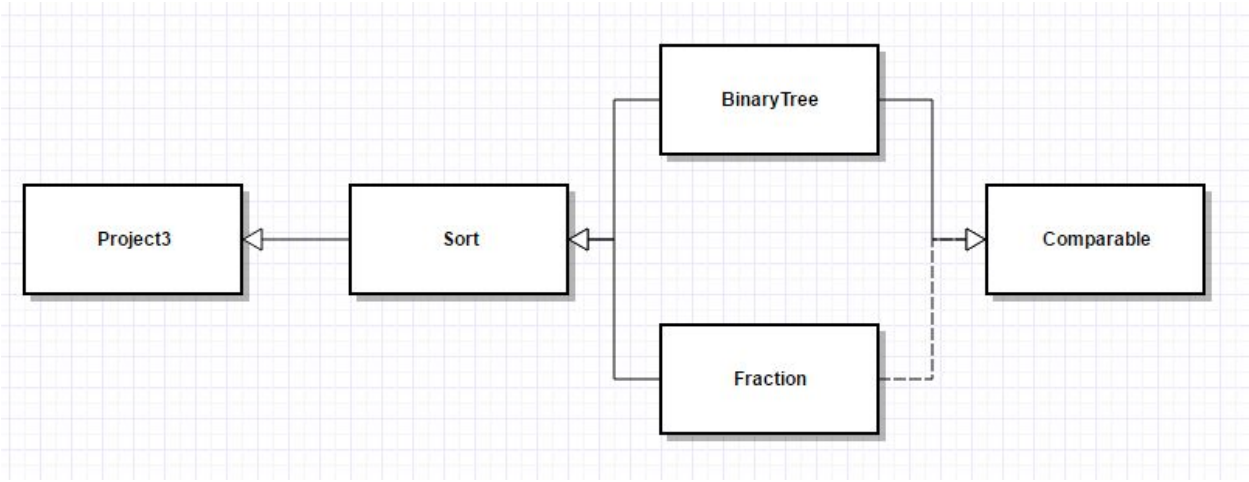
Project 3

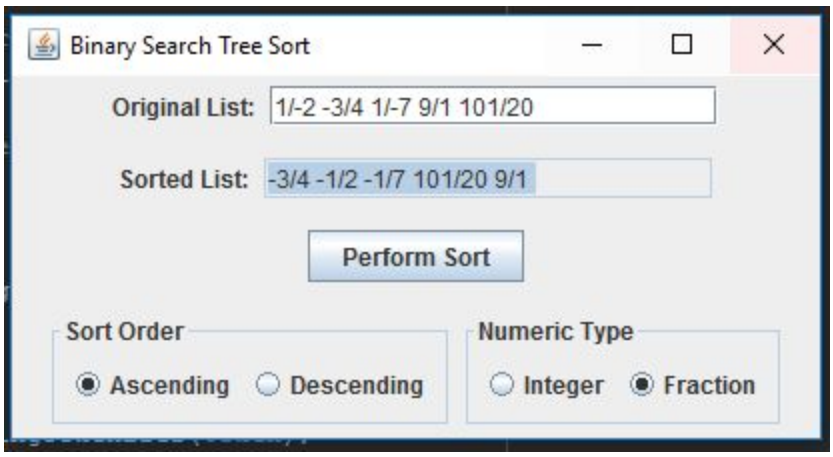
UML Diagram



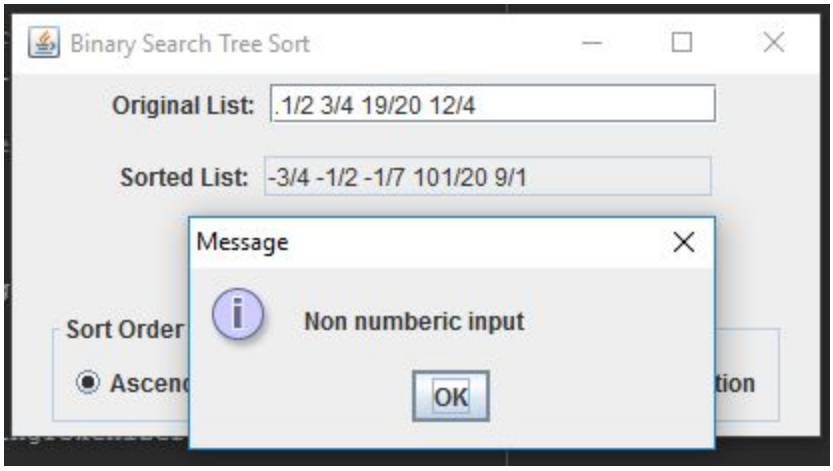
Test Table

Cases	TextField Input	RadioButton Input	Expected Output	Actual Output	Pass/Fail
1	1/-2 -3/4 1/-7 9/1 101/20	Ascending - Fraction	-3/4 -1/2 -1/7 101/20 9/1	-3/4 -1/2 -1/7 101/20 9/1	Pass
2	. 1/2 3/4 19/20 12/4	Ascending - Fraction	Error Message	Error Message	Pass
3	-1/-9 3/0 19/-20 -3/8	Ascending - Fraction	Error Message	Error Message	Pass
4	3/2 2/3 4/2 3/9 23/32	Descending - Fraction	4/2 3/2 23/32 2/3 3/9	4/2 3/2 23/32 2/3 3/9	Pass
5	33/2 4/3 -2/-4 abc 12/2 34/22 1/2	Descending - Fraction	Error Message	Error Message	Pass
