**CS 1450 Data Structures and Algorithms – Fall 2021**

**Assignment #8**

Due Date: **Nov 3, 2021** at 1:40pm (MW class), **Nov 4, 2021** at 12:15pm (TR classes)

Purpose: Using array lists, queues, a stack, and iterators.

Effort: Individual

Points: **100**

Deliverables: Upload a **.zip** file with **ONLY** your source code (.java file) to Canvas by due date.

**Copy your code and design notes into your OneNote notebook.**

**The Situation**

I’m not sure where this started or really why but each semester parrots become involved in the assignments. It’s a tradition now so obviously it must continue! In Fall 2019, the parrots were rescued from captivity in Norway, and started their long-awaited concert tour singing “***free bird”*** to all their fans. In Spring 2020, the parrots went missing but thankfully they were found, and of all places, on a cruise ship which was very unfortunate during Spring 2020 for the parrots! In Fall 2020, the parrots had grown quite tired of their trans-Atlantic voyage and finally made their way through the panama canal and rented a place in Panama City. After some rest and relaxation, the parrots headed to Tokyo in Spring 2021 for the Olympics. So for Fall 2021, the parrots pick up in Tokyo and are coordinating a new adventure. A coded message is being sent out to all parrots and your mission, if you choice to accept it, is to decode the message.

**Assignment Description**

This assignment provides the opportunity to work with array lists, queues, a stack, and iterators. You will write a program to translate a secret message that has been split into 3 parts. These parts are provided in 3 files:

* File #1 – contains one part of the secret message as characters
* File #2 – contains another part of the secret message as ascii codes
* File #3 – contains the decoding keys, 0’s and1’s, one key for each character in original message.

The completed program contains the following classes:

* 1 Secret Translator
* 1 Stack

**Specifications**

1. Create a Java class called **LastNameFirstNameAssignment8**
2. Follow “CS1450 Programming Assignments Policy”
3. Write a test program to do the following:
   1. Translate the 1st message using Arraylists
      1. Create 3 ArrayLists
         * Each ArrayList will hold the contents of one file for the 1st secret message
         * arrayMessage1.txt, arrayMessage2.txt, arrayKey.txt
      2. For arrayMessage1.txt
         * Read and place each character in message into an ArrayList of characters
      3. For arrayMessage2.txt
         * Read and place each integer in message into an ArrayList of integers
      4. For arrayKey.txt
         * Read and place each integers (i.e. 0 or 1) into an ArrayList of integers
         * The key is used during decoding, see decode method below for details.
      5. Create **3 iterators** - one for each ArrayList (see lecture #12 for iterator information)
         * Note the iterator must be created after the values have been placed into the array lists
      6. Create an instance of the secret translator class: **SecretTranslator**
      7. Call secret translator’s ***decode*** method sending in 3 iterators to decode the message
      8. Display the decoded message
4. **REPEAT steps (i – viii)** using **Queues** to hold the contents for each file for the 2nd secret message:
5. queueMessage1.txt, queeuMessage2.txt, and queueKey.txt

**Test File Information**

There is a total of 6 files – 3 files for each secret message:

* File #1 – contains one part of the secret message as characters
* File #2 – contains another part of the secret message as ascii values (integers)
* File #3 – contains the decoding keys, 0’s and 1’s, one key for each character in the message

**Secret Message#1 Files**

* arrayMessage1.txt dero-ralsnrT
* arrayMessage2.txt 33 107 119 111 116 97 ascii codes
* arrayKey.txt 1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1 0 0 decoding keys

**Secret Message #2 Files**

* queueMessage1.txt !nJ-iseruevdofyer-at-h
* queueMessage2.txt 97 112 97 110 45 116 110 97 45 114 45 100 97 101 114 45 115 111 114

114 97 112 101 84

* queueKey.txt 0 0 1 1 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 0 1 0 1 1 0 0 0 1 1 0 1 1 0 1 1 1

1 1 0 1 0 1

**Note:** The files used for grading will NOT be the same as the files that you are given.

**Classes**

**SecretTranslator**

* Description
  + Class that translates secret messages
* Private Data Fields
  + **stack** – used during translation of the secret message
* Public/Private Methods
  + Constructor

