11/10/2017

Ashley Carr

Polytechnic West

Portfolio Assessment AT1.7

By Ashley Carr

URL for GitHub:

(<https://github.com/Scobie1992/Calculator>)

**Contents**

[Data Structures 2](#_Toc498073121)

[Algorithms 3](#_Toc498073122)

[Recommended Testing Procedure 18](#_Toc498073123)

[Recommendations 20](#_Toc498073124)

# **Data Structures**

This is the name, type and purpose of each variable:

|  |  |  |
| --- | --- | --- |
| Name | Type | Purpose |
| font | Font | Sets the value for the font type, size and style |
| A | double | Holds the value for the first input |
| B | double | Holds the value for the second input |
| result | double | Holds the value for the calculation made with the first and second input |
| test | double | Tests whether the input is a double or string |
| plusButtonClicked | bool | Keeps track of which basic operator has been clicked |
| minusButtonClicked | bool | Keeps track of which basic operator has been clicked |
| divideButtonClicked | bool | Keeps track of which basic operator has been clicked |
| multiplyButtonClicked | bool | Keeps track of which basic operator has been clicked |
| error | string | If trying to divide by zero, changes the string and prints that message |
| num | double | Takes the value from the txtDisplay and parses it to a double to run the calculation |
| sqrt | double | Uses the Algebraic.Power.SquareRoot dll library to find the square root of the given number |
| cube | double | Uses the Algebraic.Power.CubeRoot dll library to find the cube root of the given number |
| inv | double | Uses the Algebraic.Power.Inverse dll library to find the inverse of the given number |
| sine | double | Uses the Trigonometric.Angle.Sine dll library to find the sine of the given number in degrees |
| cosine | double | Uses the Trigonometric.Angle.Cosine dll library to find the cosine of the given number in degrees |
| resultTangent | double | If the tangent is greater than 1000 or less than -1000 it displays an error message |
| tangent | double | Uses the Trigonometric.Angle.Tangent dll library to find the tangent of the given number in degrees |

# **Algorithms**

Flow diagrams for each method and error handling techniques:

Basic Operator Buttons:

Is txtDisplay empty?

Test txtDisplay is a double

MessageBox shows

“Please enter a Number”

MessageBox shows

“Error no input”

False

True

False

txtDisplay.Text is parsed as double A,

txtDisplay is cleared,

Correct boolean clicked is set true,

Buttons back colour is set to light green.

True

btnPlus\_Click, btnMinus\_Click, btnDivide\_Click, btnMultiply\_Click

Equals Button:

btnEquals\_Click

Is txtDisplay empty?

MessageBox shows

“Error no input”

False

True

MessageBox shows

“Please enter a Number”

False

Test txtDisplay is a double

True

Sets the back colours of the basic operators to white

txtDisplay.Text is parsed as double B

result is set to Arithmetic.Basic.Add(A,B)

True

plusButtonClicked

else if

Continue onto next page:

txtDisplay.Text is parsed as double B

result is set to Arithmetic.Basic.Sub(A,B)

True

minusButtonClicked

else if

txtDisplay.Text is parsed as double B

True

divideButtonClicked

else if

B == 0

True

else

Arithmetic.Basic.Div(A,B)

Error string is set to “Cannot Divide by Zero”

txtDisplay.Text is parsed as double B

result is set to Arithmetic.Basic.Mult(A,B)

True

multiplyButtonClicked

if

txtDisplay.Text is set to result.ToString()

True

error == “ ”

False

double A is set to 0.0

txtDisplay.Font is set to font

txtDisplay.txt is set to error

Arithmetic Class Library:

Project - Arithmetic

Method - Add

Takes in doubles a and b

Returns (a + b)

Class - Basic

Takes in doubles a and b

Returns (a – b)

Method - Sub

Method - Div

Takes in doubles a and b

Returns ((double)a / b)

Takes in doubles a and b

Returns (a \* b)

Method - Mult

Square Root Button:

btnSqrt\_Click

Is txtDisplay empty?

MessageBox shows

“Error no input”

False

True

MessageBox shows

“Please enter a Number”

False

Test txtDisplay is a double

True

Sets the back colour of the Square Root button to light green

txtDisplay.Text is parsed as double num

if

Continue onto next page:

txtDisplay.Text is set to SquareRoot(num).ToString()

This method takes in double x and returns double sqrt through

Algebraic.Power.SquareRoot(x)

True

num >= 0

False

txtDisplay.Font is set to font

txtDisplay.txt is set to “Number must be positive”

Cube Root Button:

btnCube\_Click

Is txtDisplay empty?

MessageBox shows

“Error no input”

False

True

MessageBox shows

“Please enter a Number”

False

Test txtDisplay is a double

Sets the back colour of the Cube Root button to light green

txtDisplay.Text is parsed as double num

txtDisplay.Text is set to CubeRoot(num).ToString()

This method takes in double y and returns double cube through Algebraic.Power.CubeRoot(y)

True

Inverse Button:

btnInv\_Click

Is txtDisplay empty?

