2022/1/2

Final project

Brian簡智霖 Alberto 艾斌 Eric 徐上恩

Supervisor: Prof Avinash Shankaranarayanan

# **Table of Contents**

[**Table of Contents** 1](#_Toc92473633)

[**1.** **Introduction to the Program** 2](#_Toc92473634)

[1.1 Introduction 2](#_Toc92473635)

[2. Methodology 2](#_Toc92473636)

[2.1 Java Swing 2](#_Toc92473637)

[2.2 JMenu 3](#_Toc92473638)

[2.3 ScrollBar 3](#_Toc92473639)

[2.4 FileChooser 4](#_Toc92473640)

[2.5 Stack 5](#_Toc92473641)

[2.6 Linked list 7](#_Toc92473642)

[2.7 keyevent 8](#_Toc92473643)

[3. Complexity comparison 8](#_Toc92473644)

[3.1 Time Complexity comparison 8](#_Toc92473645)

[3.2 Space Complexity comparison 12](#_Toc92473646)

[4. Code Analysis 13](#_Toc92473647)

[4.1 Stack 13](#_Toc92473648)

[4.2 Linked list 13](#_Toc92473649)

[5. Results / Findings/ Conclusion 15](#_Toc92473650)

[6. Limitations 15](#_Toc92473651)

[7. References / Bibliography 15](#_Toc92473652)

# **Introduction to the Program**

## 1.1 Introduction

Notepad is an indispensable part of life. We often use undo and redo in our notebooks, so we need a good data structure to shorten the running time. In our final project, we use Stack and Linked list to implement in undo and redo. The data structure is a data organization, management, and storage format that enables efficient access and modification.

The data structure is usually based on the computer's ability to obtain and store data anywhere in its memory. The bit string specified by the pointer represents the memory address, which itself can be stored in the memory and operated by the program. Therefore, array and record data structures are based on calculating the addresses of data items using arithmetic operations, while link data structures are based on storing the addresses of data items within the structure itself. The realization of a data structure usually requires writing a set of procedures to create and manipulate instances of the structure.

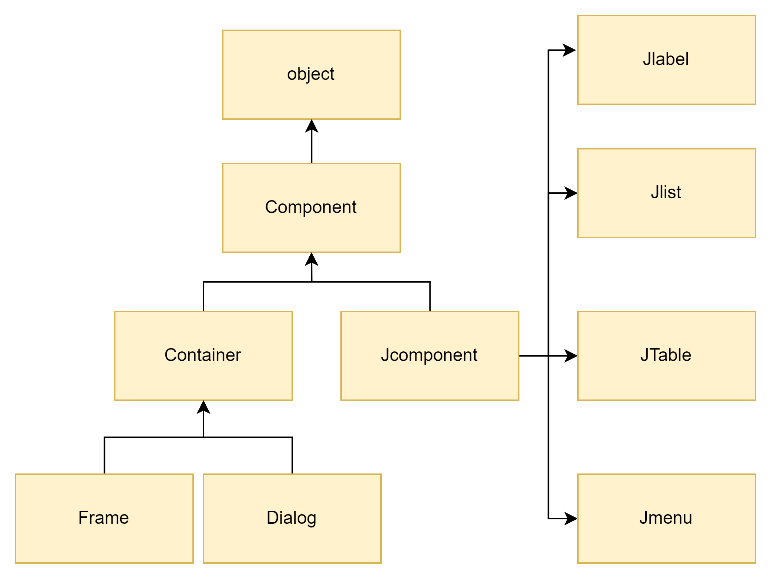
All in all, we have chosen two different data structures to implement in our notepad program.

# Methodology

## 2.1 Java Swing

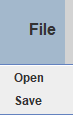
The Java Swing tutorial is part of the Java Foundation Classes (JFC) for creating window-based applications. It is based on the AWT (Abstract Window Toolkit) API and is written entirely in java.

Unlike AWT, Java Swing provides platform-independent lightweight components.

