**Question 1:**

Download a library that supports Porter’s stemmer and use it to stem the words in file “content.txt”. The results are to be stored in a new file called “stemmed-content.txt”. Then, create another file “counts-statisitcs.txt” that stores the frequency of each word in stemmed-content.txt, each <word, freq> pair in a separate line. The source code + the three files should be placed in a folder called “Q1”.

Q1: **NOTE: Specify CORRECT PATH For Ur Given Files.**

package org.tartarus.martin;

import java.io.\*;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

import java.util.Scanner;

import jdk.nashorn.internal.runtime.ListAdapter;

public class Stemmer

{ private char[] b;

private int i, /\* offset into b \*/

i\_end, /\* offset to end of stemmed word \*/

j, k;

private static final int INC = 50;

/\* unit of size whereby b is increased \*/

public Stemmer()

{ b = new char[INC];

i = 0;

i\_end = 0;

}

public void add(char ch)

{ if (i == b.length)

{ char[] new\_b = new char[i+INC];

for (int c = 0; c < i; c++) new\_b[c] = b[c];

b = new\_b;

}

b[i++] = ch;

}

public void add(char[] w, int wLen)

{ if (i+wLen >= b.length)

{ char[] new\_b = new char[i+wLen+INC];

for (int c = 0; c < i; c++) new\_b[c] = b[c];

b = new\_b;

}

for (int c = 0; c < wLen; c++) b[i++] = w[c];

}

public String toString() { return new String(b,0,i\_end); }

public int getResultLength() { return i\_end; }

public char[] getResultBuffer() { return b; }

private final boolean cons(int i)

{ switch (b[i])

{ case 'a': case 'e': case 'i': case 'o': case 'u': return false;

case 'y': return (i==0) ? true : !cons(i-1);

default: return true;

}

}

private final int m()

{ int n = 0;

int i = 0;

while(true)

{ if (i > j) return n;

if (! cons(i)) break; i++;

}

i++;

while(true)

{ while(true)

{ if (i > j) return n;

if (cons(i)) break;

i++;

}

i++;

n++;

while(true)

{ if (i > j) return n;

if (! cons(i)) break;

i++;

}

i++;

}

}

/\* vowelinstem() is true <=> 0,...j contains a vowel \*/

private final boolean vowelinstem()

{ int i; for (i = 0; i <= j; i++) if (! cons(i)) return true;

return false;

}

/\* doublec(j) is true <=> j,(j-1) contain a double consonant. \*/

private final boolean doublec(int j)

{ if (j < 1) return false;

if (b[j] != b[j-1]) return false;

return cons(j);

}

/\* cvc(i) is true <=> i-2,i-1,i has the form consonant - vowel - consonant

and also if the second c is not w,x or y. this is used when trying to

restore an e at the end of a short word. e.g.

cav(e), lov(e), hop(e), crim(e), but

snow, box, tray.

\*/

private final boolean cvc(int i)

{ if (i < 2 || !cons(i) || cons(i-1) || !cons(i-2)) return false;

{ int ch = b[i];

if (ch == 'w' || ch == 'x' || ch == 'y') return false;

}

return true;

}

private final boolean ends(String s)

{ int l = s.length();

int o = k-l+1;

if (o < 0) return false;

for (int i = 0; i < l; i++) if (b[o+i] != s.charAt(i)) return false;

j = k-l;

return true;

}

/\* setto(s) sets (j+1),...k to the characters in the string s, readjusting

k. \*/

private final void setto(String s)

{ int l = s.length();

int o = j+1;

for (int i = 0; i < l; i++) b[o+i] = s.charAt(i);

k = j+l;

}

/\* r(s) is used further down. \*/

private final void r(String s) { if (m() > 0) setto(s); }

/\* step1() gets rid of plurals and -ed or -ing. e.g.

caresses -> caress

ponies -> poni

ties -> ti

caress -> caress

cats -> cat

feed -> feed

agreed -> agree

disabled -> disable

matting -> mat

mating -> mate

meeting -> meet

milling -> mill

messing -> mess

meetings -> meet

\*/

private final void step1()

{ if (b[k] == 's')

{ if (ends("sses")) k -= 2; else

if (ends("ies")) setto("i"); else

if (b[k-1] != 's') k--;

}

if (ends("eed")) { if (m() > 0) k--; } else

if ((ends("ed") || ends("ing")) && vowelinstem())

