**Project 3 Write Up: Dolev Peleg**

**Approach, Design, and Algorithm**

My approach to this project was to start designing BasicDoubleLinkedList first, as it is the super class, then design each of its inner classes (Node and DoubleLinkedListIterator) when I encounter a method that requires them. Only after these were finished and tested, I moved on to design SortedDoubleLinkedList, the subclass.

At first glance, this project seemed not as complicated as our previous one (infix to postfix), and I designed BasicDoubleLinkedList and its inner classes with ease. I was too quick to create these classes, and did not test them as I was creating them, which caused me a lot of issues later on.

After finishing BasicDoubleLinkedList and its inner classes, I started working on SortedDoubleLinkedList. This class was more complicated to design due to the add() method having many if-else blocks, and nested if-else blocks.

After I finished working on all my classes, I tried to run the JUnit tests that were provided to me by my instructor. This is when I found out I have to debug many of my methods, even the ones that I was sure that I got right. The two methods that were the most complicated for me to debug were the methods of the inner class DoubleLinkedListIterator, and the add() method of SortedDoubleLinkedList. After some debugging, all the tests for BasicDoubleLinkedList passed, but the same methods that were used in this test, did not work for its subclass SortedDoubleLinkedList. This is when I realized that the add() method was not working correctly. I managed to find the bug, that was a logic mistake on line 91. Initially, I set currentDataNode.prev to the tail, when the correct statement was to set newNode.prev to the tail. This fixed all my issues, and all my tests were successful.

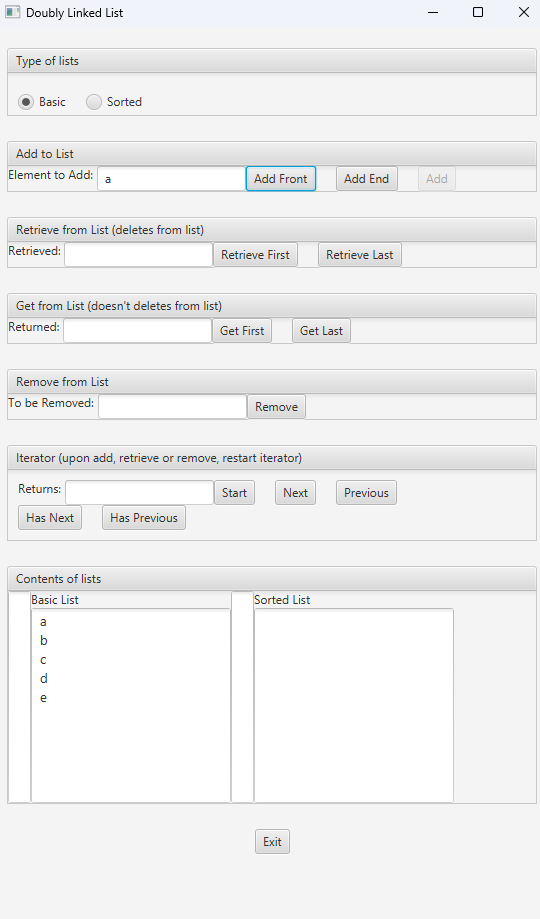
In hindsight, I should have tested all my core methods (the addition methods and iterator methods) before continuing to work on the next methods that rely on them functioning correctly. In my next project, I am sure I will learn from this mistake, and I will code slower, but more thoroughly to avoid the long and exhausting debugging process at the end.

I did not use any outside sources besides my notes, the textbook, and the PowerPoint and lectures that were provided to me.