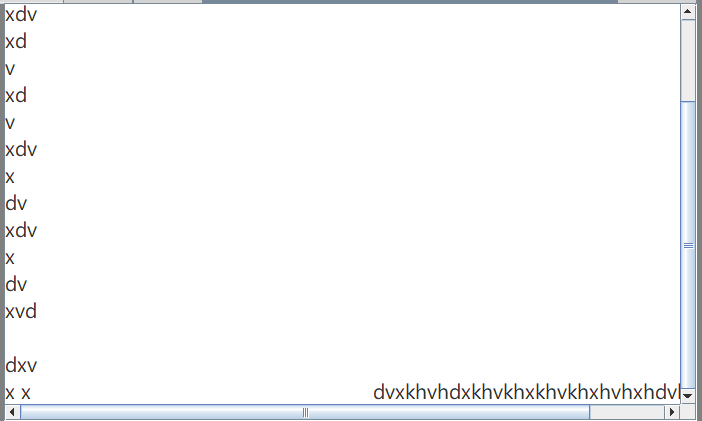
The javax.swing package provides classes for the java swing API, such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser, etc.

## 2.2 JMenu

The menu provides a space-saving way, allowing users to choose one from multiple options. Other components that allow users to make multiple choices include combo boxes, lists, radio buttons, spinners, and toolbars. If the action performed by any of your menu items is duplicated with another menu item or toolbar button.

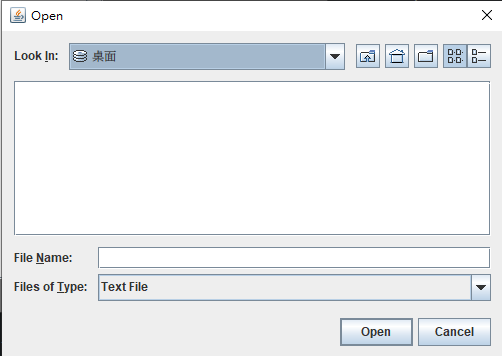


## 2.3 ScrollBar

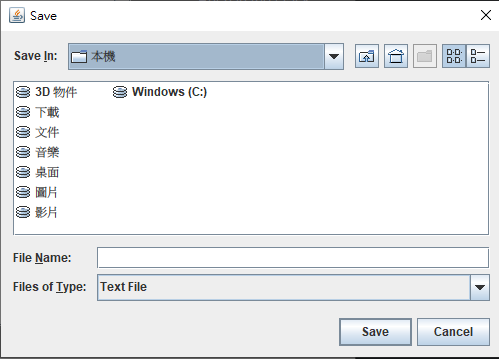
A JScrollPane provides a scrollable view of a component. When screen real estate is limited, use a scroll pane to display a component that is large or one whose size can change dynamically. Other containers used to save screen space include split panes and tabbed panes.

## 2.4 FileChooser

FileChooser can be used to call the file open dialog box for selecting a single file, the file open dialog box for selecting multiple files, and the file saving dialog box. The configuration of the displayed dialog is controlled by the value of the FileChooser property set before calling the corresponding show Dialog method. This configuration includes the title of the dialog, the initial directory displayed in the dialog, and the extended filter for the listed files. For configuration properties whose values are not explicitly set, the displayed dialog box uses its platform default values. The call to the show dialog method will be blocked until the user makes a choice or cancels the dialog. The return value specifies the selected file. If the dialog box is cancelled, the return value is equal to null.



Open File



Save File

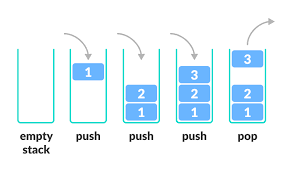
## 2***.5 Stack***

The stack is a basic data structure in which the insertion and deletion of items occurs at the top end of the stack. Therefore, it is also called the LIFO (Last In First Out) data structure.