{ k = j;

if (ends("at")) setto("ate"); else

if (ends("bl")) setto("ble"); else

if (ends("iz")) setto("ize"); else

if (doublec(k))

{ k--;

{ int ch = b[k];

if (ch == 'l' || ch == 's' || ch == 'z') k++;

}

}

else if (m() == 1 && cvc(k)) setto("e");

}

}

/\* step2() turns terminal y to i when there is another vowel in the stem. \*/

private final void step2() { if (ends("y") && vowelinstem()) b[k] = 'i'; }

/\* step3() maps double suffices to single ones. so -ization ( = -ize plus

-ation) maps to -ize etc. note that the string before the suffix must give

m() > 0. \*/

private final void step3() { if (k == 0) return; /\* For Bug 1 \*/ switch (b[k-1])

{

case 'a': if (ends("ational")) { r("ate"); break; }

if (ends("tional")) { r("tion"); break; }

break;

case 'c': if (ends("enci")) { r("ence"); break; }

if (ends("anci")) { r("ance"); break; }

break;

case 'e': if (ends("izer")) { r("ize"); break; }

break;

case 'l': if (ends("bli")) { r("ble"); break; }

if (ends("alli")) { r("al"); break; }

if (ends("entli")) { r("ent"); break; }

if (ends("eli")) { r("e"); break; }

if (ends("ousli")) { r("ous"); break; }

break;

case 'o': if (ends("ization")) { r("ize"); break; }

if (ends("ation")) { r("ate"); break; }

if (ends("ator")) { r("ate"); break; }

break;

case 's': if (ends("alism")) { r("al"); break; }

if (ends("iveness")) { r("ive"); break; }

if (ends("fulness")) { r("ful"); break; }

if (ends("ousness")) { r("ous"); break; }

break;

case 't': if (ends("aliti")) { r("al"); break; }

if (ends("iviti")) { r("ive"); break; }

if (ends("biliti")) { r("ble"); break; }

break;

case 'g': if (ends("logi")) { r("log"); break; }

} }

/\* step4() deals with -ic-, -full, -ness etc. similar strategy to step3. \*/

private final void step4() { switch (b[k])

{

case 'e': if (ends("icate")) { r("ic"); break; }

if (ends("ative")) { r(""); break; }

if (ends("alize")) { r("al"); break; }

break;

case 'i': if (ends("iciti")) { r("ic"); break; }

break;

case 'l': if (ends("ical")) { r("ic"); break; }

if (ends("ful")) { r(""); break; }

break;

case 's': if (ends("ness")) { r(""); break; }

break;

} }

/\* step5() takes off -ant, -ence etc., in context <c>vcvc<v>. \*/

private final void step5()

{ if (k == 0) return; /\* for Bug 1 \*/ switch (b[k-1])

{ case 'a': if (ends("al")) break; return;

case 'c': if (ends("ance")) break;

if (ends("ence")) break; return;

case 'e': if (ends("er")) break; return;

case 'i': if (ends("ic")) break; return;

case 'l': if (ends("able")) break;

if (ends("ible")) break; return;

case 'n': if (ends("ant")) break;

if (ends("ement")) break;

if (ends("ment")) break;

/\* element etc. not stripped before the m \*/

if (ends("ent")) break; return;

case 'o': if (ends("ion") && j >= 0 && (b[j] == 's' || b[j] == 't')) break;

/\* j >= 0 fixes Bug 2 \*/

if (ends("ou")) break; return;

/\* takes care of -ous \*/

case 's': if (ends("ism")) break; return;

case 't': if (ends("ate")) break;

if (ends("iti")) break; return;

case 'u': if (ends("ous")) break; return;

case 'v': if (ends("ive")) break; return;

case 'z': if (ends("ize")) break; return;

default: return;

}

if (m() > 1) k = j;

}

/\* step6() removes a final -e if m() > 1. \*/

private final void step6()

{ j = k;

if (b[k] == 'e')

{ int a = m();

if (a > 1 || a == 1 && !cvc(k-1)) k--;

}

if (b[k] == 'l' && doublec(k) && m() > 1) k--;

}

/\*\* Stem the word placed into the Stemmer buffer through calls to add().

\* Returns true if the stemming process resulted in a word different

\* from the input. You can retrieve the result with

\* getResultLength()/getResultBuffer() or toString().

