Caculator1.6 Technical Document

Portfolio Activity 1.7

Michael Evans P288106

2017

Contents

[Data Structures 1](#_Toc499825475)

[Algorithms 1](#_Toc499825476)

[Algebraic Library 1](#_Toc499825477)

[Trigonometric Library 2](#_Toc499825478)

[Arithmetic Library 2](#_Toc499825479)

[Main Function of Calculator1.6 3](#_Toc499825480)

[Testing Procedure Recommendation 4](#_Toc499825481)

[Error Handling techniques 4](#_Toc499825482)

[Error Handling strategies: 4](#_Toc499825483)

[Recommendations 4](#_Toc499825484)

[GitHub account 4](#_Toc499825485)

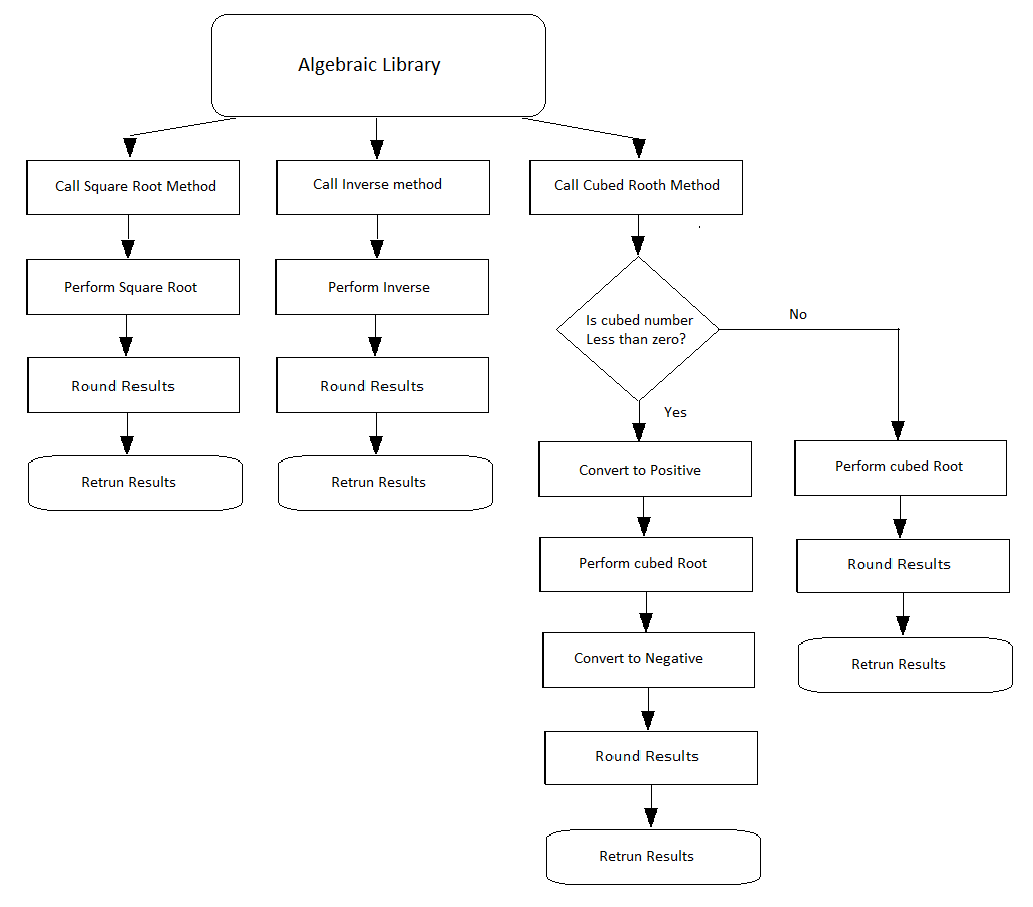
# Data Structures

|  |  |  |
| --- | --- | --- |
| Data Structures | | |
| **Name** | **Type** | **Purpose** |
| total | double | stores previous number entered |
| totalB | double | stores recent number entered. |
| pending | sting | stores string of last entered operation |

