Table of Contents:

Lessons Learned…………..……………..…Page 1.

Test Cases………………………………..…….Page 2.

UML Diagram……………………………..….Page 4.

Lessons Learned:

I learned that a method can be thought of as an object for controlling the flow of data through a program, as well as an object for manipulating data. Recursion is a means of structuring methods to gain additional flow control, and to manipulate data in a functional (i.e. f(x) type function) manner. These structures can easily grow out of control but can also be mapped and tracked.

My solution for this project has a complexity of O(n^2):

* The two methods of the Packager class, packageFractions() and packageIntegers() both operate in O(n) complexity.
* Due to its need to sort as it places nodes, the add() function of MyNode has a maximum big-o of O(n^2). I was unable to find a way to perform this recursively at a complexity of O(n) or less.
* The traversal algorithms also operate at O(n) complexity.

I need to spend time gaining a deeper understanding of generics.

Test Cases:

Test Case 1 – Integers

1. Enter 8 6 7 5 3 0 9 into “Original List” text field.
2. Click the “Integer” radio button.
3. Click the “Perform Sort” button.
4. Verify no pop-up message is displayed.

Test Case 2 – Fractions

1. Enter 1/8 1/32 1/16 893/1000 into “Original List” text field.
2. Click the “Fraction” radio button.
3. Click the “Perform Sort” button.
4. Verify no pop-up message is displayed.

Test Case 3 – Non-numeric Input

1. Enter 8 6 7 @ 3 0 9 into “Original List” text field.
2. Click the “Integer” radio button.
3. Click the “Perform Sort” button.
4. Verify pop-up message is displayed with the message “Invalid Input: @”

Test Case 4 – Mal-formed Fraction

1. Enter 8/16 16/9 8/16/9 into “Original List” text field.
2. Click the “Fraction” radio button.
3. Click the “Perform Sort” button.
4. Verify pop-up message is displayed with the message “Invalid Input: 8/16/9”

Test Case 5 – Ascending Sort - Integer

1. Enter 8 6 7 5 3 0 9 into “Original List” text field.
2. Click the “Integer” radio button.
3. Click the “Ascending” radio button.
4. Click the “Perform Sort” button.
5. Verify the “Sorted List” text field reads 0 3 5 6 7 8 9

Test Case 6 – Descending Sort – Integer

1. Enter 8 6 7 5 3 0 9 into “Original List” text field.
2. Click the “Integer” radio button.
3. Click the “Descending” radio button.
4. Click the “Perform Sort” button.
5. Verify the “Sorted List” text field reads 9 8 7 6 5 3 0

Test Case 7 – Ascending Sort – Fraction

1. Enter 1/8 8/1 1/16 16/1 16/16 into “Original List” text field.
2. Click the “Fraction” radio button.
3. Click the “Ascending” radio button.
4. Click the “Perform Sort” button.
5. Verify the “Sorted List” text field reads 1/16 1/18 16/16 8/1 16/1

Test Case 8 – Descending Sort – Fraction

1. Enter 1/8 8/1 1/16 16/1 16/16 into “Original List” text field.
2. Click the “Fraction” radio button.
3. Click the “Descending” radio button.
4. Click the “Perform Sort” button.
5. Verify the “Sorted List” text field reads 16/1 8/1 16/16 1/8 1/16

UML Diagram:

