Practical No.1

Title: Implement Conflation algorithm to generate document representative of a text file.

Program :

```
package isr;


import java.io.*;
import java.util.Scanner;


public class Conflation {
   public static void main(String[] args) throws IOException {
      try {
         File fi = new File("Input.txt");
         Scanner scl = new Scanner(fi);
         int ch;


         do {
            System.out.println("1. Display the file");
            System.out.println("2. Remove Stop Words");
            System.out.println("3. Suffix Stripping");
            System.out.println("4. Count Frequency");
            System.out.println("Enter your choice:");


            Scanner sc = new Scanner(System.in);
            ch = sc.nextInt();


            switch (ch) {
               case 1:
                  while (scl.hasNext()) {
```

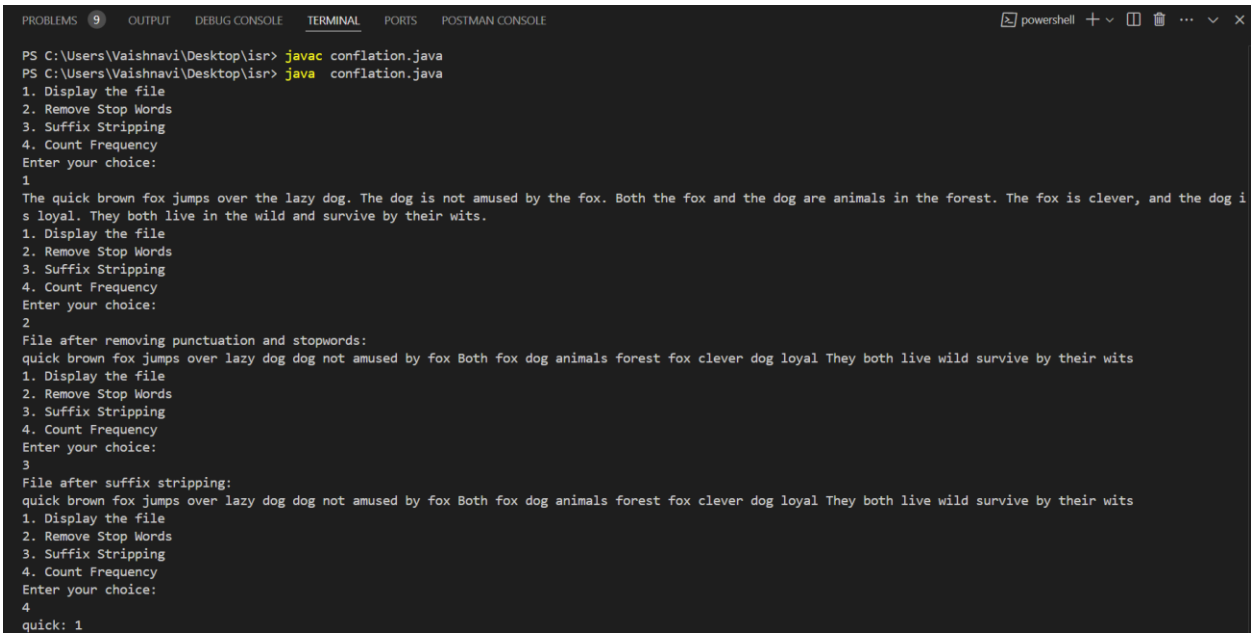
```
                System.out.print(scl.next() + " ");
            }
            System.out.println();
            break;

        case 2:
            remove_punctuation(fi);
            remove_stop_words(fi);
            break;

        case 3:
            suffix_stripping();
            break;

        case 4:
            frequency_count();
            break;
      }
    } while (ch != 4);

  } catch (FileNotFoundException e) {
    System.out.println(e);
  }
}

  private static void remove_punctuation(File fi) throws IOException {
    Scanner sc_punctuation = new Scanner(fi);
    BufferedWriter out = new BufferedWriter(new
FileWriter("without_punctuation_and_stopwords.txt"));
```