\*/

public void stem()

{ k = i - 1;

if (k > 1) { step1(); step2(); step3(); step4(); step5(); step6(); }

i\_end = k+1; i = 0;

}

public static void main(String[] args) throws FileNotFoundException

{

char[] w = new char[501];

Stemmer s = new Stemmer();

// for (int i = 0; i < args.length; i++)

try

{

// FileInputStream in = new FileInputStream(args[i]);

FileInputStream in = new FileInputStream("C:\\Users\\isgaier\\Desktop\\hw1\\Q1\\context.txt");

PrintStream out = null;

File file4 = new File("C:\\Users\\isgaier\\Desktop\\hw1\\Q2\\p2.txt");

out = new PrintStream("C:\\Users\\isgaier\\Desktop\\hw1\\Q1\\stemmed-content.txt.txt");

try

{ while(true)

{ int ch = in.read();

if (Character.isLetter((char) ch))

{

int j = 0;

while(true)

{ ch = Character.toLowerCase((char) ch);

w[j] = (char) ch;

if (j < 500) j++;

ch = in.read();

if (!Character.isLetter((char) ch))

{

/\* to test add(char ch) \*/

for (int c = 0; c < j; c++) s.add(w[c]);

/\* or, to test add(char[] w, int j) \*/

/\* s.add(w, j); \*/

s.stem();

{ String u;

/\* and now, to test toString() : \*/

u = s.toString();

/\* to test getResultBuffer(), getResultLength() : \*/

/\* u = new String(s.getResultBuffer(), 0, s.getResultLength()); \*/

out.print(u);

}

break;

}

}

}

if (ch < 0) break;

// System.out.print((char)ch);

out.print((char)ch);

}

}

catch (IOException e)

{

}

}

catch (FileNotFoundException e)

{

}

findWordsCount();

}

static private void findWordsCount() throws FileNotFoundException{

FileInputStream in = new FileInputStream("C:\\Users\\isgaier\\Desktop\\hw1\\Q1\\stemmed-content.txt.txt");

PrintStream out = null;

out = new PrintStream("C:\\Users\\isgaier\\Desktop\\hw1\\Q1\\counts-statisitcs.txt");

List arrayW = new ArrayList();

Scanner input = new Scanner(in);

while (input.hasNext()) {

String word = input.next();

arrayW.add(word);

}

Map<String, Integer> frequencies = new LinkedHashMap<String, Integer>();

for (Object wordd : arrayW) {

if (!wordd.equals("")) {

Integer frequency = frequencies.get(wordd);

if (frequency == null) {

frequency = 0;

}

++frequency;

frequencies.put((String) wordd, frequency);

}

}

out.print(frequencies.entrySet());

out.println();

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 2:**

Write an application that uses regular expressions to search for the following items in file (content.txt):

1. All words starting with capital letters and have a length of at least 4 characters. The words are to be stored in file (p1.txt), each word in a separate line.
2. All words where either the last letter or the preceding one is ‘e’, provided that all other letters are consonants (non-vowels). The words are to be stored in file (p2.txt), each word in a separate line.
3. Abbreviations that consists of uppercase letters and zero or more dots such that
   1. The abbreviation should start with a letter
   2. No consecutive dots are allowed

The abbreviations are to be stored in file (p3.txt), each word in a separate line.

The source code + the three files (content.txt, p1.txt, p2.txt, p3.txt) should be placed in a folder called “Q2”. **NOTE: Specify CORRECT PATH For Ur Given Files.**

Q2:

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package nlp2;

import java.io.File;

import java.io.FileWriter;

import java.io.PrintWriter;

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\*

\* @author pc

\*/

public class one {

StringBuilder htText2 ,htText3,htText4;

Scanner in ;

PrintWriter pwFile = null;

public one(){

htText2 = new StringBuilder();

htText3 = new StringBuilder();

htText4= new StringBuilder();

}

public void readQ1(){

try {

in = new Scanner(new File("C:\\Users\\pc\\Desktop\\Q2\\w.txt"));

//while(in.hasNext()){

while(true){

htText2.append(in.next());

}

}catch(Exception e){

System.out.println(e);

in.close();

}

}

public void readQ2(){

try {

in = new Scanner(new File("C:\\Users\\pc\\Desktop\\y.txt"));

//while(in.hasNext()){

while(true){

htText3.append(in.next());

}

}catch(Exception e){

System.out.println(e);

in.close();

