CS 3305: Data Structures

Assignment 03 – Stacks Check Due Date in D2l (100 points total)

Note: Never hard-code test data in the test program, unless explicitly stated otherwise in the assignment. Always allow the user to enter the test data using a menu option.

GENERAL SUBMISSION REQUIREMENTS

Upload all files individually as specified, not as zip files, to Assignments in D2L. Do not email files.

Make sure your program compiles, runs and produces the correct output.

Ensure you have the correct file name(s), and author header, as specified in the Assignment.

Always use meaningful labels for prompts, inputs, and outputs.

Always use comments, indentation and whitespace as shown in examples.

Objectives The purpose of this lab is to reinforce stack concepts in Java

Important Note: Code must be running and produce results before being uploaded.

Assignment 03 - PART 1 Stacks (50 points) - Note Part 1 is a separate deliverable:

In some other applications, we might need two stacks with the same type of data. If we implement the stacks as arrays, there is a chance that one array (and hence one stack) becomes filled, causing our computation to end prematurely. This might be a shame, since the other array (stack) might have plenty of room. One way around this problem is to **implement two stacks as one large array** rather

than two smaller arrays.

YOU MUST WRITE YOUR OWN Stack Class For Part 1 – Review Section 10.6 of the text for designing a stack class.

Requirements

No files are provided with any part of this Assignment. You may hard code your test data.

Assume you need two stacks with the same type of data. If you implement the stacks as arrays, there is a chance that one array (and hence one stack) will become filled, causing our computation to end prematurely. This might be a shame since the other array (stack) might have plenty of room. One way around this problem is to implement two stacks as one large array rather than two smaller arrays (stacks).

Write a class for a pair of stacks. A pair of stacks is simply an object with two stacks. Call these stacks StackA and StackB. You need separate methods for each stack:

pop_a and pop_b

push_a and push_b

is_empty_a and is_empty_b

is_full // you only need one *is_full* because if there's

    // room in the array there's room for either a or b

Variables will be needed to keep track of the top of each stack, call them top_a and top_b

Implement the stack pair as a single array. The two stacks grow from the two ends of the array, so for example, one stack could fill up one quarter of the array, while the other fills up three quarters.

Use *int* for the underlying type of the stack data.

If the array runs out of room, print a statement to the screen that the array is full and end the program.

Testing

In order to test your stack pair, write the main method in the class, that will push a few values onto the StackA , then push a few values onto StackB . The program should pop the values on StackA and check that they are correct as they are popped. The program should pop the values on StackB and check that they are correct as they are popped.

Part 1 requires user input. Input data for StackA, and then some for StackB and then print both. Then delete something from each stack and print again.

HINT:

1. You can write all of this in one class file (variable declarations, constructors, methods, including main method)
2. Methods are contained in a class. To run a Java program, the program must have a main method (see method header below).

public static void main(String args[])

The main method is the entry point where the program starts when it is executed.

For this exercise, add your main method at the bottom of your class file.

3. Make sure you implement ALL of the methods specified.
4. Make sure you show output from calling all of the methods specified. You can use a print statement in a method saying, "Results from calling ______ method:" You can probably get several calls in one output screenshot
5. Review Chapter 1 Summary, Liang p. 28

Do not forget to include author header in each submitted file as shown, no header, no points!

```
// Name: <your name>
// Class: CS 3305/ put your section number after the /
// Term: Fall YYYY
// Instructor: Sharon Perry
// Assignment: 3-Part-1-Stacks
```

DELIVERABLE INSTRUCTIONS – Part 1

Capture a **READABLE** screenshot(s) of your program output and paste into a word/pdf document. Readable means readable! Screenshots ***should not be an entire desktop*** – use some type of snipping tool. After your output screenshots, copy and paste the source code for your program into the word/pdf doc. Save doc as a file named LastName-A3-Part-1-Stacks.docx or .pdf. word. Last step is to upload everything to D2L, to clarify upload

1. word/pdf doc that contains output screenshot and copy/pasted source code; AND
2. your .java file that contains your source code.

Only your latest submission is kept so if you update a file and want to resubmit you must upload all files. Submit everything to the assignment submission folder in D2L by the due date posted in D2L.

No zip file or email submissions are accepted. Late penalties are in effect for this assignment.

MAKE SURE YOUR CODE HAS COMMENTS ! We are getting submissions without comments in the code. No comments in source code = (-20) points *per each Part of the assignment.* This penalty grows as time goes on !!

Late penalties of 10 % per day are in effect for this assignment.

Assignment 03 – PART 2 Palindromes (50 points) :

In this part, you will use stacks to recognize palindromes. Palindromes are strings that read the same backward as forward (for example "madam"). Write a program to read a line in from the keyboard and print to the display, whether or not it is a palindrome. You MUST use three stacks to implement the program. For simplicity use only lower case letters in your test strings. You may use the stack class from Java for Part 2.

Use the following logic

1. Push original string onto Stack_1 and also Stack_2
2. Pop Stack_1 and push onto Stack_3 until Stack_1 is empty
3. At this point Stack_2 is the original string and Stack_3 is the reverse
4. Compare Stack_2 and Stack_3 using dot =
5. If Stack_2 is equal to Stack_3 the string is a palindrome

Test your string by entering data from the keyboard and writing out to the screen whether the string is a palindrome.

Do not forget to include author header in each submitted file as shown, no header, no points!

```
// Name: <your name>
// Class: CS 3305/ put your section number after the /
// Term: Fall YYYY
// Instructor: Sharon Perry
// Assignment: 3-Part-2-Palindromes
```

DELIVERABLE INSTRUCTIONS – Part 2

Capture a **READABLE** screenshot(s) of your program output and paste into a word/pdf document. Readable means readable! Screenshots ***should not be an entire desktop*** – use some type of snipping tool. After your output screenshots, copy and paste the source code for your program into the word/pdf doc. Save doc as a file named LastName-A3-Part-2-Palindromes.docx or .pdf. word. Last step is to upload everything to D2L, to clarify upload

1. word/pdf doc that contains output screenshot and copy/pasted source code; AND
2. your .java file that contains your source code.

Only your latest submission is kept so if you update a file and want to resubmit you must upload all files. Submit everything to the assignment submission folder in D2L by the due date posted in D2L.

No zip file or email submissions are accepted. Late penalties are in effect for this assignment.

MAKE SURE YOUR CODE HAS COMMENTS ! We are getting submissions without comments in the code. No comments in source code = (-20) points *per each Part of the assignment.* This penalty grows as time goes on !!

Late penalties of 10 % per day are in effect for this assignment.