**Home Assignment - 7  
Due Date: 11/16/2018, 11.59 pm**

**Total Points: 100**

**End of Chapter 8 Questions**

8.10.1 – Multiple Choice Exercises – 3, 5, 7, 12 (1 point each: 1\* 4 = 4 points)

8.10.2 – Reading and Understanding Code – 18, 20, 22, 24 (2 points each: 2 \* 4 = 8 points)

8.10.3 – Fill in the Code – 28, 30, 32, 35 (3 points each: 3 \* 4 = 12 points)

8.10.4 – Identifying errors in Code – 36, 39, 41, 43 (2 points each: 2 \* 4 = 8 points)

**End of Chapter 9 Questions**

9.10.1 – Multiple Choice Exercises – 5, 6, 7, 8, 11 (1 points each: 1 \* 5 = 5 points)

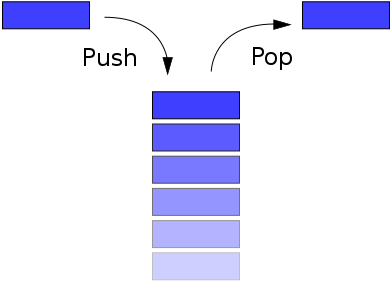
9.10.2 – Reading and Understanding Code – 20, 22, 23, 25, 29   
 (2 points each: 2 \* 5 = 10 points)

9.10.3 – Fill in the Code – 36, 39, 40 (3 points each: 3 \* 3 = 9 points)

**Programming Exercise (44 points):**

Stack is a container of elements that are inserted and removed according to the Last-in-First-out (LIFO) principle – the last element that goes into the stack is the first one that comes out like a stack of books. The last book added to the stack is the first one that comes out of the stack. The three operations that a stack structure must support are

1. Push operation – inserting an item to the stack
2. Pop operation – deleting an item from the stack
3. Top operation – that returns the last item pushed on to the stack.



(Courtesy: https://upload.wikimedia.org/wikipedia/commons/2/29/Data\_stack.svg)

You will implement a user-defined class called ***Stack*** with the following:

***Instance Variables***

* An int array called list to act as a container of the elements of the stack
* An int value count that will keep count of elements in the stack. (Please remember an array can be partially filled.

***Constructors and Methods***

* A default constructor that initializes the list instance variable to an array of length 5
* An overloaded constructor that takes an int array as a parameter, and an int value that provides a count of the elements in the array that is being passed as a parameter.
* Get method – size( ) to return the count of elements in the stack.
* push() method that takes an int value as a parameter which will be added to the stack. Please note this method would have to first check if the array is full by calling the isFull ( ) method before adding the element to the stack. If the array is full then the method will call a helper method (private method) addCapacity( ) that will increase the capacity of the list to double the original size and copy the values from the original array.
* pop() method that will remove the element from the top of the stack and return that element.
* top( ) method that returns the element at the top of the stack
* isEmpty ( ) method that checks if the stack is empty
* isFull ( ) method that checks if the stack is full
* Method maxValue( ) that returns the maximum value that is currently stored in the stack.
* Get method – getList( ) to return a copy of list. Note this method should return a copy of the list with elements copied from the original list to preserve encapsulation.
* toString( ) method that returns the contents of the stack as a String.
* equals( ) method that checks if the contents of two Stack objects are the same.

Next implement a test program, StackTest with the main( ) method and do the following in the test program

1. Create an int array with length 10 and store 10 random int values in the array between 1 and 100. Make use of the Random class to generate these random values.
2. Create a Stack object using the overloaded constructor and pass the array you created in the first step.
3. Print the contents of the stack.
4. Add the following values to the stack by calling the push method: 32, 5, 73, 82, 19
5. Print the contents of the stack.
6. Print the maximum value in the stack
7. Perform a pop operation on the stack and print the value the pop method returns
8. Print the contents of the stack.
9. Call the top( ) method to print the element at the top of the stack
10. Perform another pop operation on the stack and print the value the pop method returns
11. Call the top( ) method to print the element at the top of the stack
12. Print the contents of the stack.
13. Perform one more pop operation on the stack and print the value the pop method returns
14. Call the top( ) method to print the element at the top of the stack
15. Print the contents of the stack.

**Things to submit:**

* In a word document type your name and assignment number
* Type answers to each of the end of chapter questions. For code, change the font face to Courier New.
* Copy and paste the source code of Stack.java and StackTest.java
* Copy and paste the output of one run of the StackTest program.
* Copy and paste the screen shot of the output window.
* Submit the word file along with the zipped NetBeans folder using the link in Blackboard.