Project 1 - Linked List & Middle Node (Java)

Brandon Cheng
Section: 323.25

Due: 02/8/2023

IV. Main(...) **********Step 1: inFile open with args[0] outFile open with args[1]deBugFile open with args[2] Step 2: listHead get a new listNode with (“dummy”), as the dummy node for listHead to point to. Step 3: constructLL (listHead, inFile, deBugFile) Step 4: printList (listHead, outFile) // Print the complete list to outFileStep 5: middleNode findMiddleNode (listHead, deBugFile) Step 6: if middleNode != null // in case the list is empty outFile middleNode’s data // with caption “the word in the middle of list is”Step 7: Close all files

```
import java.io.*;
import java.util.*;



class listNode{
	public String data;
	public listNode next;

	public listNode(String data) { //listNode constructor
	this.data = data;
	this.next = null;
	}

}

class LList{
   String deBugFile;
   private BufferedWriter output;

	listNode listHead = new listNode("dummy");
	listNode middleNode  = new listNode(null);

	public void constructLL(listNode listHead, String inFile, String deBugFile){
	try {
		Scanner input = new Scanner(new FileReader(inFile));
		try {
			output = new BufferedWriter(new FileWriter(deBugFile));

	output.write("In constructLL method\n ");

			while(input.hasNext()){
		String data = input.next();
		listNode newNode = new listNode(data);
		listInsert(listHead, newNode, deBugFile);
		printList(listHead, deBugFile);
	}

			input.close();
			output.close();

		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	 } catch (FileNotFoundException e) {
```

```
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}

        public void listInsert(listNode listHead, listNode newNode, String deBugFile){
        try {
         output = new BufferedWriter(new FileWriter(deBugFile, true));
        output.write("\nIn listInsert method\n");

        listNode Spot = findSpot(listHead, newNode);
        newNode.next = Spot.next;
        Spot.next = newNode;
        output.write("Returns from findSpot where Spot.data is " + Spot.data + "\n");
        output.close();
        } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        }
        }

        public listNode findSpot(listNode listHead, listNode newNode){
        while(listHead.next != null && newNode.data.compareToIgnoreCase(listHead.next.data)>0){
        listHead = listHead.next;
        }
        return listHead;
        }

        public void printList(listNode listHead, String outFile){
         int count = 0;
        try {
         output = new BufferedWriter(new FileWriter(outFile, true));
        output.write("listHead ->");

        while(listHead != null){
        if (listHead.next != null) {
                output.write("(" + listHead.data + ", " + listHead.next.data + ") -> ");
        } else {
                output.write("null\n");
        }
        listHead = listHead.next;
        count++;
                        if(count >= 5) {
                                output.write("\n");
                                output.write("\n");
                                count = 0;
                        }
        }
        output.close();
        } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        }
        }
```

```
	public listNode findMiddleNode(listNode listHead, String deBugFile) throws IOException{
	output = new BufferedWriter(new FileWriter(deBugFile, true));
	output.write("In findMiddleNode method "); //debug

	listNode walker1 = listHead.next;
	listNode walker2 = listHead.next;
	while(walker2 != null && walker2.next !=null){
		walker1 = walker1.next;
		walker2 = walker2.next.next;
		output.write("\n walker1's data is " + walker1.data + " ");

	}
	output.close();
	return walker1;

	}
}

public class ChengB_Project1_Main {
	public static void main(String[] args) throws IOException{

	String inFile = new String(args[0]);
	 String outFile = new String(args[1]);
	String deBugFile = new String(args[2]);

	LList linkedList = new LList();
	listNode listHead = new listNode("dummy");

	linkedList.constructLL(listHead, inFile, deBugFile);
	linkedList.printList(listHead, outFile);
	linkedList.findMiddleNode(listHead, deBugFile);

	}
}
```

OUTPUTS

My debug file is more than 10 pages long.

Here is my output.txt file

listHead ->(dummy, 84) -> (84, a) -> (a, a) -> (a, a) -> (a, about) ->

(about, aging) -> (aging, American) -> (American, an) -> (an, and) -> (and, and) ->

(and, and) -> (and, and) -> (and, and) -> (and, apprentice) -> (apprentice, as) ->

(as, baseball) -> (baseball, battle) -> (battle, been) -> (been, been) -> (been, being) ->

(being, between) -> (between, boy) -> (boy, by) -> (by, catching) -> (catching, confident) ->

(confident, Cuba) -> (Cuba, day) -> (day, days) -> (days, each) -> (each, end) ->

(end, experienced) -> (experienced, far) -> (far, favorite) -> (favorite, fish) -> (fish, fish) ->

(fish, fish) -> (fish, fisherman) -> (fisherman, fishermen) -> (fishermen, fishing) -> (fishing, Florida) ->

(Florida, food) -> (food, forbidden) -> (forbidden, form) -> (form, gear) -> (gear, gone) ->

(gone, Gulf) -> (Gulf, has) -> (has, has) -> (has, hauling) -> (hauling, having) ->

(having, he) -> (he, He) -> (He, him) -> (him, his) -> (his, his) ->

(his, his) -> (his, his) -> (his, his) -> (his, in) -> (in, instead) ->

(instead, into) -> (into, is) -> (is, is) -> (is, its) -> (its, large) ->

(large, Man) -> (Man, Manolin) -> (Manolin, Manolin) -> (Manolin, marlin) -> (marlin, near) ->

(near, next) -> (next, night) -> (night, north) -> (north, now) -> (now, of) ->

(of, of) -> (of, of) -> (of, of) -> (of, Old) -> (Old, on) ->

(on, opens) -> (opens, out) -> (out, parents) -> (parents, player) -> (player, preparing) ->

(preparing, sail) -> (sail, salao) -> (salao, Santiago) -> (Santiago, Santiago) -> (Santiago, Santiago) ->

(Santiago, Santiagos) -> (Santiagos, Sea) -> (Sea, seen) -> (seen, shack) -> (shack, so) ->

(so, story) -> (story, story) -> (story, Straits) -> (Straits, streak) -> (streak, Stream) ->

(Stream, successful) -> (successful, talking) -> (talking, tells) -> (tells, tells) -> (tells, that) ->

(that, that) -> (that, that) -> (that, the) -> (the, the) -> (the, the) ->

(the, The) -> (The, the) -> (the, The) -> (The, the) -> (the, the) ->

(the, The) -> (The, to) -> (to, to) -> (to, to) -> (to, told) ->

(told, unluckiness) -> (unluckiness, unlucky) -> (unlucky, unlucky) -> (unlucky, venture) -> (venture, visits) ->

(visits, will) -> (will, with) -> (with, with) -> (with, with) -> (with, without) ->

(without, worst) -> (worst, young) -> null