Lee Moriarity

Professor Kelly

CIS 112 ONL01

11 September 2022

Lab #2

There are 2 parts to this LAB:

PART 1: Complete exercises #20 (stack, p. 151) and #40 (linked list, p. 156) .  Put your answers in a Word document.

PART 2: Do Programming Exercise #30 (EditString, p. 154) at the end of Chapter 2.

HINT: You might want to push the entire string on the stack (a stack of Strings) so when you want to undo you can just get rid of the last thing on stack.  The Reverse could use another stack but one of characters or just use a StringBuilder.

NOTE: All labs must use the author's code that I gave you  ("bookfiles" folder) .  So here for this exercise do not use Java's Stack class but rather the author's stack classes.  You can just copy the needed bookfiles to your programming folder and comment out the package and import lines if you want to do the development that way.

After you compile and run your program successfully (testing everything), submit your java file(s) and your Word document on Blackboard for credit.

Part 1

Exercise #20:

20. Show what is written by the following segments of code, given that item1, item2,

and item3 are int variables, and s is an object that is the abstract description of

a stack as given in the section. Assume that you can store and retrieve variables of

type int on s.

1. item1 = 1; item2 = 0; item3 = 4; // Assign integer values to item 1, 2 and 3.

s.push(item2); s.push(item1); s.push(item1 + item3); // s = [0, 1, 5]

item2 = s.top(); // item2 = 5

s.push (item3\*item3); s.push(item2); s.push(3); // s = [0, 1, 5, 16, 5, 3]

item1 = s.top(); // item1 = 3

s.pop(); // s = [0, 1, 5, 16, 5]

System.out.println(item1 + " " + item2 + " " + item3); // print 3 5 4

while (!s.isEmpty()) // while s is not empty (s = [0, 1, 5, 16, 5])

{ // item1 = 5; print 5

item1 = s.top(); s.pop(); // s = [0, 1, 5, 16]

System.out.println(item1); // item1 = 16; print 16

{ // s = [0, 1, 5]

// item1 = 5; print 5

// s = [0, 1]

// item1 = 1; print 1

// s = [0]

// item1 = 0; print 0

// s is empty

1. item1 = 4; item3 = 0; item2 = item1 + 1; // Assign integer values to item 1, 3, and 2

s.push(item2); s.push(item2 + 1); s.push(item1); // s = [5, 6, 4]

item2 = s.top(); s.pop(); // item2 = 4; s = [5,6]

item1 = item2 + 1; // item1 = 5

s.push(item1); s.push(item3); // s = [5, 6, 5, 0]

while (!s.isEmpty()) // while s is not empty ( s = [5, 6, 5, 0])

{ // item3 = 0; print 0

item3 = s.top(); s.pop(); // s = [5, 6 ,5]

System.out.println(item3); // item3 = 5; print 5

} // s = [5, 6]

// item3 = 6; print 6

// s = [5]

// item3 = 5; print 5

// s is empty

System.out.println(item1 + " " + item2 + " " + item3); // print 5 4 5

Exercise #40:

40. We developed Java code for traversing a linked list. Here are several alternate, possibly flawed, approaches for using a traversal to print the contents of the linked list of

strings accessed through letters. Critique each of them:

1. LLNode<String> currNode = letters; // This is the most effective way of travesing

while (currNode != null) // a linked list. It will print out the correct

{ // value and then move to the next item in

System.out.println(currNode.getInfo()); // the linked list until it reaches the end.

currNode = currNode.getLink();

}

1. LLNode<String> currNode = letters; // This code will not work as it is supposed

while (currNode != null) // to. Since it is defining currNode before

{ // printing, it will skip the first value and

currNode = currNode.getLink(); // also cause a NullPointerException error at

System.out.println(currNode.getInfo()); // the end of the loop.

}

1. LLNode<String> currNode = letters; // This is also a working code like the code

while (currNode != null) // in part a, except that it has an unnecessary

{ // if-else statement that is not needed.

System.out.println(currNode.getInfo());

if (currNode.getLink() != null)

currNode = currNode.getLink();

else

currNode = null;

}