}

}

public void pattenQ1(){

//String rg="(^[A-Z]+.\*{1,4})";

String rg="([A-Z]\\w{4,})";

Pattern pattern =Pattern.compile(rg);

Matcher mat = pattern.matcher(htText2);

/\* try{

FileWriter writer = new FileWriter("C:\\Users\\pc\\Desktop\\p1.txt");

pwFile = new PrintWriter(writer);

/\* pwFile.println("hello");

pwFile.println("hello");

pwFile.println("hello");

pwFile.println("hello");

pwFile.println("hello");\*/

/\*

while(mat.find()){

pwFile.println(mat.matches());

}

pwFile.close();

}catch(Exception e){

System.out.println(e);

pwFile.close();

}\*/

//num l2nha not match b find ma ra7 yrb3ha false l2nah ma bd5al 3la loop

mat.reset();

while(mat.find()){

//for (int i = 1; i <= mat.groupCount(); i++) {

System.out.println(mat.group());

//}

}

}

public void patten2(){

String rg="[^a-i]+e.?|e$";

Pattern pattern =Pattern.compile(rg);

Matcher mat = pattern.matcher(htText3);

// try{

// FileWriter writer = new FileWriter("C:\\Users\\pc\\Desktop\\Q2.txt");

// pwFile = new PrintWriter(writer);

/\* pwFile.println("hello");

pwFile.println("hello");

pwFile.println("hello");

pwFile.println("hello");

pwFile.println("hello");\*/

// pwFile.println(mat.matches());

mat.reset();

while(mat.find()){

//for (int i = 1; i <= mat.groupCount(); i++) {

System.out.println(mat.group());

//}

}

//pwFile.close();

// }catch(Exception e){

//System.out.println(e);

//pwFile.close();

}

public void readQ3(){

try {

in = new Scanner(new File("C:\\Users\\pc\\Desktop\\u.txt"));

//while(in.hasNext()){

while(true){

htText4.append(in.next());

}

}catch(Exception e){

System.out.println(e);

in.close();

}

}

public void pattenQ3(){

//[A-Z]{1,2}(\.[A-Z])\*

String rg="[A-Z]{1,2}(\\.[A-Z])\*";

Pattern pattern =Pattern.compile(rg);

Matcher mat = pattern.matcher(htText4);

// try{

// FileWriter writer = new FileWriter("C:\\Users\\pc\\Desktop\\Q2.txt");

// pwFile = new PrintWriter(writer);

/\* pwFile.println("hello");

pwFile.println("hello");

pwFile.println("hello");

pwFile.println("hello");

pwFile.println("hello");\*/

// pwFile.println(mat.matches());

mat.reset();

while(mat.find()){

//for (int i = 1; i <= mat.groupCount(); i++) {

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println(mat.group());

//}

}

}

}

**NOTE:** **Brief main:**

Main(){

one a = new one();

a.readQ1();

a.pattenQ1();

a.readQ2();

a.patten2();

a.readQ3();

a.pattenQ3();

}

**Question 3:**

In Java, use “Pattern” and “Matcher” classes to build a regular expression using which one can extract phrases that are used in the beginning of a sentence, followed by a comma, and contain at most 4 words.

For example:

**if the input is:**

“In this blog, I will focus on the challenges pertaining to model evaluation I came across while implementing a machine log analytics classification algorithm. Specifically, I will demonstrate the meaning of model evaluation metrics”

**Then, the output will be:**

In this blog

Specifically

(The input file is “content.txt” and the output will be printed on screen, each matched token in a separate line. The .java source code and “content.txt” file should be stored in folder Q3)

**NOTE: Specify CORRECT PATH For Ur Given Files.**

Q3:

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package nlp3;

import java.io.DataInputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import static java.lang.System.in;

import java.util.Scanner;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\*

\* @author pc

\*/

public class Nlp3 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

StringBuilder htText = new StringBuilder();

Scanner in ;

try {

in = new Scanner(new File("C:\\Users\\pc\\Desktop\\context.txt"));

while(true){

htText.append(in.next());

}

}catch(Exception e){

System.out.println(e);

}

String rg="(\\.)(.+?)(\\,)";

Pattern pattern =Pattern.compile(rg);

Matcher mat = pattern.matcher(htText);

while(mat.find()){

System.out.println(mat.group(2));

}

}//end main

}