6	1 5 2 6 3 4	Ascending - Integer	1 2 3 4 5 6	1 2 3 4 5 6	Pass
7	1032 293 3948 21 34 -1 -921	Ascending - Integer	-921 -1 21 34 293 1032 3948	-921 -1 21 34 293 1032 3948	Pass
8	32 56 32 98 12 &&& 2 56 98	Ascending - Integer	Error Message	Error Message	Pass
9	-23 9 564 -988 45 999	Descending - Integer	999 564 45 9 -23 -988	999 564 45 9 -23 -988	Pass
10	300 32 xyz 32 -12 92 56	Descending - Integer	Error Message	Error Message	Pass

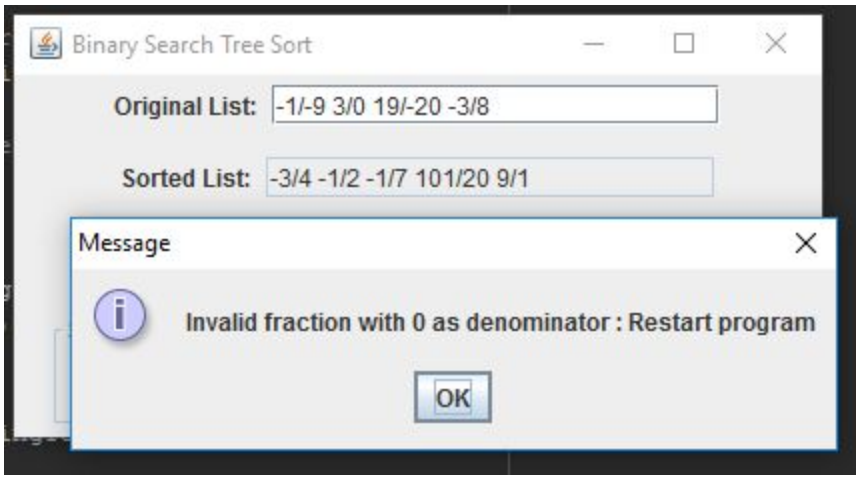
Case 1 - testing negative values in denominator as well as in numerator



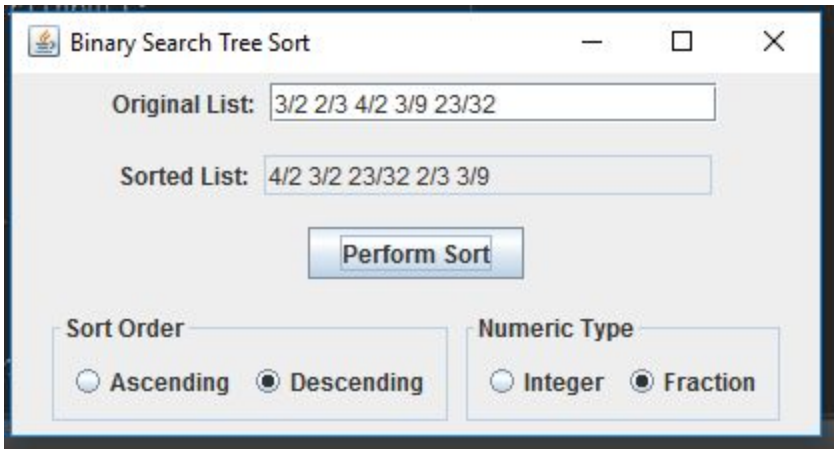
Case 2 - testing for non numeric error message