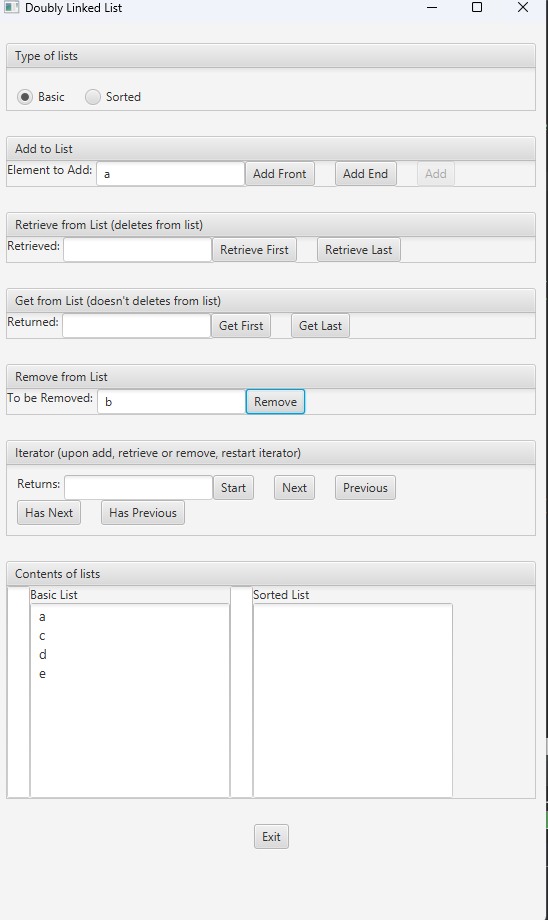
**Test Runs and Cases**

**Basic List**

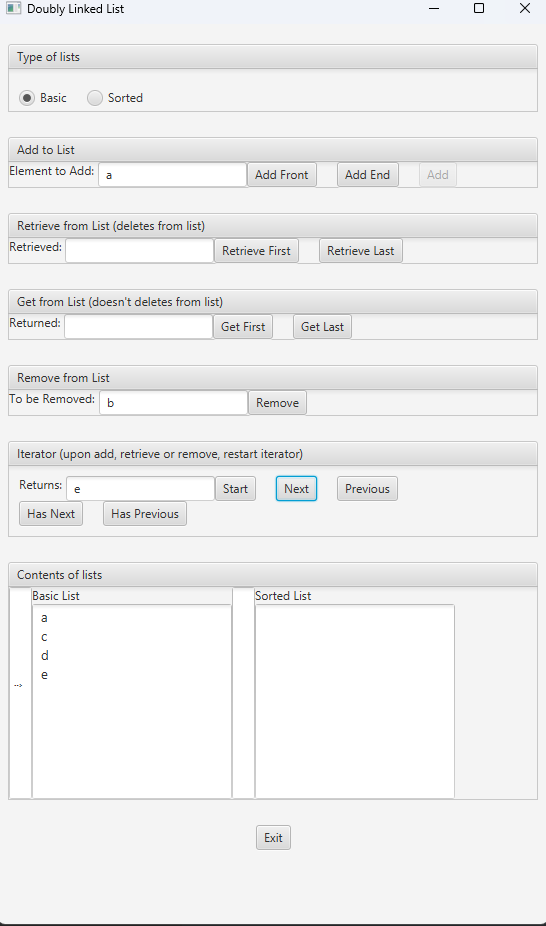
**Adding:**

****

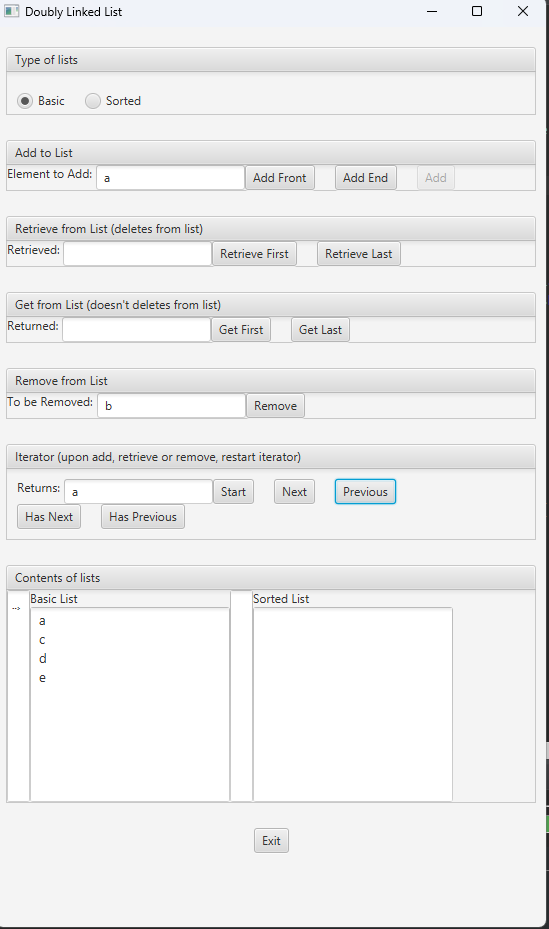
**Removing (b is removed)**

****

**Iterator next() through the list**

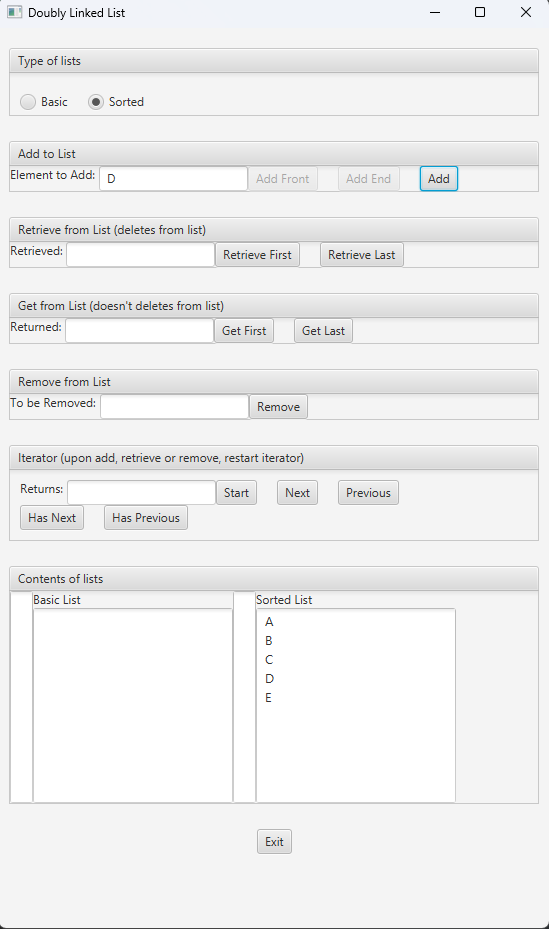
****

**Iterator previous() back the list**

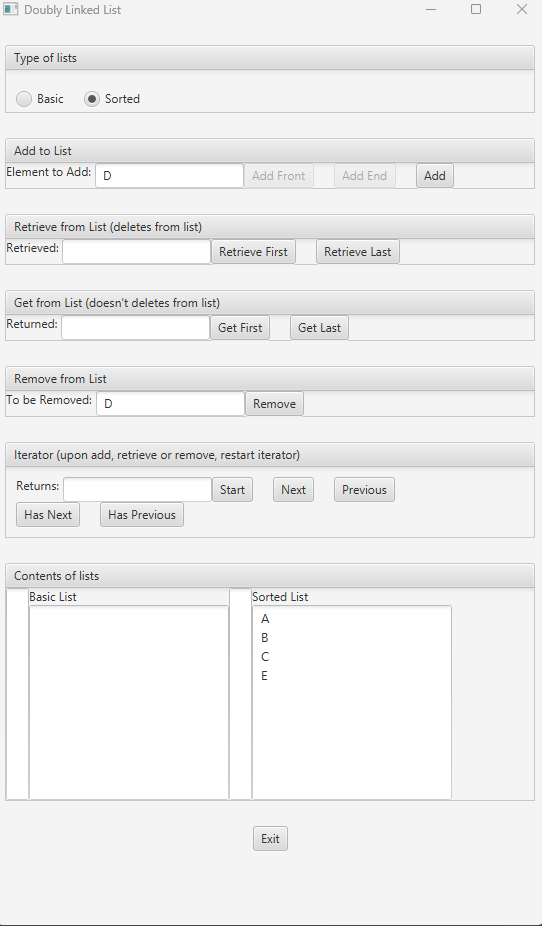
****

**SortedList**

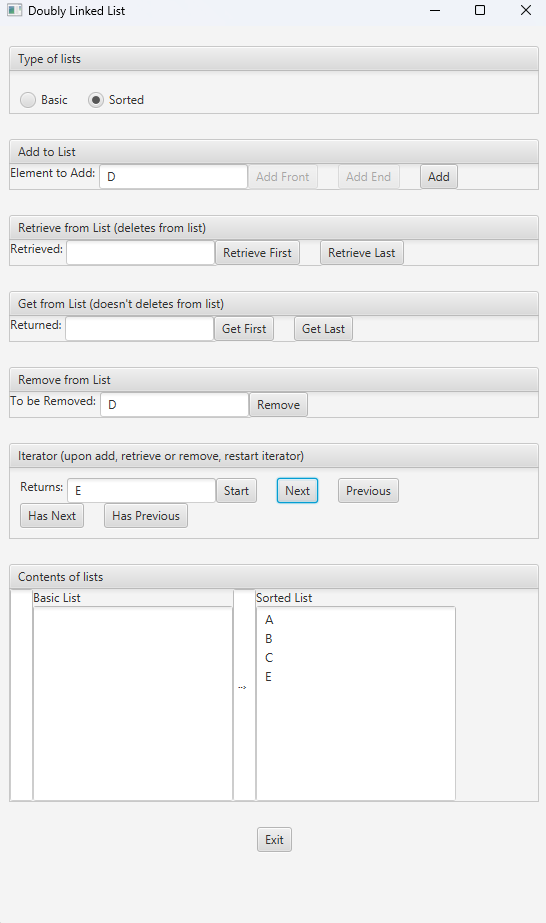
**Adding:**

****

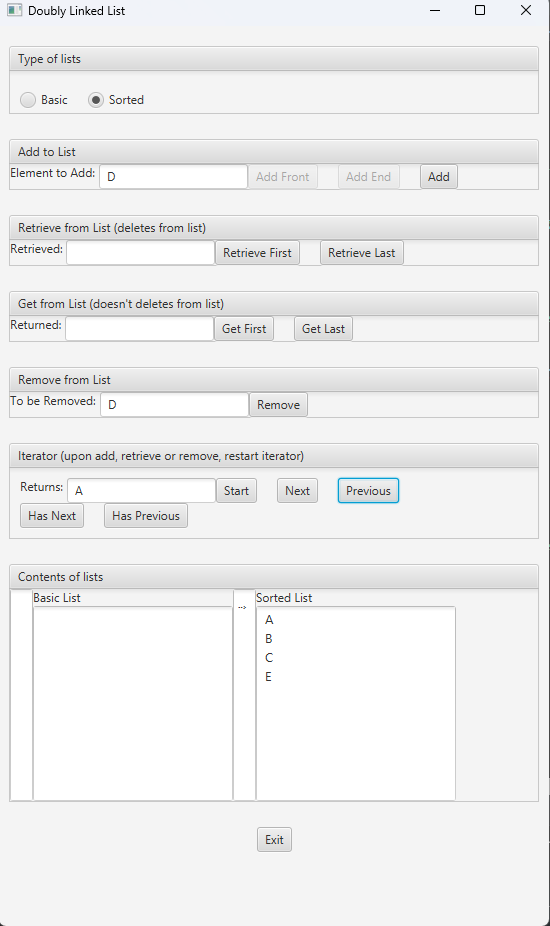