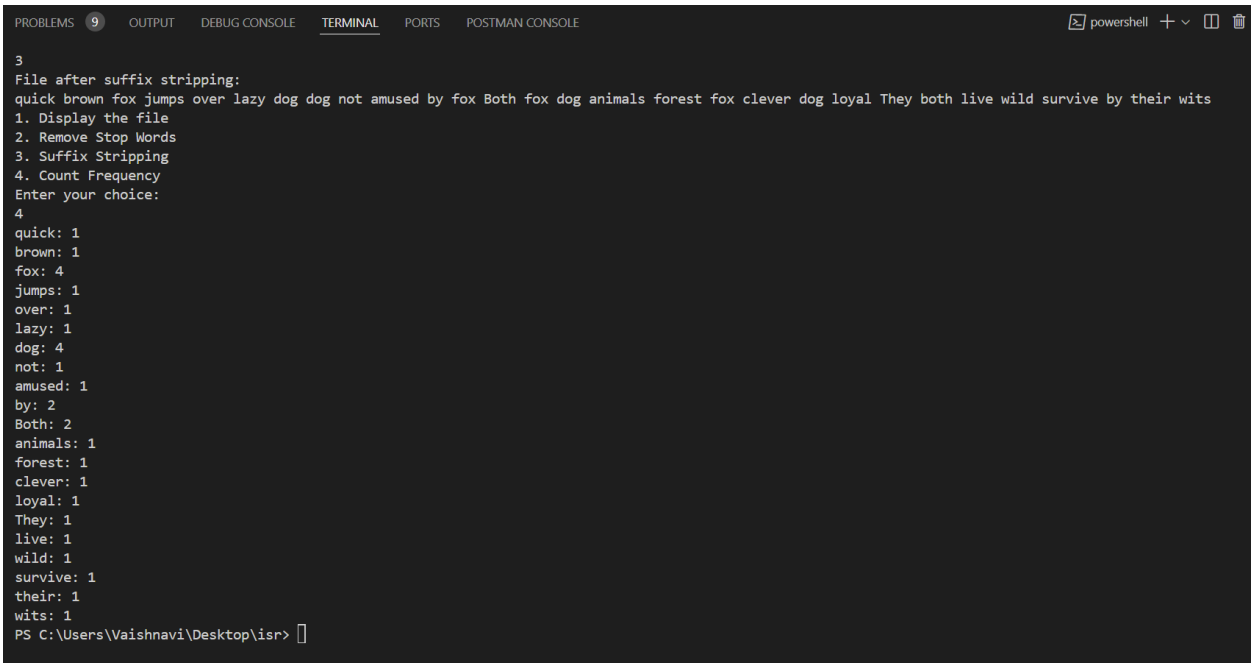
```
        while (sc_punctuation.hasNext()) {
            String str_p = sc_punctuation.next();
            String str_r = str_p.replaceAll("[^a-zA-Z]", "");
            if (!str_r.toLowerCase().equals("the") && !str_r.toLowerCase().equals("is") &&
                !str_r.toLowerCase().equals("and") && !str_r.toLowerCase().equals("of") &&
                !str_r.toLowerCase().equals("are") && !str_r.toLowerCase().equals("for") &&
                !str_r.toLowerCase().equals("in")) {
                out.write(str_r + " ");
            }
        }

        out.close();
        System.out.println("File after removing punctuation and stopwords:");
        BufferedReader br = new BufferedReader(new
FileReader("without_punctuation_and_stopwords.txt"));
        String z;
        while ((z = br.readLine()) != null) {
            System.out.println(z);
        }
        br.close();
    }

    private static void suffix_stripping() throws IOException {
        Scanner scl = new Scanner(new File("without_punctuation_and_stopwords.txt"));
        BufferedWriter out = new BufferedWriter(new FileWriter("suffix_stripping2.txt"));

        while (scl.hasNext()) {
            String str = scl.next();
```

```
        if (str.endsWith("ier")) str = str.replaceAll("ier$", "y");
        else if (str.endsWith("ied")) str = str.replaceAll("ied$", "y");
        else if (str.endsWith("iest")) str = str.replaceAll("iest$", "y");

        out.write(str + " ");
    }
    out.close();
    scl.close();

    System.out.println("File after suffix stripping:");
    BufferedReader br = new BufferedReader(new FileReader("suffix_stripping2.txt"));
    String z;
    while ((z = br.readLine()) != null) {
        System.out.println(z);
    }
    br.close();
}

private static void frequency_count() throws FileNotFoundException, IOException {
    Scanner sc = new Scanner(new File("suffix_stripping2.txt"));
    String[] words = new String[1000];
    int[] counts = new int[1000];
    int i = 0;

    while (sc.hasNext()) {
        String word = sc.next();
        boolean found = false;
        for (int j = 0; j < i; j++) {
            if (word.equalsIgnoreCase(words[j])) {
```

```
                counts[j]++;
                found = true;
                break;
            }
        }
        if (!found) {
            words[i] = word;
            counts[i] = 1;
            i++;
        }
    }

    for (int j = 0; j < i; j++) {
        System.out.println(words[j] + ": " + counts[j]);
    }
  }

  private static void remove_stop_words(File fi) {

  }
}
```

OUTPUT :


PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
powershell
PS C:\Users\Vaishnavi\Desktop\isr> javac conflation.java
PS C:\Users\Vaishnavi\Desktop\isr> java conflation.java
1. Display the file
2. Remove Stop Words
3. Suffix Stripping
4. Count Frequency
Enter your choice:
1
The quick brown fox jumps over the lazy dog. The dog is not amused by the fox. Both the fox and the dog are animals in the forest. The fox is clever, and the dog i
s loyal. They both live in the wild and survive by their wits.
1. Display the file
2. Remove Stop Words
3. Suffix Stripping
4. Count Frequency
Enter your choice:
2
File after removing punctuation and stopwords:
quick brown fox jumps over lazy dog dog not amused by fox Both fox dog animals forest fox clever dog loyal They both live wild survive by their wits
1. Display the file
2. Remove Stop Words
3. Suffix Stripping
4. Count Frequency
Enter your choice:
3
File after suffix stripping:
quick brown fox jumps over lazy dog dog not amused by fox Both fox dog animals forest fox clever dog loyal They both live wild survive by their wits
1. Display the file
2. Remove Stop Words
3. Suffix Stripping
4. Count Frequency
Enter your choice:
4
quick: 1


PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
powershell
3
File after suffix stripping:
quick brown fox jumps over lazy dog dog not amused by fox Both fox dog animals forest fox clever dog loyal They both live wild survive by their wits
1. Display the file
2. Remove Stop Words
3. Suffix Stripping
4. Count Frequency
Enter your choice:
4
quick: 1
brown: 1
fox: 4
jumps: 1
over: 1
lazy: 1
dog: 4
not: 1
amused: 1
by: 2
Both: 2
animals: 1
forest: 1
clever: 1
loyal: 1
They: 1
live: 1
wild: 1
survive: 1
their: 1
wits: 1
PS C:\Users\Vaishnavi\Desktop\isr>