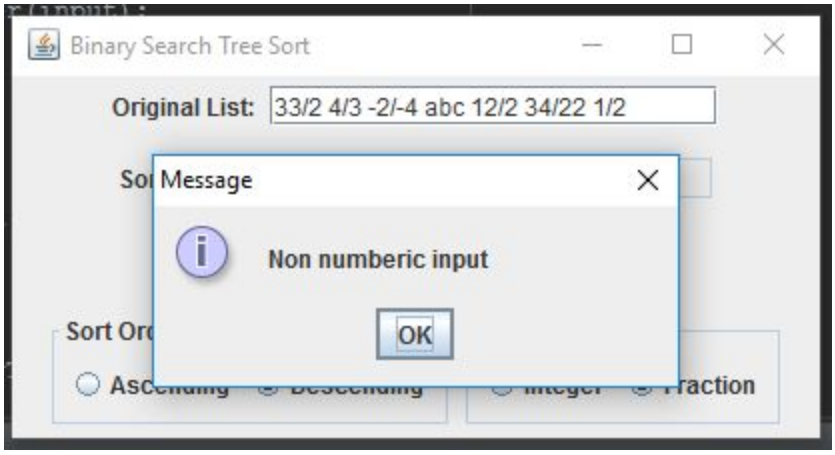
Case 3 - testing if error message will show when denominator is 0



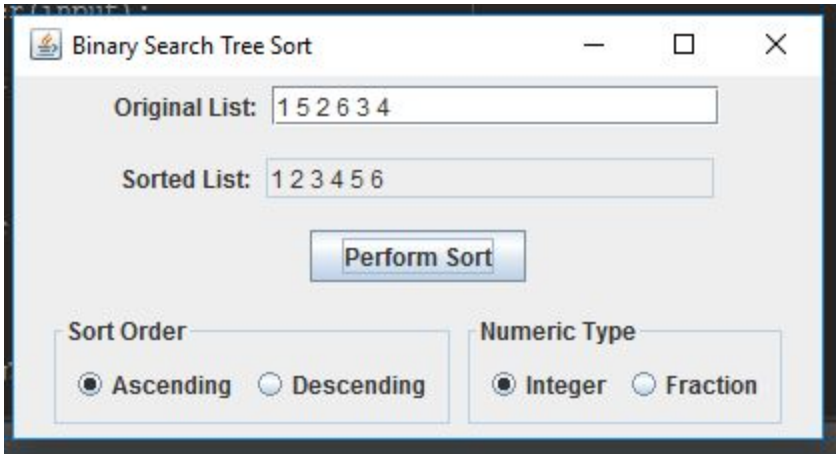
Case 4 - testing the sort with improper fractions



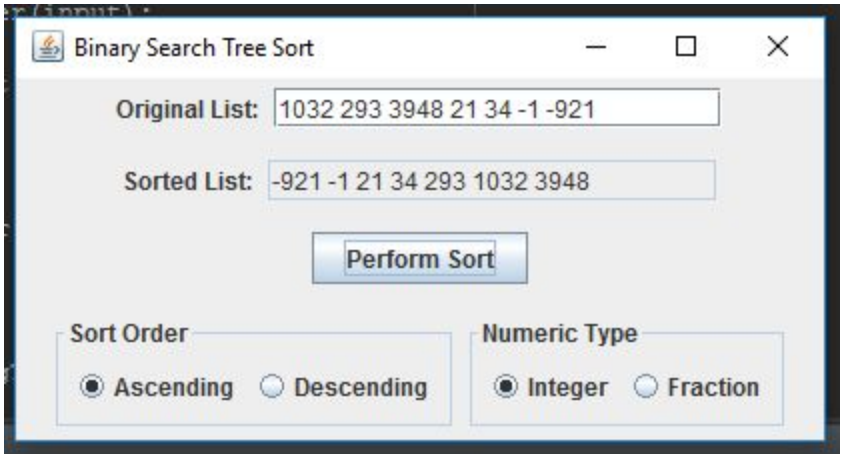
Case 5 - testing for non numeric error message with fractions



