# Introduction

This assignment will provide an ADT template, an algorithm system utilizing two data structures stack and queue. The concept should help readers grasp some of these structures ' important activities better. Furthermore, the program's test and error handling activities will be checked to effectively calculate the algorithms provided.

# I. Implement a complex ADT

# 1.Identify the problem

We will provide a menu system consisting of assignment 2 briefs as needed.

two main functions:

* Transfer Messages

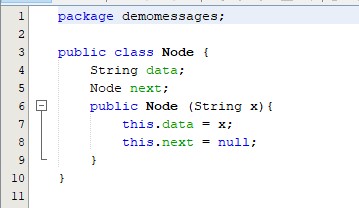
When user press 1: It will transfer a list of messages, store in a queue.

* Processing Messages

When user press 2: It adds all messages from queue to the stack and processes all messages.

# 2. Program implementation

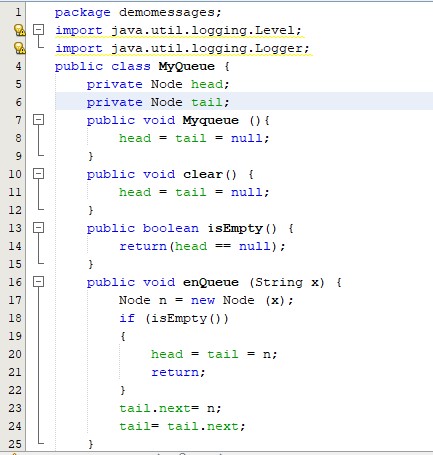
Below are several algorithms for initializing a menu according to the above requirements: First, the node class will be initialized

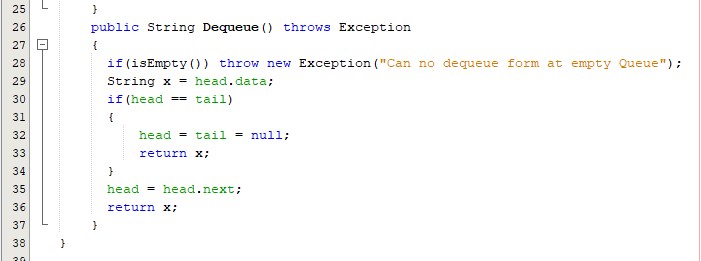


Next, apply the stack and queue structure to the algorithm, create "MyQueue" and "Mystacks" classes

# Queue code :

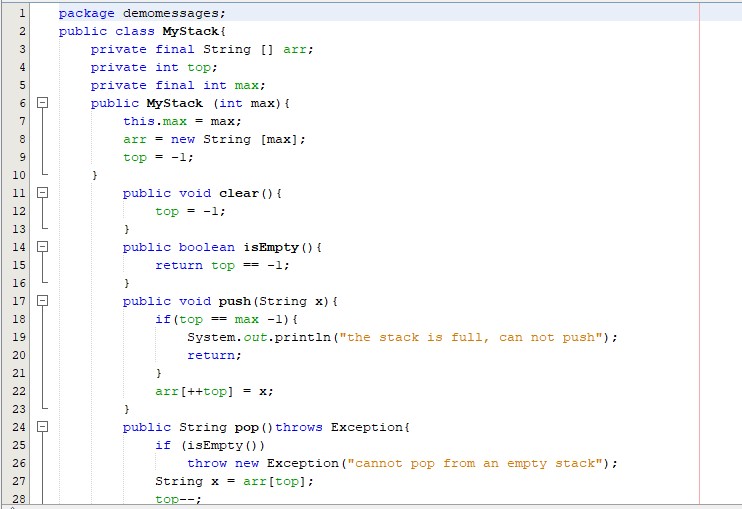
We must repeat some basic theory about queues before we write code about queue structure. A queue is simply a waiting line that grows by adding elements to its end and shrinks by removing elements from the front. (Adam Drozdek, 2006). The queue is an open data structure for both ends: one for storing (adding) new elements and one for removing them.



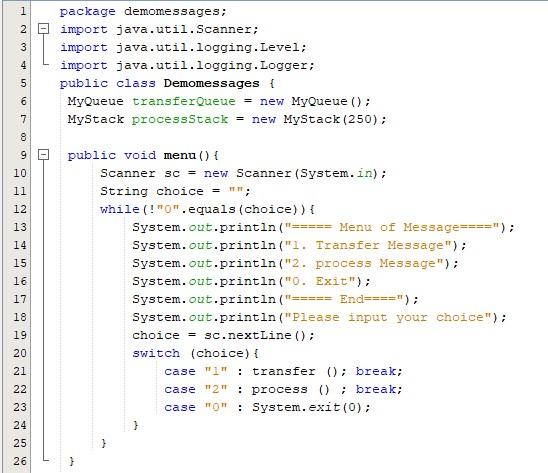


Stack code :

Before the stack structure is added to the algorithm.” A stack is a linear data structure that can be accessed only at one of its ends for storing and retrieving data. It's an abstract type of information that operates with two main operations: push and pop.”

