Programming Assignment 3

Jose A. Valdivia Rojas

Robert Alexander

**CMSC203 Assignment 3 Implementation**

**Class:** CMSC204 CRN 21479

**Program:** Assignment #3

**Instructor:** Robert Alexander

**Summary of Description**: Your assignment is to write a generic double singly-linked list class with an iterator, and a generic sorted double singly-linked list class with an iterator that inherits from your generic double singly-linked list class. The GUI has been provided for you for this assignment to help you visualize your linked list. Your list classes will also be tested with Junit tests. Upload the initial files from Blackboard and your working files in a directory into the repository in GitHub you created in Lab 1 and take a screen shot of the files

**Concepts:**

Exception handling

Generic Classes

Double Linked List

Ordered Double Linked List

Iterators

Comparators

**Due Date:** 10/13/2021

**Integrity Pledge:** I pledge that I have completed the programming assignment independently. I have not copied the code from a student or any source.

Jose A. Valdivia Rojas

**Part1: UML DIAGRAM**: Here is a UML diagram for Assignment 3 program:

GENERIC CLASSES

|  |
| --- |
| BasicDoubleLinkedList<T> implements Iterable<T> |
|  |
| + BasicDoubleLinkedList()  + Node<T>  + addToEnd( data: T ): BasicDoubleLinkedList<T>  + addToFront( data: T): BasicDoubleLinkedList<T>  + getFirst(): T  + getLast(): T  + getSize(): int  + isEmpty(): boolean  + iterator ( ) : ListerIterator<T>  + remove(targetData: T, comparator: Comparator<T>) : BasicDoubleLinkedList<T>  + retrieveFirstElement() : T  + retrieveLastElement() : T  + toArrayList (): ArrayList<T> |

|  |
| --- |
| SortedDoubleLinkedList<T> |
|  |
| + SortedDoubleLinkedList  + add ( data: T ): SortedDoubleLinkedList <T>  + addToEnd( data: T ): BasicDoubleLinkedList<T>  + addToFront( data: T): BasicDoubleLinkedList<T>  + iterator ( ) : ListerIterator<T>  + remove(data: T, comparator: Comparator<T>) : SortedDoubleLinkedList <T> |

EXPECTION CLASSES

|  |
| --- |
| UnsupportedOperationException |
|  |
| + UnsupportedOperationException ( )  + UnsupportedOperationException ( str: String) |

|  |
| --- |
| NoSuchElementException |
|  |
| + NoSuchElementException ( )  + NoSuchElementException ( str: String) |

**TABLE TEST**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Input** | **Actual Input** | **Expected Output** | **Actual Output** | **Did the test pass?** |
| 1 | addToEnd(Alex) | Alex | Alex | Alex | YES |
| 2 | addToEnd(Fiona) | Fiona | Alex, Fiona | Alex, Fiona | YES |
| 3 | addToFront(Derek) | Derek | Derek, Alex, Fiona | Derek, Alex, Fiona | YES |

**Part3: Screenshots:**

**Doubly-Linked-List-GUI**

**Graphical user interface, application

Description automatically generated**

**BasicDoubleLinkedListTest**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

**SortedDoublyLinkedListTest**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

**LESSON LEARNED**

A valuable skill I learned from this project is designing a Doubly Linked List and Sorted Doubly Linked List algorithms from scratch with GUI provided that convert a List program that stores objects. The goal of this project was to design a generic Data Structure, Basic DLL and Sorted DLL, from scratch which are classes to store any objects of different type in a list made up of nodes with forward and backward navigation . Furthermore, the inner class Node will create nodes that references the previous and next nodes, and each of previous and next nodes will have other previous and next nodes that finally will create a chain. By doing this project, I could gather all the knowledge like the *Generics, Double Linked List, Sorted DLL, Exceptions, and Iterators*  concepts to make the Doubly Linked List program; However, the program was not easy like it sounds. I found several issues with the add method in the SortedDLL class. Therefore, I had to go for help for a tutor on the STEM center; Luckily, my tutor and me could figure it out how to resolve the problem. The code that I was missing was the while loop that I needed to traverse through the list. On next projects, I will keep studying to get more knowledge and create more fun programs.

**SCREENSHOT GITHUB:**

**Check List:** <Provide answers to the column Y/N or N/A >**:**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N** | **Comments** |
|  | **Assignment files:** |  |  |
|  | * FirstInitialLastName\_ Assignment#\_Moss.zip | **Yes or No** | **Y** |
|  | * FirstInitialLastName\_Assignment#.docx/.pdf | **Yes or No** | **Y** |
|  | * Source java files | **Yes or No** | **Y** |
|  | **Program compiles** | **Yes or No** | **Y** |
|  | **Program runs with desired outputs related to a Test Plan** | **Yes or No** | **Y** |
|  | **Documentation file:** |  |  |
|  | * Comprehensive Test Plan | **Yes or No** | **Y** |
|  | * Screenshots related to the Test Plan | **Yes or No** | **Y** |
|  | * Screenshots of your GitHub account with submitted Assignment# (if required) | **Yes or No or N/A** | **Y** |
|  | * UML Diagram (if required) | **Yes or No or N/A** | **Y** |
|  | * Algorithms/Pseudocode (if required) | **Yes or No or N/A** | **Y** |
|  | * Flowchart (if required) | **Yes or No or N/A** | **Y** |
|  | * Lessons Learned | **Yes or No** | **Y** |
|  | * Checklist is completed and included in the Documentation | **Yes or No** | Y |