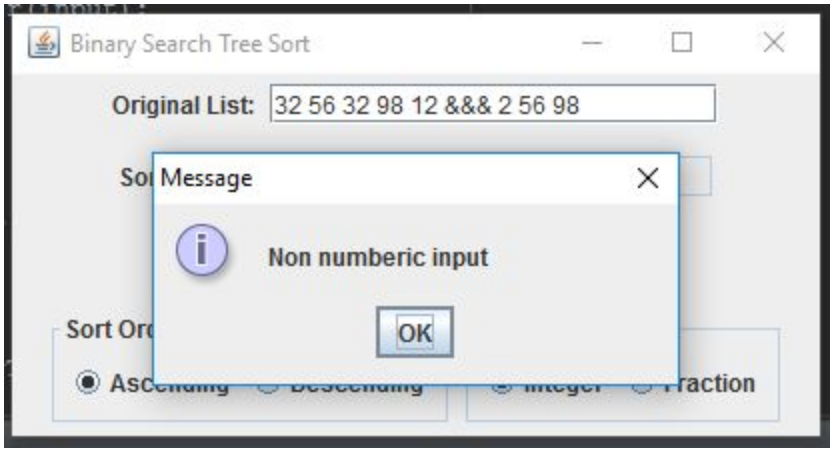
Case 6 - testing simple values to see if they will sort correctly



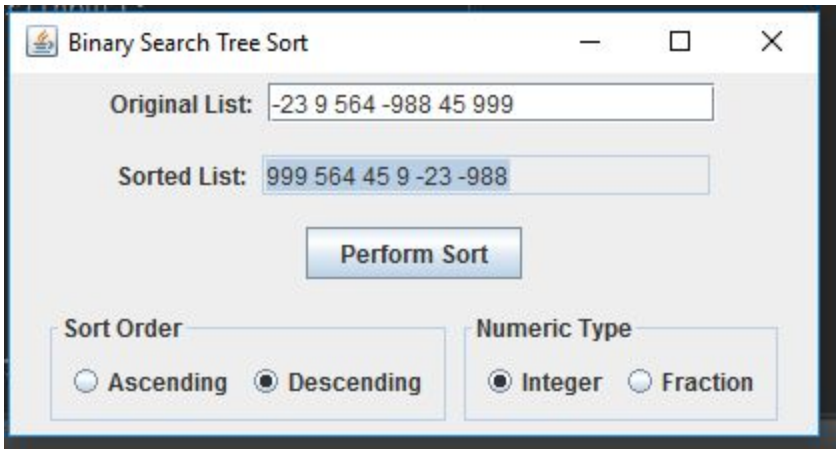
Case 7 - testing if radio buttons are working properly with larger numbers and mixed positive and negative values



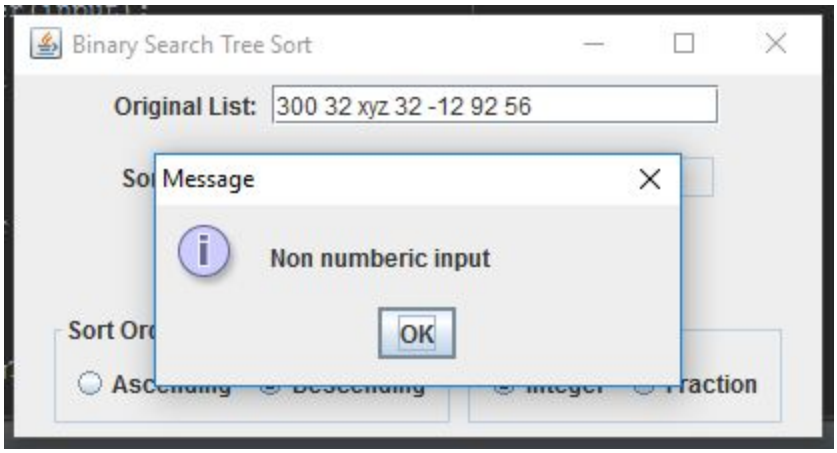
Case 8 - testing if non letter non numeric symbols cause error message



Case 9 - testing to see if negative and positive numbers sort properly



Case 10 - testing if letter characters cause error message



First off I want to say I did notice my grammatical error with numberic instead of numeric, but not until I finished all of my test cases, so please ignore this error. I learned a great deal about how generic classes work while doing this project as well as became more acquainted with binary search trees. In the previous project I largely copy and pasted from the readings and then messed around with the already written code while this time around I actually wrote the code out which helped a lot with understanding how BST's work. I did a few things like used StringTokenizers to separate and put back together values because at this point, I really get how to use this method well. I also used a series of if statements in the Fraction class's compareTo method to get the proper values when testing cases where a negative value was in the numerator, denominator as if both were negative values as well as if the denominator was 0 with successful results. I did code this to where if the denominator was 0 an error message would show and the program would close automatically, which might be seen as negative but I did include in the error message a note to restart the program. Overall this program functions as it should according to the project instructions as well as has the necessary classes as was required and I was able to do it with a lot less help than I needed for the previous two projects, which makes me feel like I am getting better, albeit I still have a lot to learn!