GUI Calculator 1.7

Technical documentation

Jason Bignell

Contents

[Data Structures 3](#_Toc510443142)

[ Algorithms 3](#_Toc510443143)

[Class: Program 3](#_Toc510443144)

[Class: Calculator 3](#_Toc510443145)

[Error handing techniques 4](#_Toc510443146)

[Recommended testing procedure (how should this software be tested before commercial release) 5](#_Toc510443147)

[ Recommendations on upgrades and future enhancements 5](#_Toc510443148)

# Data Structures

Name, type and purpose of each variable (in a table format)

|  |  |  |
| --- | --- | --- |
| Name | Type | Purpose |
| subTotal | double | Hold the total for calulations |
| outPut | double | Holds the final total for placing in the textbox |
| plusButtonClicked | boolean | Used to tell which button was last clicked |
| minusButtonClicked | boolean | Used to tell which button was last clicked |
| divideButtonClicked | boolean | Used to tell which button was last clicked |
| multiplyButtonClicked | boolean | Used to tell which button was last clicked |

# Algorithms

## Class: Program

Method: Main

Show the form

# Class: Calculator

Method: Calculator() (Constructor)

Initialize the form

Methods: 0-9 buttons + . (Dot) button

When clicked

Display.text = itself + the text of the button clicked

Methods: btnClear Click

When Clicked

Clear the output textbox

Method: btnSqrt Click

When clicked

If the number is numeric

Add it to a temp double

If the temp double is greater then or equal to 0

Call the Square root function on the temp varible and assign it to the display

Else

Tell the user we need a positive number

Set the display to 0

Method: btnCubeRT Click

When clicked

If the number is numeric

Add it to a temp double

If the temp double is greater then or equal to 0

Call the Cube root function on the temp varible and assign it to the display

Else

Tell the user we need a positive number

Set the display to 0

Method: btnInv Click

When clicked

If the number is numeric

Add it to a temp double

If the temp double is greater then or equal to 0

Call the Inverse function on the temp varible and assign it to the display

Else

Tell the user we need a positive number

Set the display to 0

Method: btnTan Click

When clicked

If the number is numeric

Add it to a temp double

If the temp double is greater then or equal to 0

If the temp varible is not 90

Call the Tan function on the temp varible and assign it to the display

Else

Inform the user that Tan(90) is impossible

Else

Tell the user we need a positive number

Set the display to 0

Method: btnSin Click

When clicked

If the number is numeric

Add it to a temp double

If the temp double is greater then or equal to 0

Call the Sine function on the temp varible and assign it to the display

Else

Tell the user we need a positive number

Set the display to 0

Method: btnCos Click

When clicked

If the number is numeric

Add it to a temp double

If the temp double is greater then or equal to 0

Call the Cos function on the temp varible and assign it to the display

Else

Tell the user we need a positive number

Set the display to 0

## Error handing techniques

The program is coded is such a way as to avoid throwing exceptions. Rather, it will inform the user of what has occurred, and will abort the activated function

# Recommended testing procedure

The procedure for testing the application is as follows:

1. Open/run the program

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | First Value | Required Button | Second Value |
| Inputting Letters | A | + | b |
| Test Number Buttons and clear | 123456789.0 | Clear | 0.987654321 |
| SQRT Button | 9 | SQRT | n/a |
| Negative Input | -1 | SQRT | n/a |
| CubeRT Button | 27 | CubeRT | n/a |
| Negative Input | -1 | CubeRT | n/a |
| Inv Button | 8 | Inv | n/a |
| Negative Input | -1 | Inv | n/a |
| Tan Button | 45 | Tan | n/a |
| Negative Input | -1 | Tan | n/a |
| Sin Button | 45 | Sin | n/a |
| Negative Input | -1 | Sin | n/a |
| Cos Button | 0 | Cos | n/a |
| Negative Input | -1 | Cos | n/a |
| + Button | 1 | + | 2 |
| Incorrect Input | -1 | + | 2 |
| - Button | 4 | - | 3 |
| Negative Input | -1 | - | 3 |
| / Button | 6 | / | 3 |
| Negative Input | -1 | / | 5 |
| \* Button | 8 | \* | 7 |
| Negative Input | -1 | \* | 7 |

1. Enter the first value
2. Press the required button
3. Enter the second Value (If required)
4. Press the Equal button
5. Verify results
6. Press the Clear Button

# Recommendations on upgrades and future enhancements

Possible future Enhancements:

1. Undo Functions

It would be helpful to have a command history to enable functions such as undo and redo

2.Typing directly

It would also be helpful to be able to type directly into the output box, however this runs into the issue of letters entering the data used for calculations. This could be worked around by making the buttons activate on keypress but this was beyond the scope of the project

3.More mathematical functions

More mathematical functions are always nice to have in a calculator

4. working with negatives

Currently the calculator refuses to work on negative numbers. This is how it was intialy designed. However the ability to use negative numbers in calculations could prove a benefit.