**CSC 323 Project 1.2 (Java)**

Tyler Gaugler

Due Date: 9/13/2016

Algorithm Steps for this project:

Step 0: prepare the skeleton of your program, including classes, etc.  
  
Step 1:  inFile1 <-- open common word file  
             outFile1 <-- open outFile-1  
  
Step 2: CwordlistHead <-- make a new linked list by the list constructor where CwordlistHead points to a dummy node  
  
step 3:  commonWord <-- read a word from inFile1  
  
step 4:  spot <-- findSpot(CwordlistHead , commonWord) // see algorithm steps below  
  
step 5:  if spot != null // not duplicates  
                5.1: newNode <--  make a new node for  commonWord  
                5.2: call listInsert (Spot, newNode)  // you should know how to insert newNode after Spot  
  
step 6:  debugPrint(outFile1, CwordlistHead)  // output the common word linked list to outfile1  
step 7: repeat step 3 to step 6 until the inFile1 is empty  
  
step 8: inFile2 <-- open English text file  
Step  9: insertionCnt <-- 0  
Step 10: textListHead <-- make a new linked list by the list constructor where textListHead points to a dummy node  
step 11:  textWord <-- read a word from inFile2  
step 12:  if (isCommonWord (CwordlistHead, data) == false) // data is not a common word  
        12.1:  spot <-- findSpot(textListHead , textWord) // see algorithm steps below  
12.2: if spot == null  // duplicates  
                                Spot.count ++  
                        else  
                                newNode <--  make a new node for  textWord  
 listInsert (Spot, newNode)  // insert newNode after Spot  
insertionCnt++  
step 13: if insertionCnt >= 5  
         debugPrint(outFile1, CwordlistHead)  // print linked list every 5th insertion  
         insertionCnt <-- 0 //reset  
  
step 14: repeat step 11 to step 13 until the inFile2 is empty  
  
step 15: outFile2 <-- open output file2  
  
step 16: printSortedList (outFile2 , textListHead)

Source Code:

//project\_driver.java

import java.io.\*;

import java.util.Scanner;

public class project\_driver {

public static void main(String args[]){

linkedList commonWordList= new linkedList();

try{

Scanner commonWordFile = new Scanner(new FileReader(args[0]));

FileWriter fw1 = new FileWriter(args[2]);

while(commonWordFile.hasNext()){

String data= commonWordFile.next();

listNode spot= linkedList.findSpot(commonWordList,data);

if(spot.data.equals(data)){

spot.counter++;

}

else commonWordList.listInsert(spot, new listNode(data));

fw1.write("Insert Data("+ data +")\n");

linkedList.debugPrint(fw1,commonWordList.listHead); // commonWordList is printed after every insertion.

}

commonWordFile.close();

fw1.write("\n\n");

Scanner englishWordFile = new Scanner(new FileReader(args[1]));

int insertionCount=0;

linkedList englishWordList= new linkedList();

while(englishWordFile.hasNext()){

String data= englishWordFile.next();

if( !linkedList.isCommonWord(commonWordList.listHead,data)){

listNode spot= linkedList.findSpot(englishWordList, data);

if(spot.data.equals(data)){

spot.counter++;

}

else{

englishWordList.listInsert(spot, new listNode(data));

insertionCount++;

if(insertionCount >= 5){

fw1.write("Insert Data("+ data +")\n");

linkedList.debugPrint(fw1, englishWordList.listHead);

insertionCount=0;

}

}

}

}

fw1.close();

englishWordFile.close();

if(args[3] != null){

FileWriter fw2 = new FileWriter(args[3]);

linkedList.printList(fw2, englishWordList.listHead);

fw2.close();

}

}

catch(Exception e){

e.printStackTrace();

}

}

}

//linkedList.java

import java.io.FileWriter;

import java.io.IOException;

public class linkedList {

listNode listHead;

public linkedList(){

listHead= new listNode();

listHead.next= new listNode("dummy");

}

public listNode listHead(){

return listHead;

}

public void listInsert(listNode spot, listNode newNode){

newNode.next= spot.next;

spot.next= newNode;

}

public static listNode findSpot(linkedList l, String d){

listNode spot= l.listHead.next;

while(spot.next != null){

if(spot.next.data.compareTo(d) > 0){

return spot;

