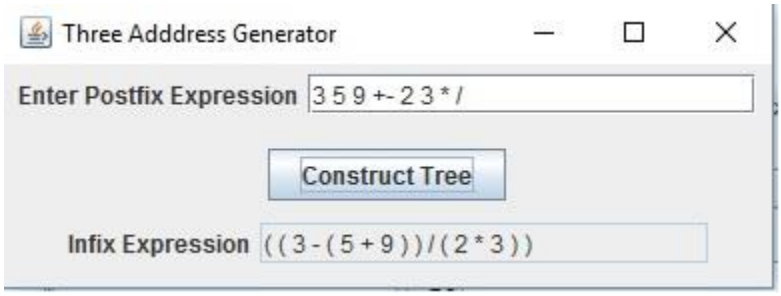
CMSC 350 Project 2

The second programming project involves writing a program that accepts an arithmetic expression of unsigned integers in postfix notation and builds the arithmetic expression tree that represents that expression. From that tree, the corresponding fully parenthesized infix expression should be displayed and a file should be generated that contains the three address format instructions. This topic is discussed in the week 4 reading in module 2, section II-B. The main class should create the GUI shown below:



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

Pressing the *Construct Tree* button should cause the tree to be constructed and using that tree, the corresponding infix expression should be displayed and the three address instruction file should be generated.

The postfix expression input should not be required to have spaces between every token. Note in the above example that `9+-` are not separated by spaces.

The above example should produce the following output file containing the three address instructions:

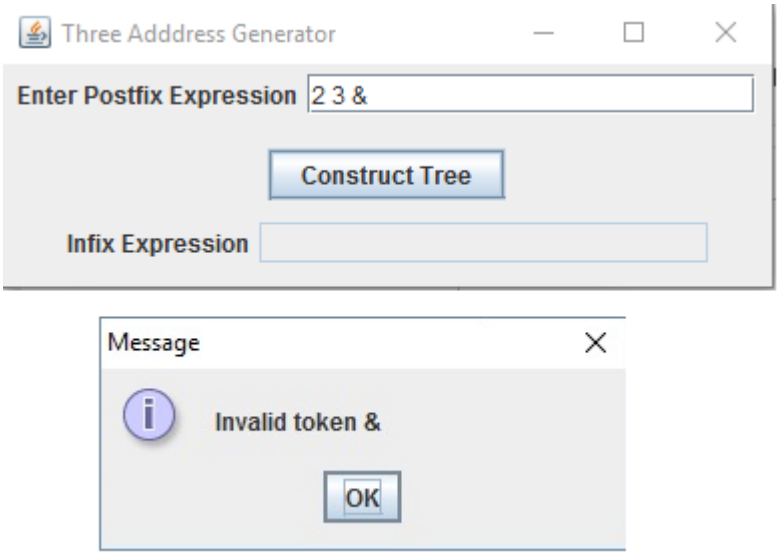
```
Add R0 5 9
Sub R1 3 R0
Mul R2 2 3
Div R3 R1 R2
```

It is not necessary to reuse registers within an expression as shown in module 2, section II-B, and you can assume there are as many available as needed. Each new expression should, however, begin using registers starting at `R0`.

Inheritance should be used to define the arithmetic expression tree. At a minimum, it should involve three classes: an abstract class for the tree nodes and two derived classes, one for operand nodes and another for operator nodes. Other classes should be included as needed to accomplish good object-oriented design. All instance data must be declared as private.

You may assume that the expression is syntactically correct with regard to the order of operators and operands, but you should check for invalid tokens, such as characters that are not valid operators or operands such as `2a`, which are not valid integers. If an invalid token is detected a

`RuntimeException` should be thrown and caught by the main class and an appropriate error message should be displayed. Below is an example:



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