CSCI-101 Programming 1

Lab 11, Part a - Week of Nov 13

INSTRUCTIONS

Inside the labs directory create a file named **Lab11a.java** and a file named **IntegerStack.java**.

Add the following to the **IntegerStack** class:

1. A private field named **stack** that holds an array of integers.

2. A private field named **size** that holds an integer which indicates how many elements are currently stored in the stack and indicates at which index to add the next element.

3. A constructor that takes an integer as an argument and initializes the **stack** field with an array of integers whose length is equal to the value passed into the constructor.

4. A method named **push** which has an integer parameter named elm. If there is room in the stack to store the element, the element is stored at the lowest possible index and **size** is incremented.

5. A method named **pop**. If **size** is greater than zero, the last element added to the stack is returned and **size** is decremented. If **size** is zero, a **NoSuchElementException** is thrown.

6. A method named **size** that returns the number of elements currently stored in the stack.

7. The Object class' **toString** method is overridden. The string returned from the method should contain an opening parenthesis, followed by a comma separated list of the elements stored in the stack, followed by a closing parenthesis.

8. The Object class' **equals** method is overridden. Two stacks are considered equal if they store the same number of elements and have the same exact elements stored in the same order.

In the **main** method in **Lab10a.java** do the following.

a. Create an instance of the **IntegerStack** class that can hold 10 elements.
b. Ask the user to enter 5 integers, read in the values, and store them in the stack.
c. Use the **toString** method to print the contents of the stack to the screen.
d. Print the number of elements that are stored in the stack using the **size** method.

e. Create a second instance of the **IntegerStack** class that can hold 10 elements.
f. Ask the user to enter 6 integers, read in the values, and store them in the second stack.
g. Use the **toString** method to print the contents of the second stack to the screen.
h. Print the number of elements that are stored in the second stack using the **size** method.
i. Remove the last element added to the second stack using the **pop** method.
j. Use the **toString** method to print the contents of the second stack to the screen.

k. Use the **equals** method to determine if the two stacks are equal. If they are equal, print "Stacks are equal" to the screen, otherwise print "Stacks are not equal".

Run the following tests.

Test 1

Input:

User input: 1,2,3,4,5

User input: 1,2,3,4,5,6

Output:

(1,2,3,4,5)

(1,2,3,4,5,6)

(1,2,3,4,5)

Stacks are equal

Test 2

Input:

User input: 1,2,3,4,5

User input: 1,2,3,4,4,6

Output:

(1,2,3,4,5)

(1,2,3,4,4,6)

(1,2,3,4,4)

Stacks are not equal