MessageBox shows

“Error no input”

False

True

MessageBox shows

“Please enter a Number”

False

Test txtDisplay is a double

True

Sets the back colour of the Inverse button to light green

txtDisplay.Text is parsed as double num

if

Continue onto next page:

txtDisplay.Font is set to font

txtDisplay.txt is set to “Cannot Divide by Zero”

True

num == 0

False

txtDisplay.Text is set to Inverse(num).ToString()

This method takes in double i and returns double i through

Algebraic.Power.Inverse(i)

Algebraic Class Library:

Project - Algebraic

Class - Power

Method - SquareRoot

Takes in double x

double sqrt = Math.sqrt(x)

Returns sqrt

Method - Inverse

Takes in double i

double inv = 1 / i

Returns inv

Takes in double y

If y < 0.0 return -CubeRoot(-y)

Else returns Math.Pow(y, 1.0 / 3.0)

Method - CubeRoot

Sine Button:

btnSin\_Click

Is txtDisplay empty?

MessageBox shows

“Error no input”

False

True

MessageBox shows

“Please enter a Number”

False

Test txtDisplay is a double

Sets the back colour of the Sine button to light green

txtDisplay.Text is parsed as double num

txtDisplay.Text is set to Sine(num).ToString()

This method takes in double s and returns double sine through Trigonometric.Angle.Sine(s)

True

Cosine Button:

btnCos\_Click

Is txtDisplay empty?

MessageBox shows

“Error no input”

False

True

MessageBox shows

“Please enter a Number”

False

Test txtDisplay is a double

Sets the back colour of the Cosine button to light green

txtDisplay.Text is parsed as double num

txtDisplay.Text is set to Cosine(num).ToString()

This method takes in double c and returns double cosine through Trigonometric.Angle.Cosine(c)

True

Tangent Button:

btnTan\_Click

Is txtDisplay empty?

MessageBox shows

“Error no input”

False

True

MessageBox shows

“Please enter a Number”

False

Test txtDisplay is a double

Sets the back colour of the Tangent button to light green

True

txtDisplay.Text is parsed as double num

resultTangent is set to Tangent(num)

This method takes in double t and returns double tangent through Trigonometric.Angle.Tangent(t)

if

Continue onto next page:

txtDisplay.txt is set to “Invalid Input”

True

resultTangent > 1000 or resultTangent < -1000

False

txtDisplay.Text is set to resultTangent.ToString()

Trigonometric Class Library:

Project - Trigonometric

Class - Angle

Method - Sine

Takes in double s

double sineD = s \* (Math.PI / 180.0)

double sineR = Math.Sin(SineD)

double result = Math.Round(sineR, 15)

return result

Method - Cosine

Takes in double c

double cosineD = c \* (Math.PI / 180.0)

double cosineR = Math.Cos(cosineD)

double result = Math.Round(cosineR, 15)

return result

Method - Tangent

Takes in double t

double tangentD = t \* (Math.PI / 180.0)

double tangentR = Math.Tan(tangentD)

double result = Math.Round(tangentR, 15)

return result

# **Recommended Testing Procedure**

How this software should be tested before commercial release:

By the point that the application has been handed off to the testing team they should have a software development life cycle model that they are following. This should influence how regularly and how in-depth the testing is done. For example, a waterfall model will have a longer series of tests ran on each bit of the code after the whole application has been built, whereas in an agile model the code would be tested at each phase it has been built.

After the application has been built by the development team it should be then handed off to a testing team. The testing team would confirm that the application meets the original project requirements and functions as it should. Also running the application through the debugging mode to see what is happening at each point of the application to confirm everything works as it should and the variables are the correct values throughout the application. By confirming that the logic in the application is correct, the team can also check that the memory usage in the application runs most efficiently with proper memory management.

There are many testing procedures to choose from that have a unique testing goal. For a calculator application, there are not too many issues that need to be addressed before commercial release. The testing procedures that should be used are usability, compatibility, dynamic, performance and security. These procedures have their own functions and purposes that are:

* **Usability**
  + Usability testing will help us know if there is any need to introduce a change or modification to the application in order to make it more user friendly.

(<http://www.exforsys.com/tutorials/testing-types/usability-testing.html>)

* **Compatibility**
  + The software product that is more compatible with different computing environments at the end user level will be the most successful one, both commercially and structurally.

(<http://www.exforsys.com/tutorials/testing-types/compatibility-testing.html>)

* **Dynamic**
  + Dynamic testing will ensure that the software will be stable and functioning during the installation and right through its life cycle soon after its installation at the end users level.

(<http://www.exforsys.com/tutorials/testing-types/dynamic-testing.html>)

* **Performance**
  + Performance testing determines the speed and effectiveness of a software application and can be used as a diagnostic tool to detect bottlenecks in communication technology.
  + Performance requirements should come in a concrete, verifiable manner so that you can confirm the efficiency of the software application.

(<http://www.exforsys.com/tutorials/testing-types/performance-testing.html>)

* **Security**
  + Security testing is an important and integral part of the software development process that should be conducted before introducing the application into the commercial domain.
  + Securing applications can ensure system safety and security and can impede attacks by hackers.

(<http://www.exforsys.com/tutorials/testing-types/security-testing.html>)

In conclusion for a more advanced application there would be a necessity for more testing procedures. But for an application like this calculator these basic testing procedures are all that’s needed to make this software ready for commercial release.

# **Recommendations**

On upgrades and future enhancements:

It is recommended that regular updates to functionality and look & feel be scheduled at regular times, for example, every 3 months. This should include fixes for problems reported by users, and each release should include a Release Notes document so the users can see what is new. Other areas of enhancement might include performance improvements, or refactoring code to make the application smaller and more efficient. New features might be introduced over time, either at the request of users, or if competitors introduce new features that reduce the appeal of this app.

The app should check when started by the user, to see if an update is available, based on version number. If so, it should prompt the user to download the new version and install it. This installation will remove the old version and install the new one.