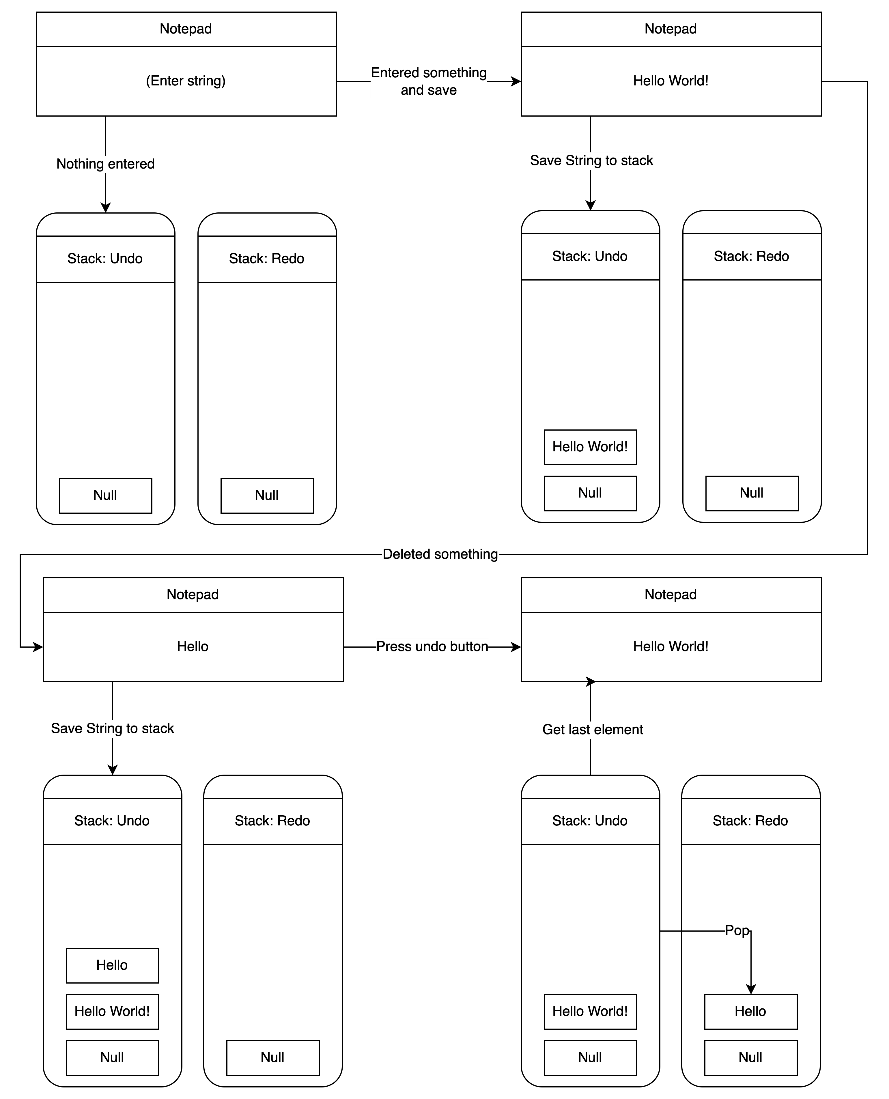
There are 2 main operations related to the stack.

