CSCI 232 Program 2

Due Tuesday June 6th @ 11:59 PM. Please submit this assignment (.java files) to the appropriate dropbox on D2L

Background and Instructions

In this program, You are going to use Java's built in HashSet and HashMap classes to build a document checking application. Your program will be able to identify misspelled words, and print out some interesting information about the frequency of words that appear in the document.

Part I: Loading a Hash Set with English words (`loadHashSet()`)

In order to identify misspelled words, we need to have a "database" of known, correctly-spelled English words. You **must** use a Java HashSet to hold the Strings of all the words that exist in `words.txt`. The input file `words.txt` is essentially a list of (almost) all words in the English language. Later on, when you are spell checking the document in part 3, you can use this Hash Set to determine if a word is misspelled or not. In the DocumentChecker class, there is already an instance field for this HashSet, called **`words`**.

Part II: Loading a Hash Map with word Frequency (`loadHashMap()`)

You **must** use a Java HashMap to calculate the number of occurrences for each unique word in the input file `input.txt`. This input file is the document that the user want to be spellchecked. This method should go through each word in the input file, and calculate the number of occurrence that word appears in the document. The keys of your HashMap will be the word (String), and the value that is linked with the key will be its frequency.

For example, the word "a" appears four times in the document. "and" appears once, "check" three times, and so on…

```
a: 4
and: 1
check: 3
code: 1
documunt: 1
```

Capitalization and punctuation matter here, and you will need to do some String manipulation to handle this. "The", "the", "the," and "tHe?" should all count towards the number of occurrences for the word "the".

Part III: Spellchecking the document (`spellcheck()`)

With your loaded HashSet from part I, you will read in from input.txt again, and check each word to see if they appear in the HashSet or not. If they do not appear in the HashSet, then the word is considered to be misspelled, and needs to flagged in the output. Once again, capitalization and punctuation matter here, and you will likely need to do some similar String manipulation to ensure the spellchecking works. The original input should be printed out, but any misspelled words need to wrapped in angle brackets < >.

Original Input: `Hai there, what is yuor nameee?`

Output of program: `<Hai> there, what is <yuor> <nameee?>`

Part IV: Printing out word frequency alphabetically (`wordCountAlphabetically()`)

With your loaded HashMap from part II, you will print out the contents of the HashMap **sorted by the key values**.

Hint: Consider creating a new Set of just the words (`keySet()`), and then sort that Set. Then do a for loop through this sorted Set, and use the HashMap to get each word's frequency.

Part V: Printing out word frequency by frequency (`wordCountByFrequency()`)

With your loaded HashMap from part II, you will print out the contents of the HashMap **sorted by the word frequencies from greatest to least**.

In the sample input, "the" and "a" were the most commonly used words, both having a frequency of 4.

```
4: the, a
3: spell, is, check, to
2: hi, test, program
1: code, use, for, your, output, identical, and, of, needz, mispeled,
```

Hint: This one is tricky. Consider creating a new HashMap. Instead of having a HashMap that computes the ***Count per Word*** (this is what you did in part II), write a HashMap that computes the ***Words per Count***. So, the keys will be an Integer, and the value will be a HashSet of Strings that had that count.

4: [the, a]

3: [spell, is, check, to]

2: [hi, test, program]

…

Then, you can do a for loop to print out this HashMap in a nice, clean manner (you may need to use the **`.join`** String method to combine contents of a Set).

Sample Output

Menu output 1: Spellchecking the input file

```
Document Spell Check Program

---------------------------

1: Spell Check your document (input.txt)

2: Print word frequency alphabetically

3: Print word frequency from greatest to least

4: Exit program

Your choice?

1

Welcome to the <documunt> spell check program. The
spell check code needs to output the identical text,
except for <mispeled> words. You <shoould> test your
program on a <varietyi> of input. Your spell check
<needz> to use a Hash Map and Hash Set. Is yours
<werking?> A a a a is is the the test hi hello hi
```

Menu Option 2: Print word frequency alphabetically.

```
Document Spell Check Program
----------------------------
1: Spell Check your document (input.txt)
2: Print word frequency alphabetically
3: Print word frequency from greatest to least
4: Exit program
Your choice?
2
a: 6
and: 1
check: 3
code: 1
documunt: 1
except: 1
for: 1
hash: 2
hello: 1
hi: 2
identical: 1
input: 1
is: 3
map: 1
mispeled: 1
needs: 1
needz: 1
of: 1
on: 1
output: 1
program: 2
set: 1
shoould: 1
spell: 3
test: 2
text: 1
the: 5
to: 3
use: 1
varietyi: 1
welcome: 1
werking: 1
words: 1
you: 1
your: 2
yours: 1
```

Menu Option 3: Print word frequency from greatest to least.

.

```
Document Spell Check Program

----------------------------

1: Spell Check your document (input.txt)

2: Print word frequency alphabetically

3: Print word frequency from greatest to least

4: Exit program

Your choice?

3

6: a
5: the
3: spell, is, check, to
2: hi, test, program, your, hash
1: code, use, for, output, identical, and, of, needz, mispeled,
text, welcome, map, you, on, needs, set, varietyi, words, werking,
shoould, input, documunt, except, hello, yours
```

Starting Code:

You should not modify the Demo class, but you are allowed to more methods to the SpellChecker class if you think you need them.

- Program2Demo.java(https://www.cs.montana.edu/pearsall/classes/summer2023/232/programs/Program2Demo.java)
- SpellChecker.java (https://www.cs.montana.edu/pearsall/classes/summer2023/232/programs/DocumentChecker.java)
- words.txt (https://www.cs.montana.edu/pearsall/classes/summer2023/232/programs/words.txt)
- input.txt (https://www.cs.montana.edu/pearsall/classes/summer2023/232/programs/input.txt)

Restrictions

You **must** use a HashSet to store the English words, and you **must** use a HashMap to keep track of word frequencies.

Grading

• HashMap and HashSet are loaded correctly (**`loadHashSet(), loadHashMap()`**) - **20 points**

• Your program correctly identifies misspelled words, even in the cases of punctuation and strange capitalization (**`spellcheck()`**) - **30 points**

• Menu option #2 correctly prints out the frequency of words, sorted alphabetically (**`wordCountAlphabetically()`**) – **20 points**

• Menu option #3 correctly prints out the frequency of words, sorted by number of occurrences (**`wordCountByFrequency ()`**) – **20 points**

• Each method written by the user has a comment describing what the method does- **10 points**

NOTE: If your code does not compile, correctness cannot be verified, and you won't receive any points for your code. Turn in code that compiles!