Programming Project 4 – Custom Stack Implementation

Note: When you turn in an assignment to be graded in this class, you are making the claim that you neither gave nor received assistance on the work you turned in (except, of course, assistance from your instructor). Any assistance from other sources, including the internet, is an honor code violation.

In this project you will implement a Stack.

1. You will write a `CustomStack<E>` class that implements the `StackInterface<E>` shown here:

```
package cmsc256;

public interface StackInterface<E> {
   /** Adds a new entry to the top of this stack.
       @param newEntry  An object to be added to the stack. */
   public void push(E newEntry);

   /** Removes and returns this stack's top entry.
       @return  The object at the top of the stack.
       @throws  EmptyStackException if the stack is empty before the operation.
*/
   public E pop();

   /** Retrieves this stack's top entry.
       @return  The object at the top of the stack.
       @throws  EmptyStackException if the stack is empty. */
   public E peek();

   /** Detects whether this stack is empty.
       @return  True if the stack is empty. */
   public boolean isEmpty();

   /** Removes all entries from this stack. */
   public void clear();
}
```

2. The class will use the BRIDGES `Slelement<E>` class to represent an element in the stack. The SLelement class represents a singly-linked node. More information and a sample program can be found here - http://bridgesuncc.github.io/tutorials/SinglyLinkedList.html

3. To view the contents of the stack you will need a `display()` method within the `CustomStack<E>` class that will provide an output of the stack. The following code may be used:

```
public void display() {
    if(isEmpty()) {
        System.out.println("The stack is empty");
    }
    else {
        SLelement<T> current = topNode;
        StringBuffer output = new StringBuffer();
        output.append("Top of stack: " + current.getValue() + "\n");

        while(current.getNext() != null) {
            current = current.getNext();

            if(current.getNext() == null)
                output.append("Stack bottom: ");
            else
                output.append("              ");

            output.append(current.getValue() + "\n");
        }
        System.out.println(output.toString());
    }
}
```

4. Add a main method to demonstrate your implementation on a stack of integer by performing a sequence of push and pops and output the stack contents. Here is example:

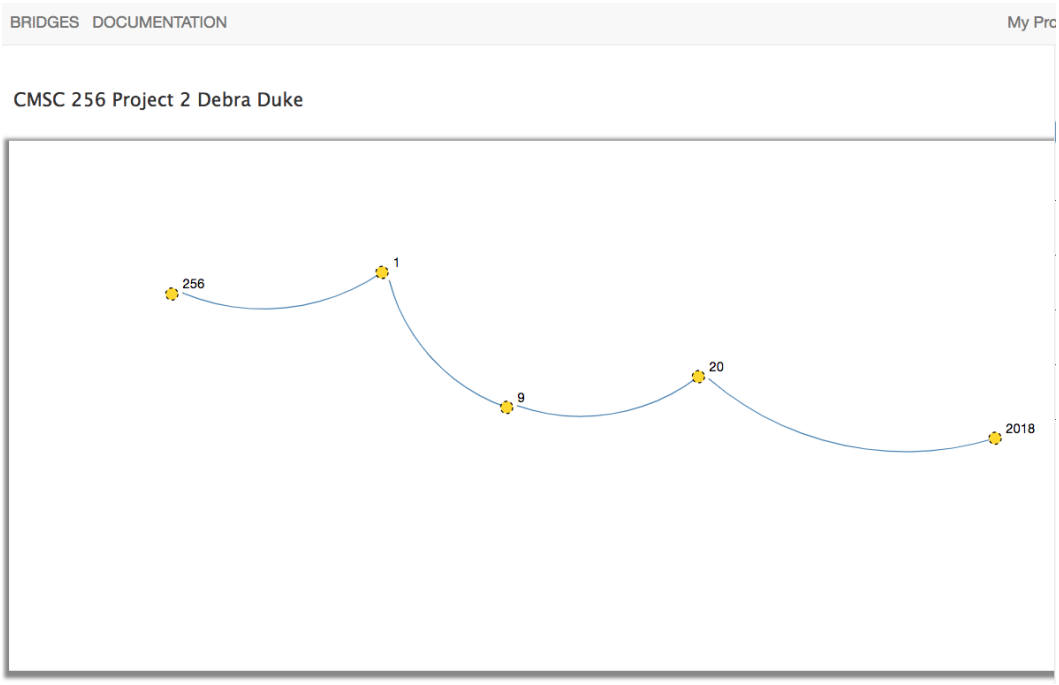
```
Pushed 256, 10, 18, 20, and 2018
Top of stack: 2018
              20
              18
              10
Stack bottom: 256

Called pop twice:
Top of stack: 18
              10
Stack bottom: 256

A call to peek returns 18
Top of stack: 18
              10
Stack bottom: 256

Called pop three times:
The stack is empty
```

5. Use the BRIDGES visualizer to display the contents of the full stack, before any elements are popped off.



Your file must be written in a specific package – **`cmsc256`** and will be uploaded to Gradescope for grading. Follow the Coding Style Guideline in Blackboard and include a comment block that includes your name, the name of the project along with the file name and a brief description of the purpose of the class at the top of **the source code file that you submit**.

Ask questions about any part of the programming project that is not clear!