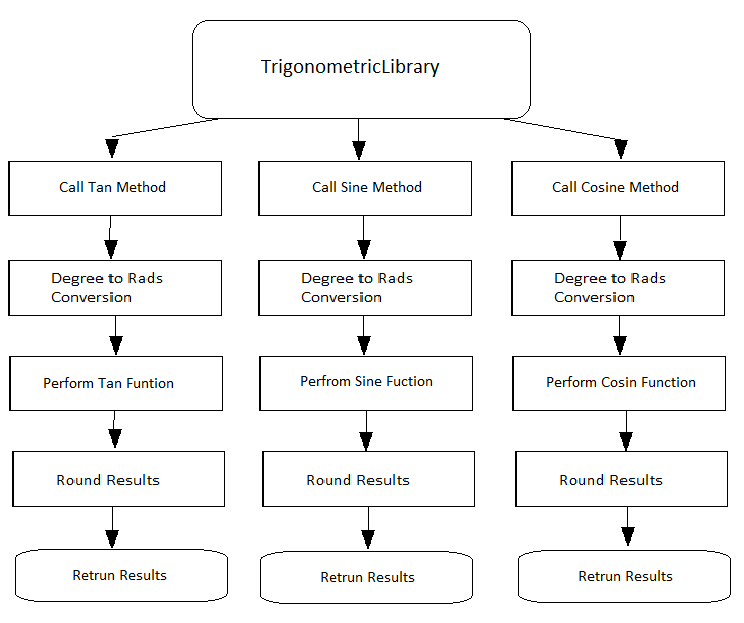
# Algorithms

Below are a series of flow diagrams show the libraries and Calculator programme.