Push data to the top of the stack

Pop and remove the data item at the top of the stack

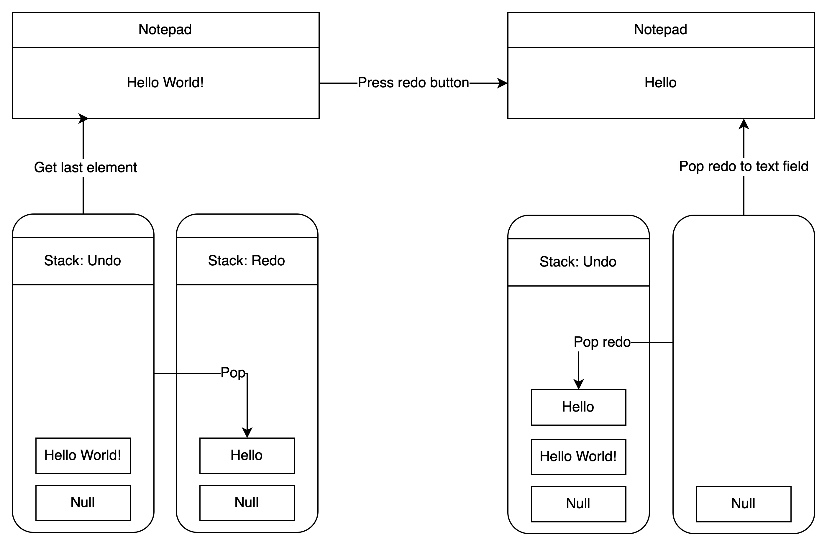


**How our code works (Undo):**



As we can see form the illustration, first we stare at a blank notepad, which at this moment both the undo and redo stack are null; Then in the next step when we typed something (“Hello World!” in this case) and as we pressed the save button, the code will save whatever is in the notepad into the undo stack; In the following step we deleted something (“World!” in this case) and as we hit the save button, the code saves the changed text into stack, so now in the undo stack has two elements; In the last pare, when pressed the redo button the code will first pop the undo stack and save it as a stack element into the redo stack, then it gets the last element of the undo stack to reach the goal of undoing the text of the text field.

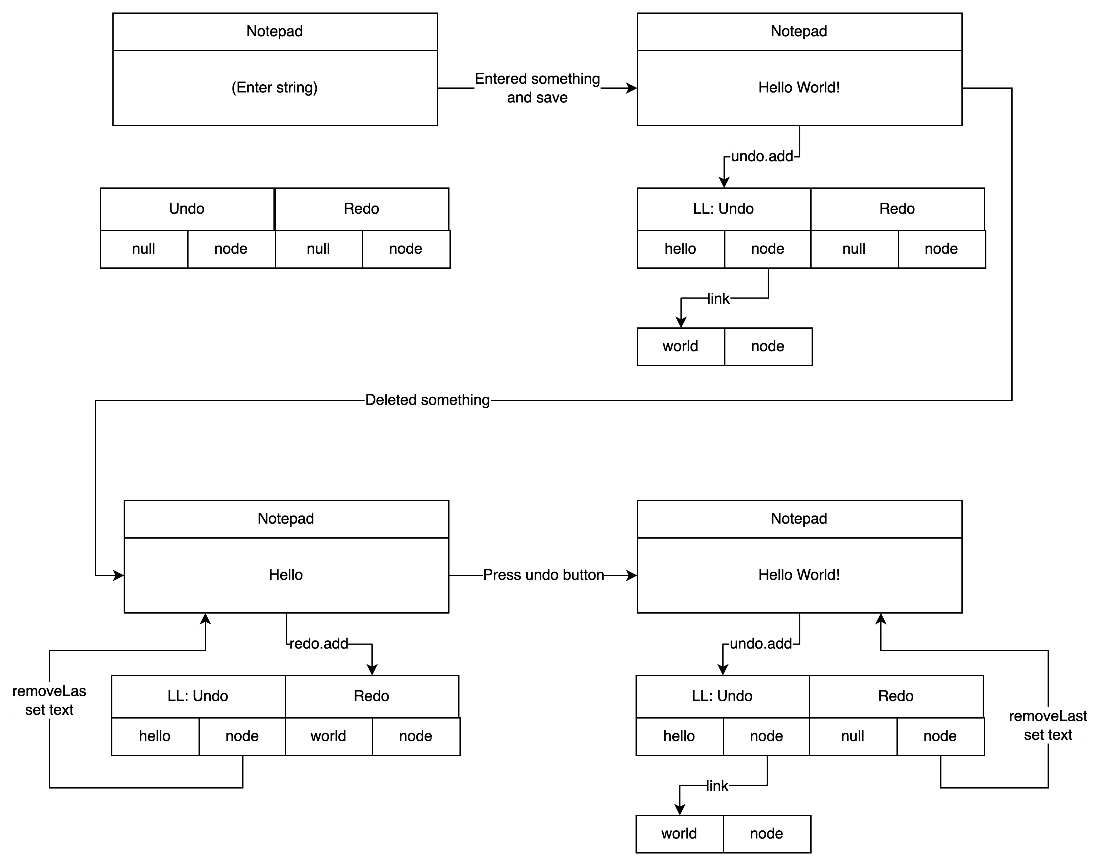
**How our code works (Redo):**



In this illustration we will express how we achieve redo; (The first part of the illustration is followed by the last diagram) In the first part we just saved something into the redo stack (in this case “Hello”); The next step after we pressed the redo button, the code will pop the element of redo (“Hello”) into the undo stack and text field simultaneously by using temp in order to achieve the redo function.