**Removing (D is removed)**

****

**Iterator next() through the list**

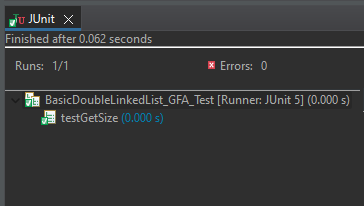
****

**Iterator previous() back the list**

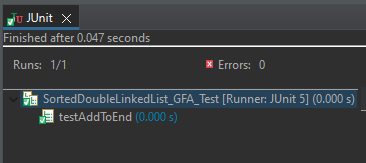
****

**JUnit Testing**

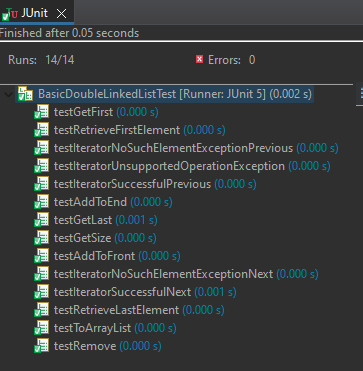
**BasicDoubleLinkedList\_GFA\_Test**

****

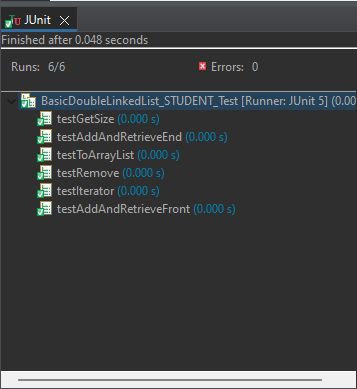
**SortedDoubleLinkedList\_GFA\_Test**

****

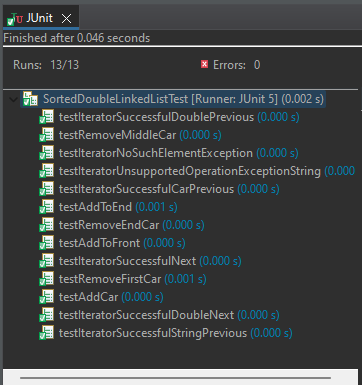
**BasicDoubleLinkedListTest**

****

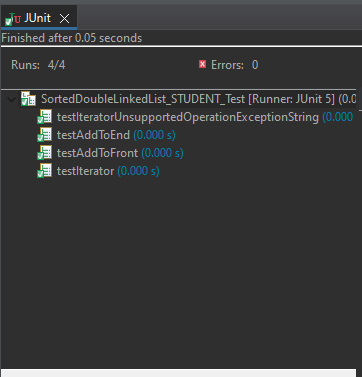
**BasicDoubleLinkedList\_STUDENT\_Test**

****

**SortedDoubleLinkedListTest**

****

**SortedDoubleLinkedList\_STUDENT\_Test**

****

**Learning Experiences**

I learned a lot about how to use a double-linked list in this project. Our book does not provide a lot of information on how to create and use this types of lists, so making one “from scratch” was a challenging, yet important experience.

It is worth mentioning that although I gained a lot of coding experience and useful coding skills during this project, my lessons were how to test and debug my code better. If I were to test my methods as I was creating them, it could have been a lot easier for me to debug at the end of the project. I had a lot of bugs, which improved my debugging skills, but now I understand how crucial it is to prevent these bugs in the first place by testing the core methods early.

Nest project I will test earlier, and feel more comfortable working with linked lists, doubly linked lists, sorted lists, and iterators.

**Assumptions**

1. The user will be using JUnit 5, Java, and JavaFX
2. A new iterator will be created after each change to the list (addition, removal, etc).

**Enhancements**

No enhancements were made.