Kelly M McCutchen
CMSC 350
July 1, 2018

Project 3

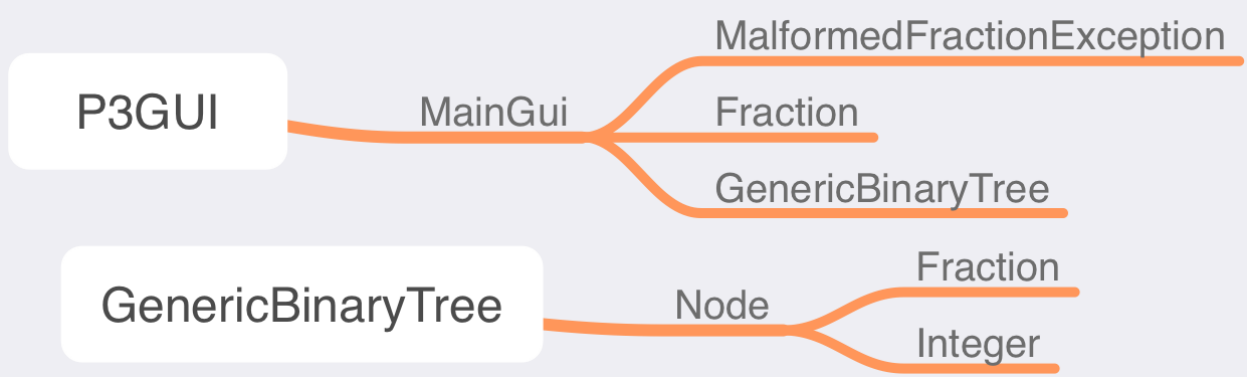
1. Assumptions, Main Design Decisions, and Error Handling

The main assumptions made during this program were for the formatting of fractions in terms of white space before/after the slashes. White space was accounted for as a delimiter and tokens are created using this principle. Also, I assumed all values would be in integer values (no decimal components).

In terms of design decision, the code follows generally with the guidelines given in the project description. I did implement a generic function for tree generation using generic ArrayLists. When creating the event handler code for 'Perform Sort' button, it seemed more straightforward to be handle a variety of list types instead of creating custom code for lists for both Fraction and Integer types. I did implement an additional constructor for the Fraction class to handle numerator/denominator input. The decision to handle strings outside of the class seemed like an appropriate way to process lists (i.e. tokenize string and determine validity, and if both numerator/denominator were appropriate for inputs, then created fraction and added to an ArrayList<Fraction> list.

Error handling is in keeping with the design document and creates messages for both MalformedFractionExceptions (custom error type) and NumberFormatExceptions during Integer parsing from strings.

