Project-9.4

-- Java SE Runtime Environment 8

1. **Read file**

   I create five array to store the whole words in a file.

```
String[] wordArr1 = read(file1);
```

   I use a string builder to store the word stream, and transform it to an array. And I will delete some unnecessary words or characters.

```
StringBuilder stringBuilder = new StringBuilder();
while (file_input.hasNext()) {
    String str = file_input.nextLine();
    String only_ab = str.replaceAll("[^a-zA-Z]", " ");
    String lowerCase = only_ab.toLowerCase();
    stringBuilder.append(lowerCase);
    stringBuilder.append(" ");
}
String wordStr = delOthers(stringBuilder.toString());
String[] wordArr = wordStr.split(" ");
```

   Part of delOthers method:

```
wordStr = wordStr.replaceAll(art, " ");
wordStr = wordStr.replaceAll(prep, " ");
wordStr = wordStr.replaceAll(pron, " ");
wordStr = wordStr.replaceAll(others, " ");
wordStr = wordStr.replaceAll(" +", " ");
```

2. **Frequency**

   I create five array list to store the word and its frequency in each file. And I use the freq method to compute the frequency of each word.

```
ArrayList <Freq> wordFreq1 = new ArrayList<Freq>();
class Freq {
    String word;
    int freq;
int n1 = freq(wordArr1, wordFreq1);
```

   I create a hash map to store the word and its frequency, and return the number of how many different words are in a file.

```
Map <String, Integer> map = new HashMap <String, Integer> ();
int number = 0;
for (int i = 0; i < wordArr.length; i++) {
    Integer count = map.get(wordArr[i]);
    if(count == null) {
        map.put(wordArr[i], 1);
        number++;
    }
    else {
        map.put(wordArr[i], ++count);
    }
}

for(Map.Entry <String, Integer> entry : map.entrySet()) {
    Freq freq = new Freq(entry.getKey(), entry.getValue());
    wordFreq.add(freq);
}
```

3. **Create a trie and insert words**

   I create a trie to store the word and its frequency, its filename.

```
Trie trie = new Trie();
```

   I use wordInsert method to insert the whole words.

```
wordInsert(trie, wordFreq1, n1, file1);
```

```
for(int i = 0; i < word.length(); i++) {
    Node child = current.subNode(word.charAt(i));
    if(child != null) {
        current = child;
    }
    else {
        current.childList.add(new Node(word.charAt(i)));
        current = current.subNode(word.charAt(i));
    }
    current.count++;
}
current.isEnd = true;
```

   For the last node, I create a structure to store the frequency and filename.

```
class LastNode {
    boolean hasWord;
    int freq;
if (filename == "1.txt") {
    current.has1.hasWord = true;
    current.has1.freq = freq;
}
```

4. **Enter word(s)**

   I use a string array to store the word the user want to search. And I transform these words into lower case.

```
// get the word(s)
String searchingWord = tfWord.getText().trim();
String searchingWordLowerCase = searchingWord.toLowerCase();
String[] searchingWordArr = searchingWordLowerCase.toString().split(" ");
```

And I create a 2-d array to store the result. First column is the filename, second column is the sum of frequency of each word (I use it as the relevancy), and other columns are the frequency of each word.

```
// create an array to store the frequency of each word
int[][] result = new int[5][wordNum];
for (int i = 0; i < 5; i++) {
    result[i][0] = i + 1;
    for (int j = 1; j < wordNum; j++) {
        result[i][j] = 0;
    }
}
```

5. **Check**

If I find the word in the trie, the method will return true, and the result array will store the frequency.

```
Node current = root;
for(int i = 0; i < word.length(); i++) {
    if(current.subNode(word.charAt(i)) == null) {
        return false;
    }
    else {
        current = current.subNode(word.charAt(i));
    }
}
if (current.isEnd == true) {
    // get the frequency of the word in each file
    result[0][index + 2] = current.has1.freq;
    result[1][index + 2] = current.has2.freq;
    result[2][index + 2] = current.has3.freq;
    result[3][index + 2] = current.has4.freq;
    result[4][index + 2] = current.has5.freq;
    return true;
}
else {
    return false;
}
```

6. **Result**

Finally, I get the result of the frequency of each word in each file. I can calculate the sum of frequency, and sort them, and display them by order. Besides, I create a UI to show the result.

```
sum += result[i][j];
bubbleSort(result, 5, wordNum);
```