***public SecretTranslator()***

* + - Allocate memory for stack
  + No getters/setters because the stack is used only inside this class
  + ***public String decode (Iterator<Character> msg1Iterator,***

***Iterator<Character> meg2Iterator,***

***Iterator<Integer>keyIterator)***

* + - Decodes the secret message using a two-step process:
    - **Step 1:** iterate through the values in the **keyIterator** and determine if value from **msg1Iterator** or **msg2Iterator** will be obtained
      * For each key in the keyIterator
        + If key is a 0:

get next character from **msg1Iterator** and

push onto the stack

* + - * + If key is a 1:

get next integer from **msg2**I**terator**,

convert to a character

push onto the stack

* + - **Step 2**: Undo the reversal
      * The characters of the secret message are now all on the stack.
      * Re-create the original message by removing one character at a time from the stack and placing the character into a return string
      * This undoes the reversing of the secret message.
      * Stacks are perfect for reversing information or undoing a reversal!

**Stack Class**

* Description
  + A stack is used in the final step to unscramble the secret message.
  + The secret message is pushed onto the stack one character at a time as described in the Secret Translator’s ***decode*** method. In the final step, you pop the characters off the stack and place them into a string. This will undo the reversing of the original string.
* Private Data fields
  + **list** – Use an ArrayList to serve as the storage container for the characters
    - To implement a stack, we only add and remove from **one end** of the array.
    - Make the array’s last occupied location represent the **top** of the stack.
    - The **last element** added to the array list is the **1st element** removed
* Public Methods
  + Constructor

***public Stack()***

* Allocated memory for array list
  + public method ***isEmpty()***
    - Returns a boolean indicating if the stack is empty
  + public method ***getSize()***
    - Returns the number of elements currently on the stack
  + public method ***push(Character value)***
    - Adds a character to the top of the stack
  + public method **pop()**
    - Removes and returns the character on the top of the stack.

**Must Do and Tips**

**Must Do**

* Use 3 **ArrayLists** to store the contents of the files for the 1st secret message
  + arrayMessage1.txt, arrayMessage2,txt, arrayKey.txt
* Use 3 **Queues** to store the contents of the files for the 2nd secret message
  + queueMessage1.txt, queueMessage2.txt, queueKey.txt
* Create your own Stack - **DO NOT** use the java.util.Stack<E> class based on Vector
  + Use an ArrayList as the storage container
  + Include methods for push, pop, isEmpty, and getSize
  + The stack can be a regular class, no need to make it generic

**Note:** For a mental image, this is the process I used to scramble the original message:

* Took the original message text and reversed it, character by character
  + Translator-worked! (original)
  + !dekrow-rotalsnarT (reversed)
* I created the keys (a bunch of 0’s and 1’s that were randomly selected) and wrote them to a file
  + arrayKey.txt
  + queueKey.txt
* Next, I split the reversed message into two strings (two parts) based on the random keys.
  + As an example, with a key pattern of 0 0 1 1 … the **first 2 characters** of the reversed message are placed into the first string (because of 0 0), the next 2 characters were placed into a 2nd string (because of 1 1), etc.
  + Now looking at the reversed string for “Translator worked!” and the random keys:
  + 1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1 0 0
  + ! d e k r o w - r o t a l s n a r T
    - String1: dero-ralsnrT
    - String2: !kwota
* The characters in the 1st string (1st part) were then written to a file
  + arrayMessage**1**.txt (dero-ralsnrT)
  + queueMessage**1**.txt (left for you)
* The characters in the 2nd string (2nd part), were converted to ascii code and that value was written to a file
  + arrayMessage**2**.txt (33 107 119 111 116 97)
  + queueMessage**2**.txt (left for you)

**Output**

**For the 1st set of files (listMessage.txt and listKey.txt):**

The secret string is: Translator-worked!

**For the 2nd set of files (queueMessage.txt and queueKey.txt,):**

The 2nd secret message is a mystery for you to solve. 😊

If your code worked properly on the first set of files then it should properly reveal what the parrots are up to in the 2nd message. Have fun!