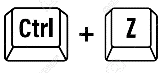
## 2.6 Linked list

Linked list is a common data structure, which uses node to record, represent, and store data, and use the pointer in each node to point to the next node, thereby connecting multiple nodes to form a Linked list, and use NULL To represent the end of the Linked list.



## 2.7 keyevent

Key event can be used to detect key press on keyboard, The KeyEvent category mainly deals with events related to keyboard keystrokes. After implementing the KeyListener interface, keyTyped(), keyPressed(), keyReleased()

In this program we use key Pressed() to detect the keyboard when pressed Space button .When pressing , it will push text into the stack.In this program¸we also use multiple detection , it will execute undo step andwill do the redo step.

# Complexity comparison

## 3.1 Time Complexity comparison

In this section we will express the time complexity among the Stack way and the LinkedList way, and at the final pare we will use a chart and some experimental data to show the difference between each method.

* The Stack method:

There are three method we are using in this project, the first two are push and pop, both of the complexity are: *O(1)*. As for get last element, the time complexity is: *O(n)*. Since it starts looking for the last element from the very front of the index.

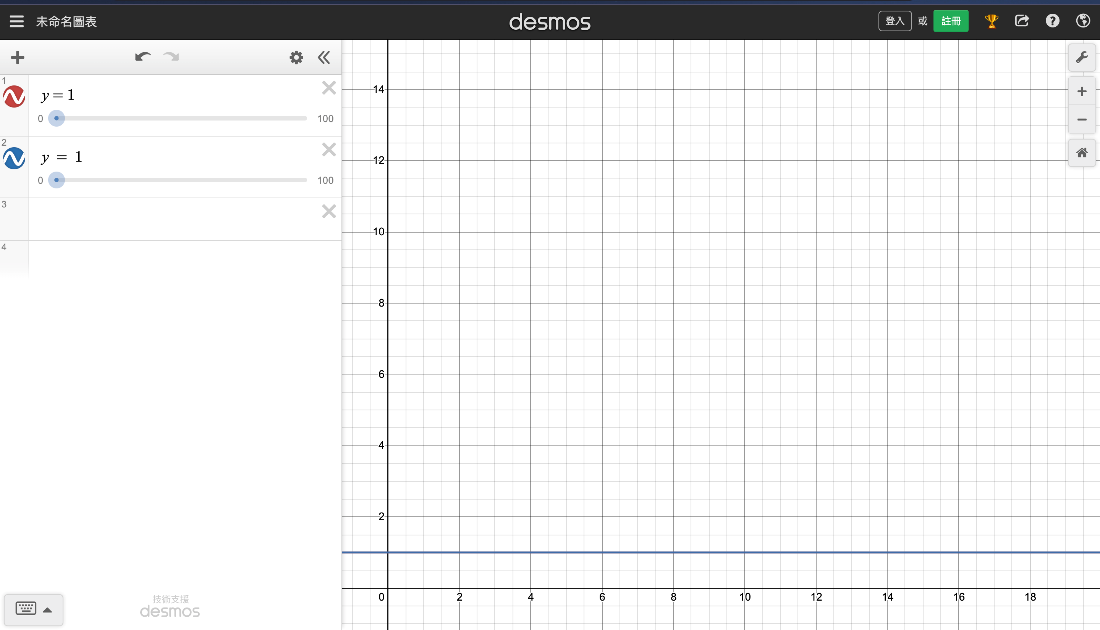
* The LinkedList method:

Both the Insertion and Deletion method are direct, no loops nor recursion need to be executed therefore the time complexity are: *O(1);* However, note that the time complexity of insertion or get elements in particular space is: *O(n)*.

