Bryce Dunlap

Final

CS5004

Final Report

My approach to this project was to first use the backround knowledge of linked lists applied in Lab 4. Admittedly this was a weak starting point in terms of how the overall structure of my project would pan out, as I never wrote a GUI application before. However, as time grew on I reshaped my project, starting from just wanting to input a simple csv file that would encrypt and decrypt the messages inside to now what is a program that allows the user to enter whatever message they want, multiple times and encrypt/ decrypt them all at once while also using a singly linked list structure in the backround. So, my singly linked list is fairly simple to understand. I have a “main” interface that would be my ListUse, which holds the function methods I can use on my linked list. In the sub classes that extend ListUse I have a class labeled EmptyList which can be viewed as my the final node in the linked list, as if I were pointing to null (in C). The other sub class which extends ListUse is NotEmptyList, which will signify the node or nodes that will hold the data if stored in the list. Also, important to note that this linked list takes in generic data type, so conveniently I can easily customize a linked list for just about anything fairly easy, (example: I changed it from storing todo tasks (datatype) in a linked list to now holding a message(string datatype). I would say this is the “main” concept in my program in terms of what is asked for within the rubric. However, if you were to ask me personally what my “main” accomplishment within this project is, I’d label the CaesarsCipherGUI.java file as the main operator. This GUI takes in quite a few attributes such as buttons, text field, text box etc. Although not entirely ideal to integrate my singly linked list with the GUI, it was a fun demonstration of applying the knowledge learned. The reality is, a linked list would become quite inefficient for this application, it worked well to be able to add to the front of the linked list each time the user adds to the text but deleting a word is where the issue of inefficiency arose. If you look closely at my code you will see that I am not deleting a word at a specific location within the linked list itself, I am simply re-writing it using a for loop and copying over what the text display has stored. I’d also like to note that I made my casears cipher methods within a sub class of “CryptographyWork” interface in order to leave room for future growth in applying different encryption algorithms.

|  |  |
| --- | --- |
| Concept 1 : Recursion | I used this concept in my linked list Look in file NotEmptyList.java on lines 15  I demonstrated this concept by repeatedly calling the method getListSize() while pointing towards the tail. This will go down the whole list until it reaches the EmptyList “node”. |
| … | … |
| … | … |
| Extension Concept 1 :  … | … |

I understand that my learning is dependent on individual effort and struggle, and I acknowledge that this assignment is a 100% original work and that I received no other assistance other than what is listed here.

Acknowledgements and assistance received:

I did not use generative AI in any form to create this content and the final content was not adapted from generative AI created content.

I did not view content from any one else’s submission including submissions from previous semesters nor am I submitting someone else’s previous work in part or in whole.

I am the only creator for this content. All sections are my work and no one else’s with the exception being any starter content provided by the instructor. If asked to explain any part of this content, I will be able to.

***By putting your name and date here you acknowledge that all of the above is true and you acknowledge that lying on this form is a violation of academic integrity and will result in no credit on this assignment and possible further repercussions as determined by the Khoury Academic Integrity Committee.***

|  |  |
| --- | --- |
| Name: Bryce Dunlap | Date: 04/10/2024 |