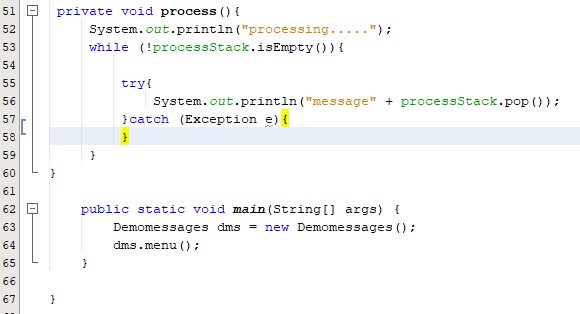


Finally, create a Demo Messages class:





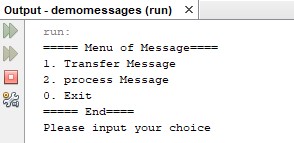
We can send messages by stack by queue and processing messages on the request shown above.



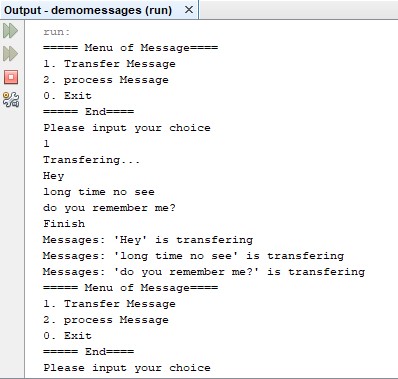
After the code of the software was written. Next, we run the displayed software and menu. There will be 2 functions shown at the request of the lesson and 1 function to exit.

1. Transfer message
2. Processing messages

0. Exit

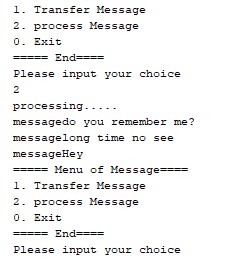


With the three functions mentioned above. If the user wants to place the message in the queue, on the display panel, the user only has to pick the number 1. Then enter the message information that will be stored by the user and click enter. The user only has to enter the "finish" text to complete the message transmission. The message is sent to the queue.

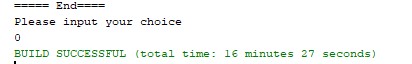


The user only has to type "2" + Type if the user wants to process the message.

The information table for the message will be stored.



The user enters "0" + Enter to finish the program. The software is done.

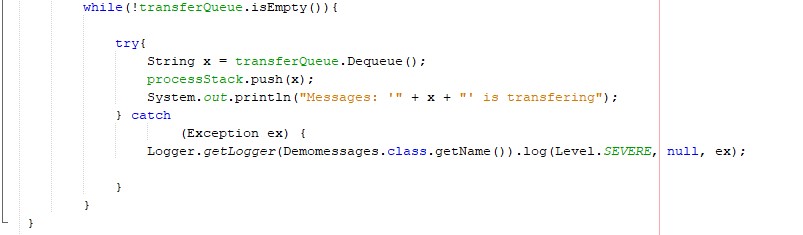


# II. Implement error handling and report test results.

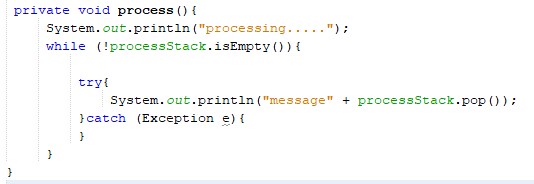
# 1. Put some exceptions in your program (throw exceptions and

# try/catch)

Try to push x to become an empty queue when the Queue is not empty



ProcessStack



# 2. Test cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cas  e | Test | Value | Result | Statu s | Note |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | Transfer  Messages | Hey long time no see do you remember me? |  | Pass | Messages  transfer successful |
| 2 | Transfer  Messages |  |  | Pass | All  characters can be Transfe |
| 3 |  | Hey long time no see do you remember me? |  | Pass | When not entered: "Finish", Transfer does not occur |
| 4 | Switch | When user enters: 5 |  | Pass | "5" is not one of the choices |
| 5 | Transfer  Messages |  |  | Fail | After enteri ng the butt on "finish" t o complete the transm ission of th e message |
|  |  |  |  |  | . "Finish" is listed in th  e queue as well. |
| 6 | Process  Messages | Queue has Messages:  “Hey”  “long time no see”  “do you remember me?” |  | Pass | Process successful  Messages |
| 7 | Process  Messages | Queue is empty |  | Fail | When  entering "Finish". "end" is considered a message |
| 8 | Exit | When user  enters: 0 |  | Pass | The program finished successfull y |

# Evaluation of the effectiveness of an algorithm

# 1. Asymptotic analysis

An algorithm's asymptotic analysis refers to the interpretation of its run-time performance by mathematical foundation/framing. We can very well assume an algorithm's best case, the average case, and worst-case scenario using asymptotic analysis.