* Theoretical:

|  |
| --- |
| ***Stack Save method:***  undo.push(text); — O(1) |
| ***Stack Undo method:***  temp = undo.pop(); — O(1)  redo.push(temp); — O(1)  SetText(temp); — O(1)  Total: — O(1 + 1 + 1) = O(3) |
| ***Stack Redo method:***  temp = redo.pop(); — O(1)  undo.push(temp); — O(1)  SetText(temp); — O(1)  Total: — O(1 + 1 + 1) = O(3) |
| ***LinkedList Save method:***  Undo.add(); — O(1) |
| ***LinkedList Undo method:***  Temp = undo.getLast(); — O(n)  Redo.add(Temp); — O(1)  Undo.removeLast(); — O(n)  SetText(temp); — O(1)  Total: — O(n + n + 1 + 1) = O(2n + 2) |
| ***LinkedList Redo method:***  Temp = redo.getLast(); — O(n)  undo.add(Temp); — O(1)  redo.removeLast(); — O(n)  SetText(temp); — O(1)  Total: — O(n + n + 1 + 1) = O(2n + 2) |

***Save:***

The save method are all the same in both way of storing datas. Basically there are no difference in the graph.

***Undo & Redo:***

We can see from the pseudocode above basically the undo and redo method are similar, and according to the time complexity we estimated, we can assume that the graph should look like:

We can observe from this chart that while dealing with small amout of data, using the LinkedList method will be slightly faster than using the Stack msthod, however when the n gets larger (amount of data changed in notepad then stored into Stack/LinkedList, in the graph is x) the slower the LinkedList gets, while Stack does not change the excuting speed since it is a constant.

* Experimental:

Since in the theory part we assume that when dealing with small amount of data the linkedlist method is going to be faster, and when dealing with large amount of data the stack method is going to be faster, we therefore designed two experiments:

***Small amount of data:***

We will enter “*Hello World!*” and observe the run time of undoing and redoing.

***Stack:***

|  |  |
| --- | --- |
| Undo | Redo |
| 188,095 | 148,144 |
| 287,176 | 265,445 |
| 216,186 | 233,522 |
| 269,503 | 243,390 |
| 125,435 | 221,049 |

Average undo: 217,279

Average redo: 222,310

***LinkedList:***

|  |  |
| --- | --- |
| Undo | Redo |
| 263,599 | 171,224 |
| 222,059 | 184,572 |
| 205,835 | 192,386 |
| 247,984 | 218,458 |
| 213,792 | 190,847 |

Average undo: 230,653.8

Average redo: 191,497.4

***Large amount of data:***

We will enter *“Your mind and spirit is the strongest weapon you have, trust in the allfather then trust yourself!”* and observe the run time of undoing and redoing.

***Stack:***

|  |  |
| --- | --- |
| Undo | Redo |
| 171,472 | 138,160 |
| 153,556 | 162,040 |
| 154,525 | 123,979 |
| 178,265 | 182,196 |
| 153,378 | 162,139 |

Average undo: 162,239.2

Average redo: 153,702.8

***LinkedList:***

|  |  |
| --- | --- |
| Undo | Redo |
| 309,185 | 264,769 |
| 244,505 | 141,702 |
| 224,475 | 221,332 |
| 202,545 | 147,219 |
| 298,965 | 261,128 |

Average undo: 255,935

Average redo: 207,230

* Conclusion:

It can be known from the above experiment that even though Stack and Linked list There is not much difference in small amount of data and large amount of data, in the experiment we observed that in the longer length of a sentence the run time is nearly 1.5 to twice as more than the shorter.

## 3.2 Space Complexity comparison

In this section we will express the space complexity among the Stack way and the LinkedList way, and at the final pare we will use a chart and some experimental data to show the difference between each method.

* General:

When we store n items (n data stores n times) then the space complexity is O(n), however storing n items once (n data stores 1 times) the space complexity will reduce to O(1). In our piece of code, we store the data whenever the user hits space key or enter key. Hence, if we type a sentence step by step, the space complexity will be O(n), which is decided by the times you stored. However, if we paste the whole sentence then save it, it would be considered to be the second situation, thus the space complexity is O(1).