}

else if(spot.next.data.compareTo(d) == 0){

return spot.next;

}

else{

spot= spot.next;

}

}

return spot;

}

public static boolean isCommonWord(listNode listHead, String data ){

listNode current= listHead.next;

while(current.next != null){

if(current.data.equals(data)){

return true;

}

else current= current.next;

}

return false;

}

public static void debugPrint(FileWriter wr, listNode listHead) throws IOException{

listNode current= listHead.next;

wr.write("listHead-->");

while(current.next!= null){

wr.write("("+current.data+", "+current.next.data+ ")->");

current= current.next;

}

wr.write("("+current.data+",null)");

wr.write("\n");

}

public static void printList(FileWriter wr, listNode listHead) throws IOException{

listNode current = listHead.next;

while(current.next!= null){

wr.write(current.data+": "+current.counter+'\n');

current= current.next;

}

}

}

//listNode.java

public class listNode {

public listNode next;

public String data;

public int counter;

public listNode(String d){

data=d;

next=null;

counter=0;

}

public listNode(){

data="";

next=null;

counter=0;

}

}

Output

// output1

Adding: you

(dummy, you)-->(you, null)

Adding: i

(dummy, i)-->(i, you)-->(you, null)

Adding: he

(dummy, he)-->(he, i)-->(i, you)-->(you, null)

Adding: she

(dummy, he)-->(he, i)-->(i, she)-->(she, you)-->(you, null)

…….

…….

Adding: be

(dummy, a)-->(a, am)-->(am, an)-->(an, and)-->(and, are)-->(are, as)-->(as, be)-->(be, for)-->(for, he)-->(he, here)-->(here, him)-->(him, how)-->(how, i)-->(i, if)-->(if, in)-->(in, is)-->(is, it)-->(it, no)-->(no, not)-->(not, of)-->(of, on)-->(on, one)-->(one, or)-->(or, she)-->(she, that)-->(that, the)-->(the, them)-->(them, there)-->(there, they)-->(they, this)-->(this, was)-->(was, were)-->(were, what)-->(what, when)-->(when, where)-->(where, who)-->(who, why)-->(why, yes)-->(yes, you)-->(you, null)

Adding: our

(dummy, ago)-->(ago, four)-->(four, our)-->(our, score)-->(score, seven)-->(seven, years)-->(years, null)

Adding: nation

(dummy, ago)-->(ago, brought)-->(brought, continent)-->(continent, fathers)-->(fathers, forth)-->(forth, four)-->(four, nation)-->(nation, new)-->(new, our)-->(our, score)-->(score, seven)-->(seven, years)-->(years, null)

Adding: all

(dummy, ago)-->(ago, all)-->(all, brought)-->(brought, conceived)-->(conceived, continent)-->(continent, dedicated)-->(dedicated, fathers)-->(fathers, forth)-->(forth, four)-->(four, liberty)-->(liberty, nation)-->(nation, new)-->(new, our)-->(our, proposition)-->(proposition, score)-->(score, seven)-->(seven, to)-->(to, years)-->(years, null)

……..

……..

