Project #3

John P. Baugh, Ph.D. – CIS 2353 – Oakland Community College – OR

Points: _____ / 125

Objectives

- To learn about stacks
- To implement a stack-like solution
- Naming the package **proj3**

Instructions

Design a stack-like data structure **using a linked chain** of nodes that you build yourself (NOT an array, NOT any pre-built data structures) that allows for **increment** and **decrement** operations. The data for these nodes should just be an **integer**.

You may refer to the linked based implementation in the book for implementation ideas. Note that this is stack-like. As far as inserting and removing elements from the stack, it behaves like a normal stack (push adds an element, pop removes an element). However, you are allowing for additional behavior that modifies elements in the stack.

You must name the class **ChangeStack**.

The required methods are as follows:

- ChangeStack
  - Constructor
  - Initializes the **numElements** to 0 and the **top** of the stack to null
- push
  - Should be a void method
  - Should take one parameter, an integer *item* representing the element being added to the stack
  - Should add the new integer to the stack at the top
- pop
  - Should return an integer
  - Should take no parameters
  - If the user tries to pop on an empty stack, you should **throw an exception, StackEmptyException**, which is a **custom Exception** class that you must implement
    - It must extend RuntimeException
  - If successful, should remove the top of the stack and return the integer data value in the top node

- peekTop
    - Should return an integer – the value at the top
    - Should take no parameters
    - If the user tries to pop on an empty stack, you should **throw an exception, StackEmptyException**, which is a **custom Exception** class that you must implement
        - It must extend RuntimeException
    - Unlike pop, it should not remove the element at the top
- increaseValues
    - Should be a void method
    - Takes two parameters: an integer *k* and an integer *amount*
    - Should increase the bottom *k* elements in the stack by the *amount*
    - If there are less than k elements in the stack, increment all the items in the stack by the given amount
    - If there are no items in the stack, do nothing
- decreaseValues
    - Should be a void method
    - Same parameters as increaseValues
    - Should decrease the bottom *k* elements in the stack by the *amount*

Deliverables

- **Create a zip file** of your .java files, as well as the screen shots of your program working (screenshots should be placed inside a .docx or .pdf file) and turn in the zip file. Name the zip file "Project3" and D2L will take care of putting your name in it.

- You will also need **screen shots of your program working**, pasted inside of a PDF or Word (.doc or .docx) document (you can create PDF from Word documents using the Save As… option)

- Also, make sure your name is in comments on **each** Java file that you turn in. For example:

```
// Fanny Hertz
// Project 3
// CIS 2353
// Fall 2023
// Prof. John P. Baugh
```