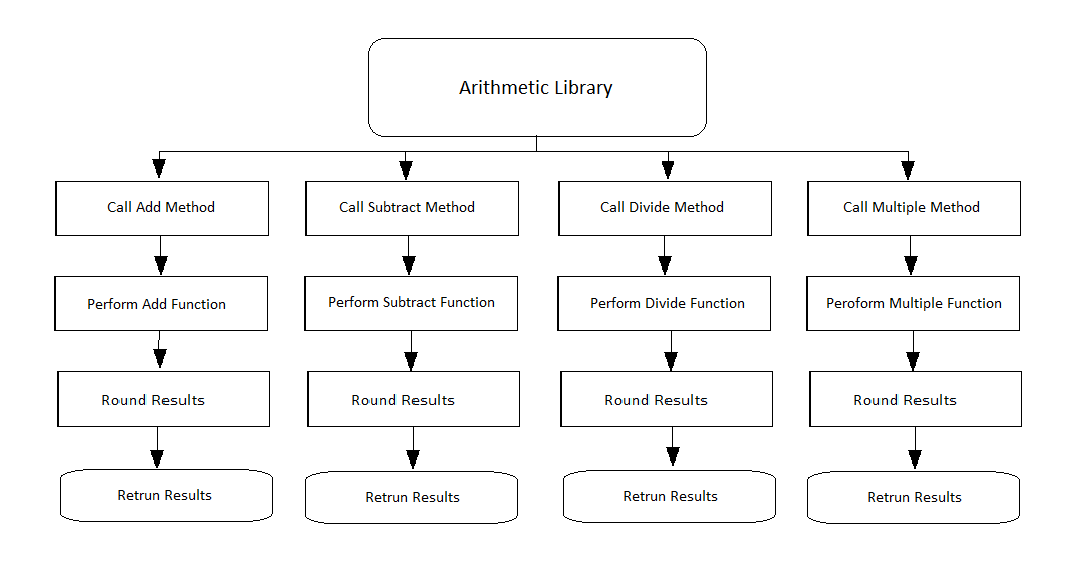
## Algebraic Library



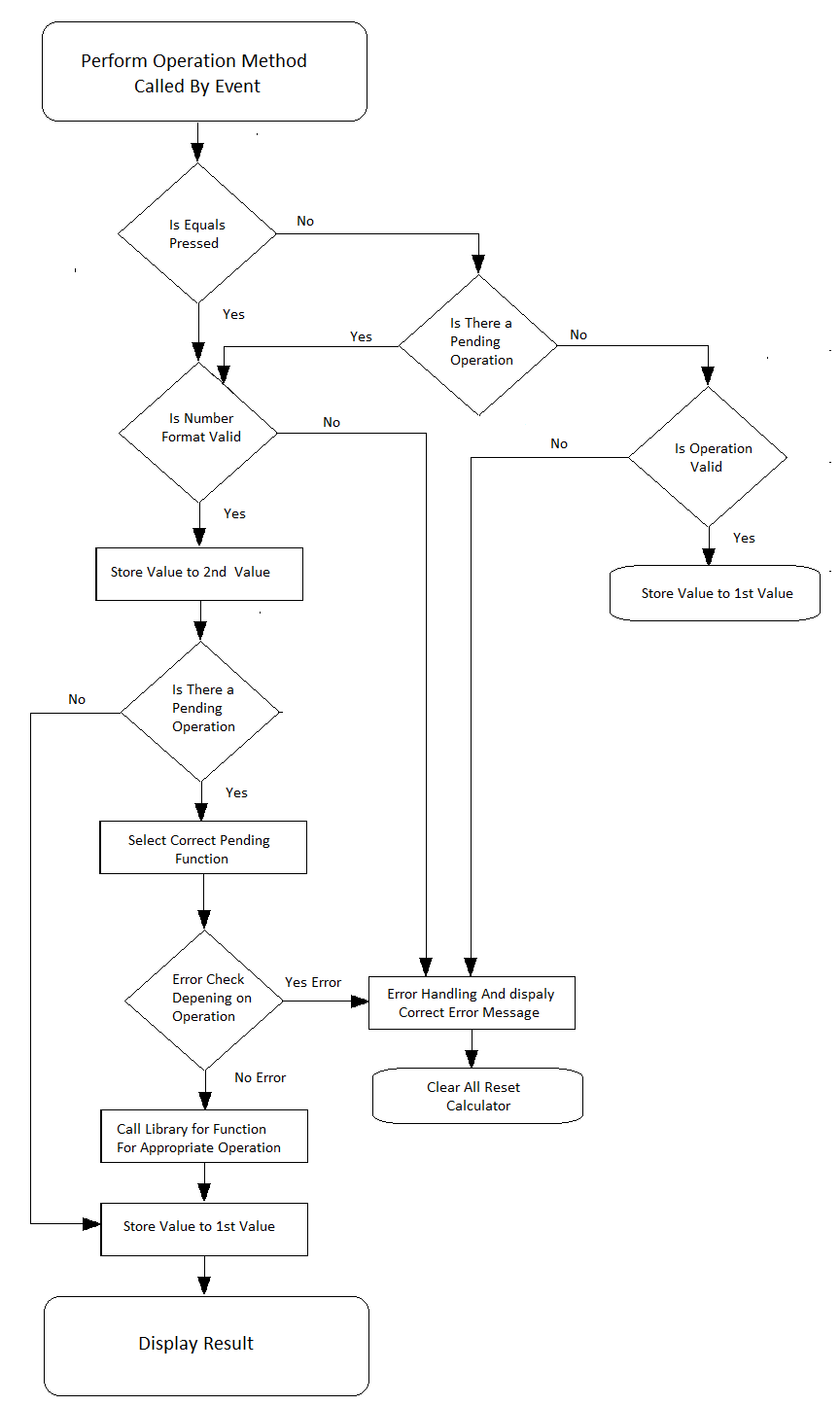
## Trigonometric Library



## Arithmetic Library



## Main Function of Calculator1.6



# Testing Procedure Recommendation

Due to the small nature of this application a complex testing procedure would be unnecessary. In stead a kind of ad-hoc style test would be best were the programmer test the feature adjuring the development of the software.

A simple modular Testing approach would the most suitable approach after development with each library being individually tested to ensure that each of the libraries are working as intended then with functioning libraries the programme itself should be tested. This has the advantage of simplifying the testing as any problems that are occurring will be largely due to the programme itself since the libraries have already been tested. The compartmental nature of this type of testing also allow for testing and development to be run in parallel, allowing for quicker delivery of the application.

# Error Handling techniques

Three main types of errors that can occur, these are as follows:

* Compile time errors such as incorrectly composed code, not really an issue as handled by complier.
* Runtime Error occur when machine itself in unable to execute the instruction for example.
* Logic Error these are the most difficult error to deal with as the complier and the running system will not detect these errors.

## Error Handling strategies:

One strategy is to use Exception too deal with run time errors that should generally steam only from incorrect user use, such as trying to divide a number by zero. The error handling occurs within the programme not within the library since it is the programme that deals with the user. Logic error will have to dealt with in the planning stage such developing the Pseudo code and software testing. Compile time errors are not a concern since they are dealt with by the compiler.

# Recommendations

The application could be extended with an update feature to help extend its features. This update feature would allow users to download any necessary updates as they require. This could also bring the users to publisher’s web site, so the users could select update or other paid products. Thus, it could be used to draw visitors to the site.

Features that could added include an ability to do the power of capability such as square or a more general to the power of any that can be found on most scientific calculators. The ability to do factorials, log and natural logs. These feature upgrades would simply require extending exiting libraries or add new libraries. Also, the calculator general display could be upgraded to show an actuarial equation on the screen simular to a graphic calculator. This would however require significant write of the programme.

# GitHub account

https://github.com/Mike27JE/Calculator1.6