**Portfolio Activity 1.7**

**Technical Calculator**

(Technical Report)

**Name:**

Renee Alsop

**ID:**

P303319

**Date:**

02/11/2017

Contents

Data Structures ..................................................................3

Algorithms ..........................................................................4

OnClick Methods...................................................4

Other Methods......................................................8

Error Handling Techniques.................................................9

Recommended testing Procedure.......................................9

Recommendations on upgrades...........................................11

GitHub Infomation and URL.................................................11

**Data Structures**

|  |  |  |
| --- | --- | --- |
| Variable Name | Variable Type | Variable Purpose |
| totalA | double | Storing first input |
| totalB | double | Storing second input |
| rtnDbl | double | Double returned from 3rd party library calculations |
| plusClicked | boolean | Determines is the plus button has been clicked |
| minusClicked | boolean | Determines is the subtraction button has been clicked |
| divClicked | Boolean | Determines is the division button has been clicked |
| multiClicked | Boolean | Determines is the multiply button has been clicked |
| PwrRTClicked | Boolean | Determines is the power root button has been clicked |
| pwrClicked | Boolean | Determines is the power button has been clicked |
| IsNeg | Boolean | Determines is the minus button has been clicked without previous input to discern if proceeding number should be negative |
| num | double | Used as local double for equations where only 1 input is required |

**Algorithms**

**Button Click Methods**

The following method is for btAdd, btDiv and btMult

button Click

Click2();

parse textbox text to totalA

check if IsNeg is true, if so set totalA to negative number.

set booleans values

check for empty string

textbox focus

Is Null Not Null

The following method is for btSub

btSub\_Click

Click2();

parse textbox text to totalA

check if IsNeg is true, if so set totalA to negative number.

set boolean values

textbox focus

IsNeg -> true

check for empty string

is Null Not Null

methods for btEquals

btEquals\_Click

Click2();

textbox focus

if plus clicked

null

else if minus clicked

check for empty string

parse textbox text to totalB

check if IsNeg is true, if so set totalB to negative number.

Not Null

else if div clicked

Call relative 3rd Party Lib method, passing totalA and totalB

else if multi clicked

check if rtnDbl == error code

Handle Zero value with UHOH() method

else if PwrRT clicked

Code Returned

else if pwr clicked

No Code Returned

display returned value in textbox reset boolean values

reset totalA

methods for algebra functions - btSQRT, btCubeRT, btSqr, btInv

Button Click

Click2();

check for empty string

Handle Error UHOH()

textbox focus

yes

No yes

check if value is zero

check if value is negative

No

yes

Use 3rd party library to calculate result

Display result

Handle Error

Clear bool

method for algebra function btPwrRT

button Click

Handle Error

Click2();

yes

check if IsNeg == true

check for empty string

textbox focus

No

parse textbox text to totalA

check if IsNeg is true, if so set totalA to negative number.

set boolean values

method for algebra function btPower

button Click

Click2();

parse textbox text to totalA

check if IsNeg is true, if so set totalA to negative number.

set boolean values

check for empty string

null not null

textbox focus

**Other Methods**

method for UhOh()

select random int between 0 and 5

play sad trombone sound

show message in message box from string array selected by random int as index

method for Click1(), Click2()

use media sound player to play sound

method for clearBool()

set all boolean values to false

**Error Handling Techniques**

There are several ways errors are handled in this calculator.

Zero Division Error

This is reported to the user via the UhOh() method, however it is returned to the main program from the external libraries by means of a code (0.0000000002) this code is to 10 decimal places, where as all other functions are returned at a maximum of 9 decimal places to fit the display. The UhOh() method, plays a sound effect, generates a random integer between 0 and 5, then uses that integer as the index reference from a string array with several different zero division responses and displays said string in a message box to the user.

Undefined Error

This is reported to the user via a message box as invalid / undefined. This error is only used in the trigonometry methods.

Impossible Error

This is reported to the user via a message box as Impossible in the event the user attempts to find the root of a negative number.

Null entry

At any point, if the user fails to enter input before pressing an action button, it will just focus on the display to prompt the user to input.

**Recommend testing procedure**

The recommended procedure for testing the calculator, would include attempting all functions with both possible and impossible equations including, but not limited to, dividing zero (try it several times for fun), attempting to find the root of a negative number, Tan 90 ect... here are some examples from testing already done:

# Arithmetic

|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual / Comment |
| Addition | | |
| -5 + -5 | -10 | -10 |
| -5 + 0 | -5 | -5 |
| 0 + 5 | 5 | 5 |
| 5 + 5 | 10 | 10 |
| Subtraction | | |
| -5 - -5 | 0 | 0 |
| 5 - 0 | 5 | 5 |
| 0 - 5 | -5 | -5 |
| 10 - 15 | -5 | -5 |
| Multiplication | | |
| 5 \* 5 | 25 | 25 |
| 5 \* 0 | 0 | 0 |
| 0 \* 5 | 0 | 0 |
| 5 \* -5 | -25 | -25 |
| Division | | |
| 10 / 5 | 2 | 2 |
| 10 / 3 | 3.3333333 | 3.33333333 |
| 10 / 0 | NP | Div Zero Error Msg |
| 0 / 10 | NP | Div Zero Error Msg |
| 10 / -2 | -5 | -5 |

# Trigonometric

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Expected | | | Actual / Comment |
| Tan | | | | |
| 0o | | 0 | 0 | |
| 30o | | 0.577350269 | 0.577350269 | |
| 45o | | 1 | 1 | |
| 60o | | 1.732050808 | 1.732050808 | |
| 90o | | NP | Undefined Error Msg | |
| Sin | | | | |
| 0o | | 0 | 0 | |
| 30o | | 0.5 | 0.5 | |
| 45o | | 0.707106781 | 0.707106781 | |
| 60o | | 0.866025404 | 0.866025404 | |
| 90o | | 1 | 1 | |
| Cos | | | | |
| 0o | | 1 | 1 | |
| 30o | | 0.866025404 | 0.866025404 | |
| 45o | | 0.707106781 | 0.707106781 | |
| 60o | | 0.5 | 0.5 | |
| 90o | | 0 | 0 | |

# Algebraic

|  |  |  |
| --- | --- | --- |
| Test | Expected | Actual / Comment |
| Square Root | | |
| 0 | NP | Div Zero Error Msg |
| 1 | 1 | 1 |
| 2 | 1.414213562 | 1.414213562 |
| -1 | NP | Impossible Error Msg |
| Cube Root | | |
| 0 | NP | Div Zero Error Msg |
| 1 | 1 | 1 |
| 2 | 1.25992105 | 1.25992105 |
| -1 | NP | Impossible Error Msg |
| Inverse | | |
| 0 | NP | Div Zero Error Msg |
| 1 | 1 | 1 |
| 2 | 0.5 | 0.5 |
| -1 | -1 | -1 |

**Recommendations on upgrades**

Version 3 should allow for the use of key press events at least for numerical input and basic arithmetic (/,\*,-,+) as well as use of the enter key to action the equals methods. Also to see if version 2.0 layout can be improved upon. Create a function on the menu bar for user to be able to enable or disable the sound effects.

**GitHub URL**

Technical calculator and this document can be found at <https://github.com/DrRaz77/Technical-Calculator.git>