write a void method called **tester** that has no parameters. This method should call **findAllCharacters**, and then for each main character, print out the main character, followed by the number of speaking parts that character has. A main character is one who has more speaking parts than most people. You'll have to estimate what that number should be. Test your method on the file **macbethSmall.txt**. and then **macbeth.txt**.
- Write a void method called **charactersWithNumParts** that has two int parameters named **num1** and **num2**, where you can assume **num1** should be less than or equal to **num2**. This method should print out the names of all those characters that have exactly number speaking parts, where number is greater than or equal to **num1** and less than or equal to **num2**. Add code in tester to test this method out.