### Portfolio Function Title: Stack

Version number: 1

### Function Description:

Allows for a user to build a stack of objects. Once a stack is built a user can push (add), pop (remove) and peek (get) on the top of the stack. A user can also find the size of the stack (GetSize) and check whether it is empty (CheckIfEmpty).

### Author:

Nathan Mankovich

### Date Written/Last Modified:

04-27-2015 / 04-30-2015

### How to use the Function:

To use the stack program just instantiate a Stack<obj> with an object of your choice and then you can use the different methods in your java code.

The program is stored in the portfolio.jar file, put that file in the same folder as YourProgram.java.

Open a Windows command window and enter:

To compile the program with YourProgram.java enter:

javac -cp .;portfolio.jar YourProgram.java

pause

Then, to run the program with the provided sample-mailing-list.csv enter:

java -cp .;portfolio.jar YourProgram

pause

User Interface: Java code and command line only.

### How the Function works:

Each method in Stack.java calls the LinkedList program to fill a singly linked list that was instantiated at the beginning of Stack.java.

The Stack is populated in the push method. Objects are removed from the top and returned using the pop method. The peek method gets the object at the top without removing it. The CheckIfEmpty method checks if the stack is empty and returns a true Boolean if it is, otherwise the method returns a false boolean. The GetSize method returns the size of the stack as an integer.

### Supported Methods (including Inputs, Outputs, Features and Results by method):

* push: This pushes an object on the top of the stack using the AddToFront method from LinkedList.
  + Input: object
  + Output: none
* pop: This pops an object off the top of the stack using the RemoveFirst method from LinkedList.
  + Input: none
  + Output: removed object
* Peek: This uses the GetFirst method from LinkedList to return the first element in the stack.
  + Input: none
  + Output: the first object
* CheckIfEmpty: This uses the CheckIfEmpty method from linked list to check if the stack is empty.
  + Input: none
  + Output: a Boolean, true if empty
* GetSize: This uses the GetLength method from linked list to get the size of the stack.
  + Input: none
  + Output: an integer that is the length of the stack

### Known problems and limitations:

If the pop or peek methods are used on an empty stack, the program will return a NoSuchElementException.

A stack is a limited data structure because a user cannot take elements off the bottom of the stack, nor can they index certain elements in order to look at them. It is a very basic data structure that can be useful to store large amounts of ordered data that will output elements in the reverse order than their input order (last pushed pops first).