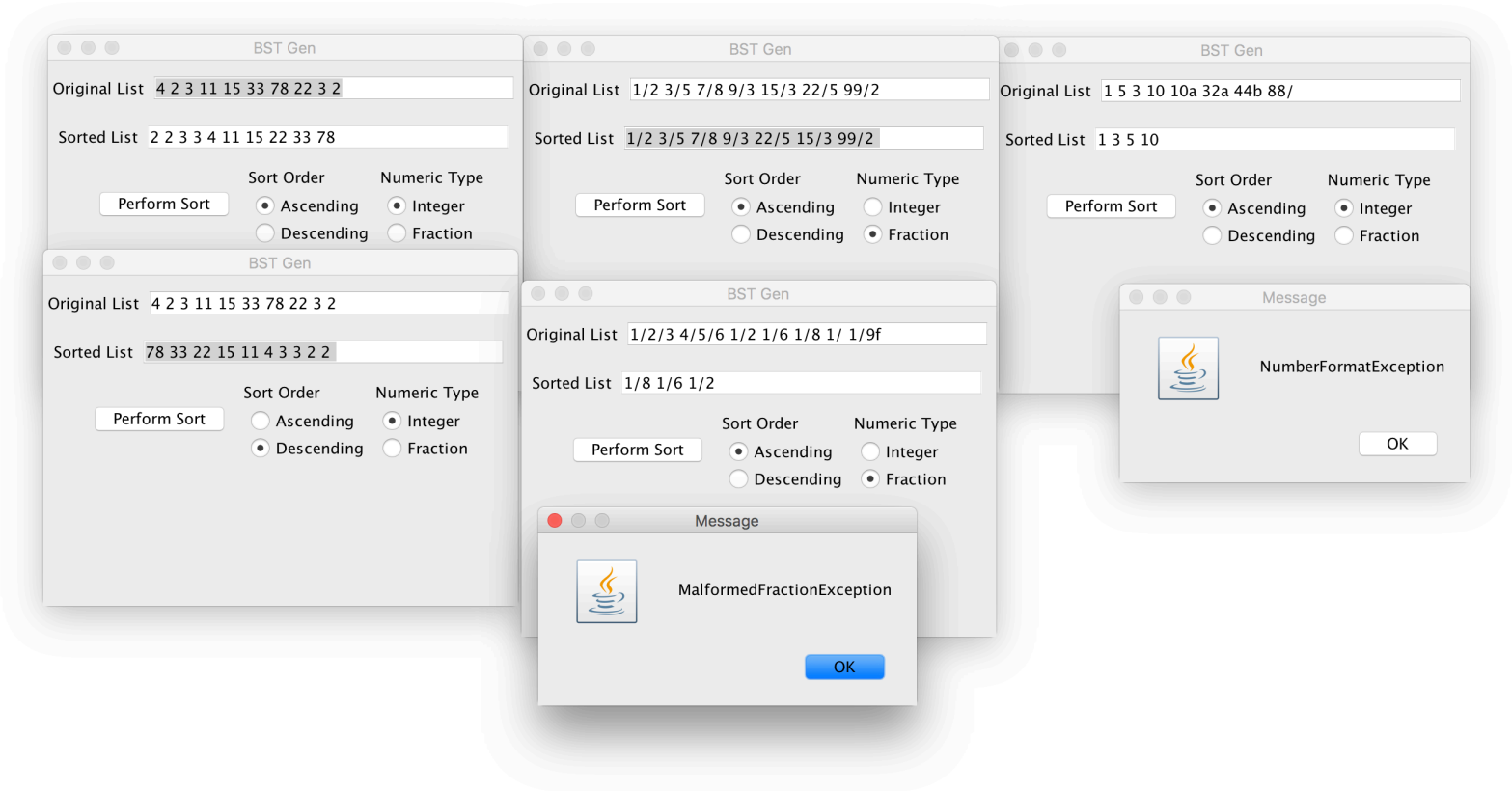
2. UML Diagram



3. Test Cases

Testing Aspect	Input Values	Expected Output	Actual Output	Pass/Fail
Basic Functionality of Integer Ascending	4 2 3 11 15 33 78 22 3 2	2 2 3 3 4 11 15 22 33 78	2 2 3 3 4 11 15 22 33 78	Pass
Basic Functionality of Integer Descending	4 2 3 11 15 33 78 22 3 2	78 33 22 15 11 4 3 3 2 2	78 33 22 15 11 4 3 3 2 2	Pass
Basic Functionality of Fraction Ascending	1/2 3/5 7/8 9/3 15/3 22/5 99/2	1/2 3/5 7/8 9/3 22/5 15/3 99/2	1/2 3/5 7/8 9/3 22/5 15/3 99/2	Pass
Basic Functionality of Fraction Descending	1/2 3/5 7/8 9/3 15/3 22/5 99/2	99/2 15/3 22/5 9/3 7/8 3/5 1/2	99/2 15/3 22/5 9/3 7/8 3/5 1/2	Pass
Integer Number Exceptions	1 5 3 10 10a 32a 44b 88/	1 3 5 10 (+4 NumberFormatExceptions)	1 3 5 10 (+4 NumberFormatExceptions)	Pass
MalformedFraction Exception	1/2/3 4/5/6 1/2 1/6 1/8 1/ 1/9f	1/8 1/6 1/2 (+MalformedFractionError + NumberExceptionError	1/8 1/6 1/2 (+MalformedFractionError + NumberExceptionError	Pass

4. Screenshots



5. Lessons Learned

This project helped to understand the sort algorithms involved. At the beginning of the course I really struggled to understand the concepts of tree generation and recursion in terms of path tracing from root to leaf values. However, this project helped me to have a more common sense understanding of the underlying subject material. Also, working with generics was a good practical exercise. I do not often utilize generics and creating generic classes and functions helped me to better think through design and program flow as well as object oriented paradigms for more concise and effective structures.