Adding: freedom

(dummy, above)-->(above, add)-->(add, advanced)-->(advanced, ago)-->(ago, all)-->(all, altogether)-->(altogether, any)-->(any, battlefield)-->(battlefield, before)-->(before, birth)-->(birth, brave)-->(brave, brought)-->(brought, but)-->(but, can)-->(can, cannot)-->(cannot, cause)-->(cause, civil)-->(civil, come)-->(come, conceived)-->(conceived, consecrate)-->(consecrate, consecrated)-->(consecrated, continent)-->(continent, created)-->(created, dead)-->(dead, dedicate)-->(dedicate, dedicated)-->(dedicated, detract)-->(detract, devotion)-->(devotion, did)-->(did, died)-->(died, do)-->(do, endure)-->(endure, engaged)-->(engaged, equal)-->(equal, far)-->(far, fathers)-->(fathers, field)-->(field, final)-->(final, fitting)-->(fitting, forget)-->(forget, forth)-->(forth, fought)-->(fought, four)-->(four, freedom)-->(freedom, from)-->(from, full)-->(full, gave)-->(gave, god)-->(god, great)-->(great, ground)-->(ground, hallow)-->(hallow, have)-->(have, highly)-->(highly, honored)-->(honored, increased)-->(increased, larger)-->(larger, last)-->(last, liberty)-->(liberty, little)-->(little, live)-->(live, lives)-->(lives, living)-->(living, long)-->(long, measure)-->(measure, men)-->(men, met)-->(met, might)-->(might, nation)-->(nation, never)-->(never, new)-->(new, nobly)-->(nobly, nor)-->(nor, note)-->(note, now)-->(now, our)-->(our, place)-->(place, poor)-->(poor, portion)-->(portion, power)-->(power, proper)-->(proper, proposition)-->(proposition, rather)-->(rather, remaining)-->(remaining, remember)-->(remember, resolve)-->(resolve, resting)-->(resting, say)-->(say, score)-->(score, sense)-->(sense, seven)-->(seven, shall)-->(shall, should)-->(should, so)-->(so, struggled)-->(struggled, take)-->(take, task)-->(task, testing)-->(testing, their)-->(their, these)-->(these, those)-->(those, thus)-->(thus, to)-->(to, under)-->(under, unfinished)-->(unfinished, us)-->(us, vain)-->(vain, war)-->(war, we)-->(we, whether)-->(whether, which)-->(which, will)-->(will, work)-->(work, world)-->(world, years)-->(years, null)

//output2

above: 0

add: 0

advanced: 0

ago: 0

all: 0

altogether: 0

any: 0

battlefield: 0

before: 0

birth: 0

brave: 0

brought: 0

but: 1

by: 0

can: 1

cannot: 2

cause: 0

civil: 0

come: 0

conceived: 1

consecrate: 0

consecrated: 0

continent: 0

created: 0

dead: 2

dedicate: 1

dedicated: 3

detract: 0

devotion: 1

did: 0

died: 0

do: 0

earth: 0

endure: 0

engaged: 0

equal: 0

far: 1

fathers: 0

field: 0

final: 0

fitting: 0

forget: 0

forth: 0

fought: 0

four: 0

freedom: 0

from: 1

full: 0

gave: 1

god: 0

government: 0

great: 2

ground: 0

hallow: 0

have: 4

highly: 0

honored: 0

increased: 0

larger: 0

last: 0

liberty: 0

little: 0

live: 0

lives: 0

living: 1

long: 1

measure: 0

men: 1

met: 0

might: 0

nation: 4

never: 0

new: 1

nobly: 0

nor: 0

note: 0

now: 0

our: 1

people: 2

perish: 0

place: 0

poor: 0

portion: 0

power: 0

proper: 0

proposition: 0

rather: 1

remaining: 0

remember: 0

resolve: 0

resting: 0

say: 0

score: 0

sense: 0

seven: 0

shall: 2

should: 0

so: 3

struggled: 0

take: 0

task: 0

testing: 0

their: 0

these: 1

those: 0

thus: 0

to: 7

under: 0

unfinished: 0

us: 2

vain: 0

war: 1

we: 9

whether: 0

which: 1

will: 0

work: 0

world: 0

Input

//input 1

you  
i he she  
are is  
was am were  
of and if or  
him a an one  
yes no  
on in that  
the they them who  
what where  
when how why  
it here there  
not this for  
as be

//input2

four score and seven years ago our fathers brought forth on this continent a new nation conceived  
in liberty and dedicated to the proposition that all men are created equal now we are engaged in a  
great civil war testing whether that nation or any nation so conceived and so dedicated can long  
endure we are met on a great battlefield of that war we have come to dedicate a portion of that  
field as a final resting place for those who here gave their lives so that nation might live it is  
altogether fitting and proper that we should do this but in a larger sense we cannot dedicate we  
cannot consecrate we cannot hallow this ground the brave men living and dead who struggled here  
have consecrated it far above our poor power to add or detract the world will little note nor long  
remember what we say here but it can never forget what they did here it is for us the living  
rather to be dedicated here to the unfinished work which they who fought here have thus far so  
nobly advanced it is rather for us to be here dedicated to the great task remaining before us that  
from these honored dead we take increased devotion to that cause for which they gave the last full  
measure of devotion that we here highly resolve that these dead shall not have died in vain that  
this nation under god shall have a new birth of freedom and that government of the people by the  
people for the people shall not perish from the earth