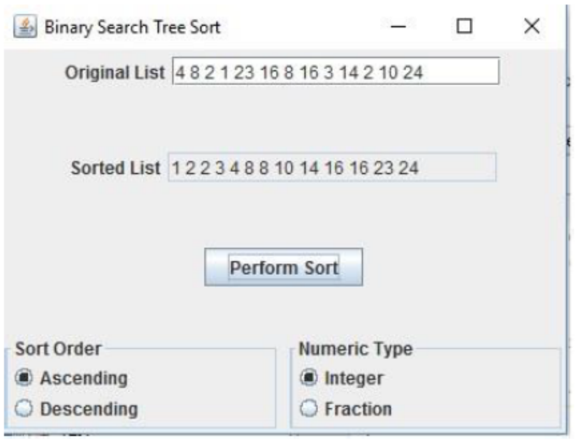
CMSC 350 Project 3

The third programming project involves writing a program that performs a sort by using a binary search tree. The program should be able to sort lists of integers or lists of fractions in either ascending or descending order. One set of radio buttons should be used to determine whether the lists contain integers or fractions and a second set should be used to specify the sort order. The main class should create the GUI shown below:



Original List 4 8 2 1 23 16 8 16 3 14 2 10 24

Sorted List 1 2 2 3 4 8 8 10 14 16 16 23 24

Sort Order: [✓ Ascending] Descending

Numeric Type: [✓ Integer] Fraction

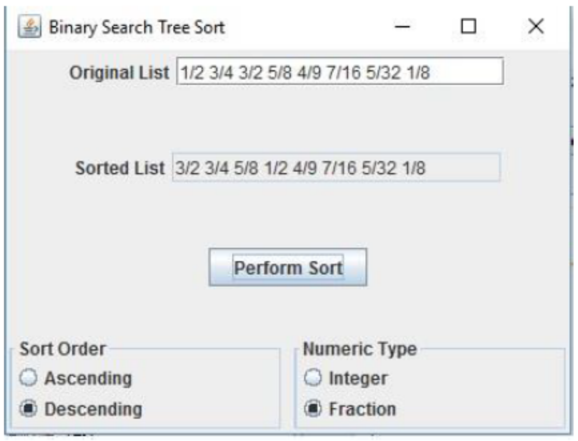
The GUI must be generated by code that you write. You may not use a drag-and-drop GUI generator.

Pressing the *Perform Sort* button should cause all the numbers in the original list to be added to a binary search tree. Then an inorder traversal of the tree should be performed to generate the list in sorted order and that list should then be displayed in the sorted list text field.

In addition to the main class that defines the GUI, you should have a generic class for the binary search tree. That class needs a method to initialize the tree, one that allows a new value to be inserted in the tree and one that performs an inorder tree traversal that generates and returns a string that contains the tree elements in sorted order. The insert method does not need to rebalance the tree if it becomes unbalanced. It should allow duplicate entries and it must be written using recursion. It is not necessary to have a method to delete a node from the tree nor one to check whether a particular value is in the tree.

The third class required for this project is one that defines fractions. It should have a constructor that accepts a string representation of a fraction and a `toString` method. It must implement the `Comparable` interface, which means a `compareTo` method is also required.

A second example of a run of this program is shown below that sorts fractions in descending order:


Binary Search Tree Sort

Original List: 1/2 3/4 3/2 5/8 4/9 7/16 5/32 1/8

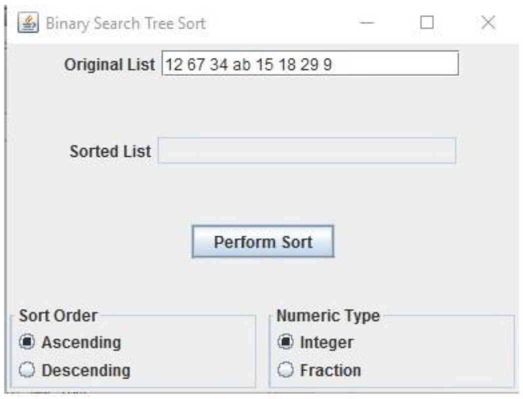
Sorted List: 3/2 3/4 5/8 1/2 4/9 7/16 5/32 1/8

Sort Order: Ascending [✓ Descending]

Numeric Type: Integer [✓ Fraction]

Note that fractions are to be written with a slash separating the numerator and denominator with no spaces on either side of the slash.

The only error checking required of this program is to check for nonnumeric input which includes improperly formatted fractions such as `3/4/8`. Such malformed fractions should cause a `NumberFormatExpression` to be thrown. The main class must catch these exceptions and display an appropriate error message as shown below:

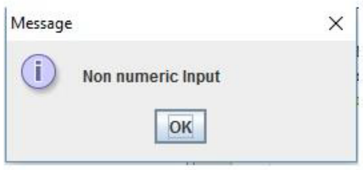

Binary Search Tree Sort

Original List: 12 67 34 ab 15 18 29 9

Sorted List:

Sort Order: [✓ Ascending] Descending

Numeric Type: [✓ Integer] Fraction


Message

Non numeric Input

You are to submit two files.

1. The first is a `.zip` file that contains all the source code for the project, which includes any code that was provided. The `.zip` file should contain only source code and nothing else, which means only the `.java` files. If you elect to use a package the `.java` files should be in a folder whose name is the package name.

2. The second is a Word document (PDF or RTF is also acceptable) that contains the documentation for the project, which should include the following:
    a. A UML class diagram that includes all classes you wrote. Do not include predefined classes. You need only include the class name for each individual class, not the variables or methods
    b. A test plan that includes test cases that you have created indicating what aspects of the program each one is testing
    c. A short paragraph on lessons learned from the project

Grading Rubric:

Criteria	Meets	Does Not Meet
Design	**5 points**	**0 points**
	GUI is hand coded and matches required design (1)	GUI is generated by a GUI generator or does not match required design (0)
	Includes generic class for binary search tree(2)	Does not include generic class for binary search tree (0)
	Insert method uses recursion (1)	Insert method does not use recursion (0)
	Contains Fraction class that implements Comparable (1)	Does not contain Fraction class that implements Comparable (0)
Functionality	**10 points**	**0 points**
	Correctly sorts all test cases in ascending order (2)	Does not correctly sort all test cases in ascending order (0)
	Correctly sorts all test cases in descending order (2)	Does not correctly sort all test cases in descending order (0)
	Correctly sorts all test cases involving integers (2)	Does not correctly sort all test cases involving integers (0)
	Correctly sorts all test cases involving fractions (2)	Does not correctly sort all test cases involving fractions (0)
	Generates error message for nonnumeric input (2)	Does not generate error message for nonnumeric input (0)
Test Cases	**5 points**	**0 points**
	Test cases include integers (2)	Test cases do not include integers (0)
	Test cases include fractions (2)	Test cases do not include fractions (0)

	Test cases include nonnumeric input (1)	Test cases do not include nonnumeric input (0)
Documentation	**5 points**	**0 points**
	Correct UML diagram included (2)	Correct UML diagram not included (0)
	Lessons learned included (2)	Lessons learned not included (0)
	Comment blocks with class description included with each class (1)	Comment blocks with class description not included with each class (0)
Overall Score	**Meets**	**Does not meet**
	16 or more	**0-15**