* The Stack method:

The space complexity of *Stack.push()* and *Satck.pop()* is *O(1)*, so combine this concept with the storing way we mentioned, the space complexity is showed below.

|  |  |
| --- | --- |
| Stored n times | *O(n)* |
| Stored 1 time | *O(1)* |

* The LinkedList method:

Linked lists hold two main pieces of information (the value and pointer) per node. This means that the amount of data stored increases linearly with the number of nodes in the list. Therefore, the space complexity of the linked list is linear: O(n). Combining the fact of saving data, the time complexity shows below.

|  |  |
| --- | --- |
| Stored n times | *O(n) \* O(n) = O(n^2)* |
| Stored 1 time | *O(n) \* O(1) = O(n)* |

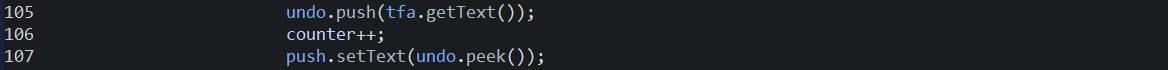
* Conclusion:

Although LinkedList is more flexible, it takes more space to store data to achieve its flexibility.

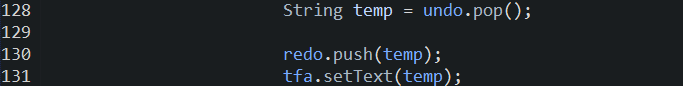
# Code Analysis

## 4.1 Stack

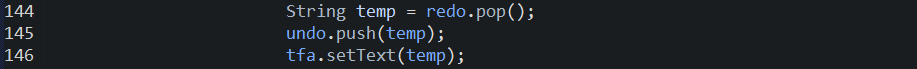
Push(Save)



Pop(Undo)



Pop(Redo)

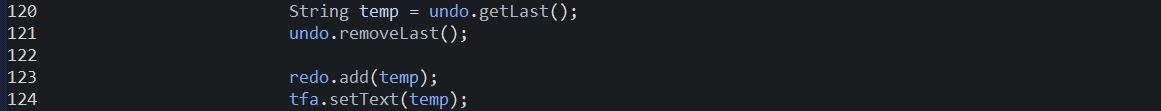


## 4.2 Linked list

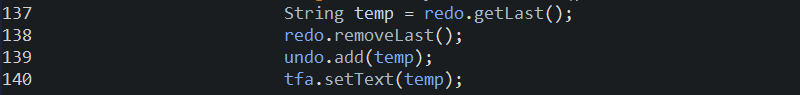
Add(Save)



(Undo)



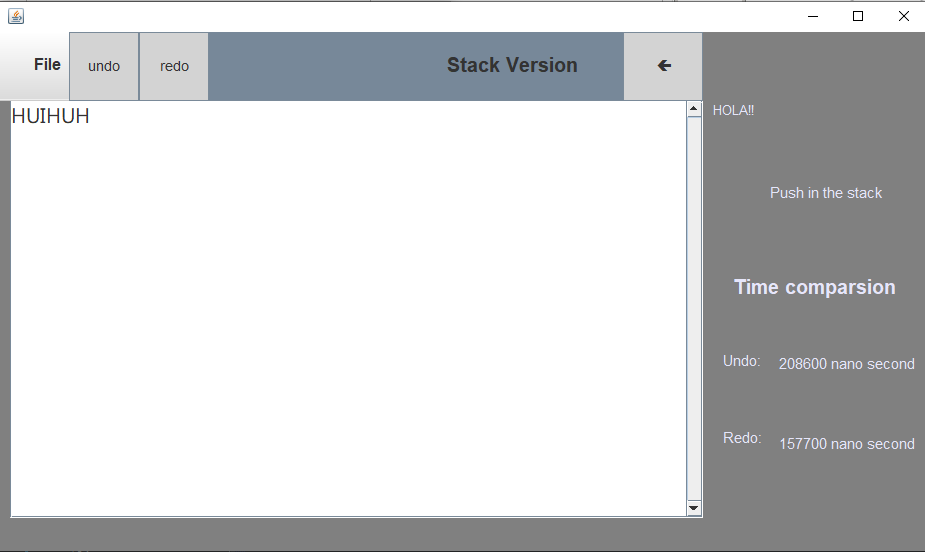
(Redo)