Asymptotic analysis is constrained by input, i.e. if there is no input to the algorithm, operating in constant time is inferred. All other variables other than the "data" are assumed to be constant.

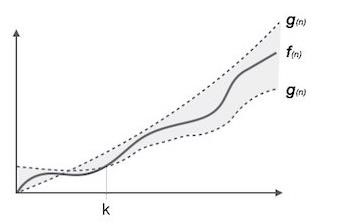
Asymptotic analysis refers to the estimation of the lifetime of any operation in abstract mathematical units. For example, one operation's runtime is calculated as f(n) and may be calculated as for another operation. It means that with the rise in n, the first operation running time will increase linearly and the running time of the second operation will increase exponentially as n increases. Similarly, if n is significantly small, the running time of both operations will be almost the same.

Usually, the time required by an algorithm falls under three types −

**Best Case** − Minimum time required for program execution.

**Average Case** − Average time required for program execution.

**Worst Case** − Maximum time required for program execution.



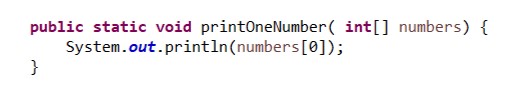
(Anon., n.d.)

# 2.Big-O Notation

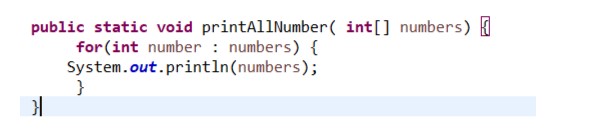
The Big-O Notation can be summarized in one sentence: how quickly the program runs, the program's reliance on its input as the input increases as the program runs.

There are three aspects that should be considered in Big-O Notation :

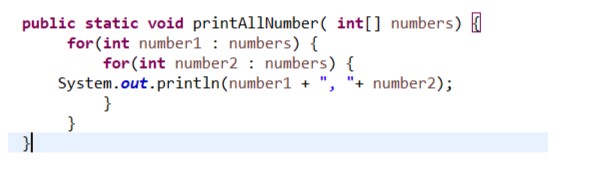
* The run time
* The input
* The increase in input. Big-O Notation and runtime For example:



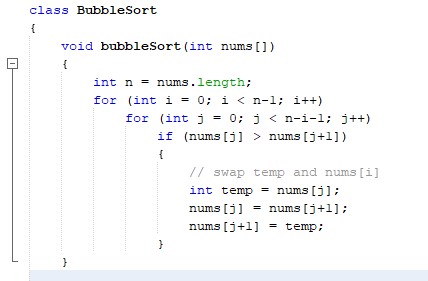
In the above, it takes O(1) to complete the scheme as it only needs to print one number throughout the whole series.



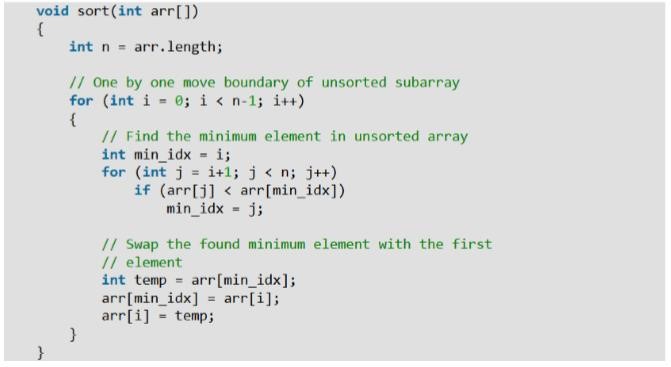
Suppose the array has 10 numbers then O(10) would have to complete the above function.



In this case, the input values are dependent on the user and the function has 2 for) (loops so this program will take O(n^2) time to complete if there was only one for) (loop then only O(n) would be needed.



The algorithm of bubble sorting has 2 for) (loops with two variables I and j so the complexity is O(n^2).



There are also 2 for) (loops in the selection sort algorithm so the complexity is also O(n^2)

# D. Conclusion

This report has clarified and identified the problem, Implement a complex ADT, in addition to executing code statements about Stack code. Implement error handling and report test results. After completing the next program will be Test cases. Finally, to evaluate the effectiveness of the algorithm and Asymptotic analysis, Big-O Notation

# References

Adam Drozdek, 2006. *Data Structures and Algorithms in Java.* s.l.:s.n.

Anon., n.d. [Online]

Available at: https://www.tutorialspoint.com/data\_structures\_algorithms/asymptotic\_analysis.htm