Lab 2 Report – Christian Graber

Task 1. The goal here is to create a base method for adding and removing items and to compare it with two other classes to test efficiency, when using different methods to accomplish these goals. This first class is just uses basic add item and remove item methods. I predict that it will most likely be the least effective in most scenarios, as it is the most basic, and does not use any methods to eliminate the number of moves, removes, or compares.

Task 2. The goal here is to create a derived class, which investigates the theory that by starting from the end of the array, fewer items will need to be moved on an insert. This derived class attempts to make slightly more efficient method of insertion, by starting insertion of new items from the end instead of the beginning. Going in reverse order may be easier for the computer to process, and could require less moves and compares overall when compared to the first task. I predict this method will be the most effective, as it requires no shifting or moves, since items are stacked on each other. This should be the fastest because of this.

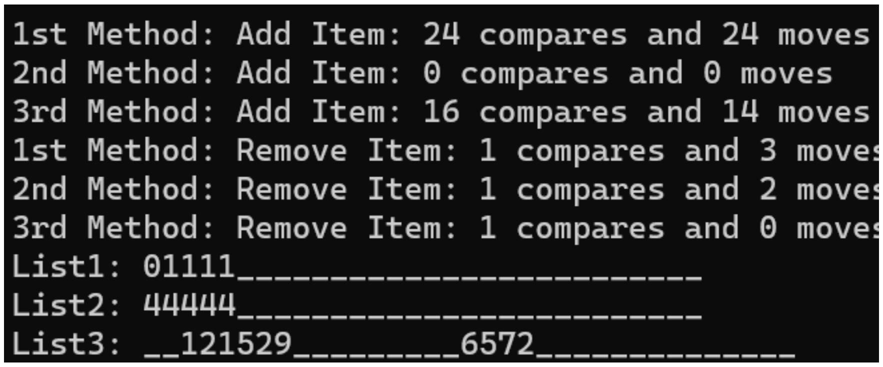
Task 3. The goal here is to create another derived class, which investigates the theory that by leaving blank spots within the array, it is possible that fewer move operations are required for an insert. This derived class attempts to make a more efficient method of insertion (AddItem) by leaving blank spots between objects in the array, which allows for insertion of objects, rather than entire object shifts. I predict this method will be more efficient than the base operation (1st method) but not the 2nd method.

SAMPLE TEST OUTPUTS:

*The below tests demonstrate the efficiency of each of the methods in adding and removing items for an array. They are separated by size, with a photo of each output from a sample test.

30 objects:

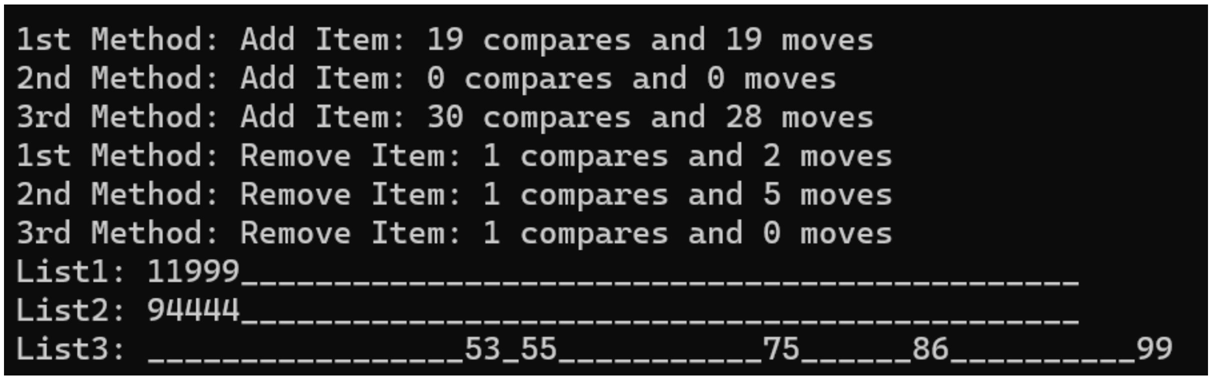
For 30 objects, it appears the 2nd add item method was overall the most effective, with 0 compares and 0 moves in this run of the test. Also, the 1st add item method was least effective for this array. In terms of the remove method, the 1st method was the least efficient, and resulted in the most moves.



```
1st Method: Add Item: 24 compares and 24 moves
2nd Method: Add Item: 0 compares and 0 moves
3rd Method: Add Item: 16 compares and 14 moves
1st Method: Remove Item: 1 compares and 3 moves
2nd Method: Remove Item: 1 compares and 2 moves
3rd Method: Remove Item: 1 compares and 0 moves
List1: 01111_________________________
List2: 44444_________________________
List3: __121529_________6572______________
```

50 objects:

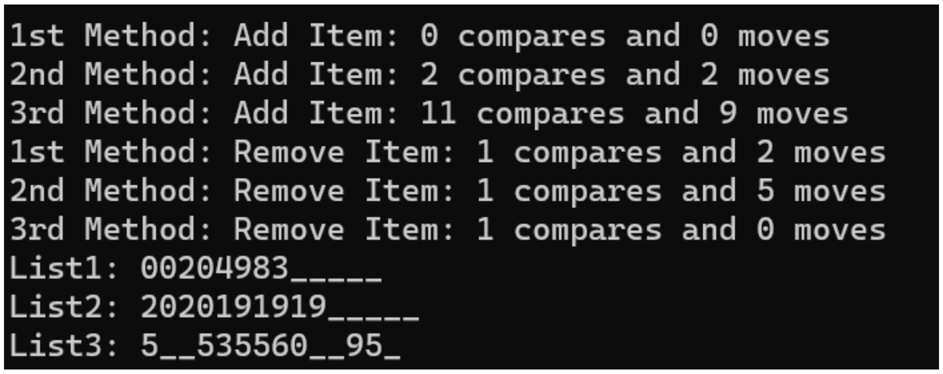
For 50 objects, it appears the 2nd add item method was the overall most effective, with 0 compares and 0 moves in this run of the test. Also, the 3rd add item method was least effective for this array. In terms of the remove method, the 2nd method was the least efficient, and resulted in the most moves.



```
1st Method: Add Item: 19 compares and 19 moves
2nd Method: Add Item: 0 compares and 0 moves
3rd Method: Add Item: 30 compares and 28 moves
1st Method: Remove Item: 1 compares and 2 moves
2nd Method: Remove Item: 1 compares and 5 moves
3rd Method: Remove Item: 1 compares and 0 moves
List1: 11999_____________________________________________
List2: 94444_____________________________________________
List3: _________________53_55___________75______86__________99
```

10 Objects:

For 10 objects, it appears the 1st add item method was overall most effective, with 0 compares and 0 moves in this run of the test. Also, the 3rd add item method was least effective for this array. In terms of the remove method, the 2nd method was the least efficient, and resulted in the most moves.



```
1st Method: Add Item: 0 compares and 0 moves
2nd Method: Add Item: 2 compares and 2 moves
3rd Method: Add Item: 11 compares and 9 moves
1st Method: Remove Item: 1 compares and 2 moves
2nd Method: Remove Item: 1 compares and 5 moves
3rd Method: Remove Item: 1 compares and 0 moves
List1: 00204983_____
List2: 2020191919_____
List3: 5__535560__95_
```