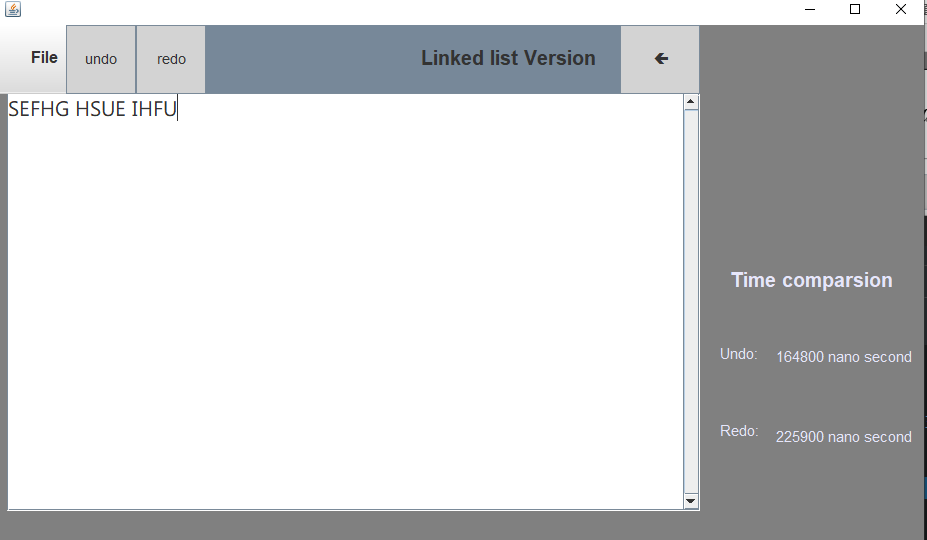


***Menu.java***



***Note\_Stack.java***



***Note\_Linked list.java***

# Results / Findings/ Conclusion

In the compression part of the run time, although the data we tested might not be enough for big data, what we did is remove the smallest 5% and largest 5% in order to reach the Normal distribution, and the result matches the theory more than the data's that is not been removed, which make sense, since the more experimental data we get the more data is close to the average which is the definition of Normal distribution.

# Limitations

This program cannot distinguish the difference between Stack and Linked list running time when having a small amount of data and this program is no way to have better functions like other notepads, but if we have more time, we can improve the entire system and sell our software online.

# References / Bibliography

[1]

Professor Notes: Data Structures Handbook in Java

Shankaranarayanan, A., & Amaldas, C. (n.d.). Professor Notes: Data Structures Handbook in Java (Taiwanese Edition). 1st edition, TJ Publisher.

[2]

What is a Stack in Java? | Java Stack Introduction and Examples | Java Collections | Stack in Java. (2021, April 1). YouTube. https://www.youtube.com/watch?v=nyoj9yUheR8

[3]

Java API 分類導覽 - java.awt.event.KeyEvent. (n.d.). Bb. https://pydoing.blogspot.com/2012/07/java-api-keyevent.html

[4]

FileChooser (JavaFX 8). (2015, February 10). S. https://docs.oracle.com/javase/8/javafx/api/javafx/stage/FileChooser.html

[5]

Lakshan, P. (2021, December 14). Introduction to Stacks in Java - Javarevisited. Medium. https://medium.com/javarevisited/introduction-to-stacks-71dfad8782ef

[6]

GeeksforGeeks. (2021, September 17). Stack Data Structure (Introduction and Program). https://www.geeksforgeeks.org/stack-data-structure-introduction-program/

[7]

Comparison of Searching methods in Data Structures. (n.d.-b). S. https://www.tutorialspoint.com/comparison-of-searching-methods-in-data-structures

[8]

GeeksforGeeks. (2021b, October 4). JavaFX | FileChooser Class. https://www.geeksforgeeks.org/javafx-filechooser-class/

[9]

How to use KeyEvent. (2012, November 25). Stack Overflow. https://stackoverflow.com/questions/13552229/how-to-use-keyevent

[10]

What are the time complexities of various data structures? (2011, September 3). Stack Overflow. https://stackoverflow.com/questions/7294634/what-are-the-time-complexities-of-various-data-structures