**Deliverable 3**

CRC Model

Calculator is main class with functions for:

* Choose Function (User chooses initial function to use)
* Parse Additional Function (After choosing first and entering values, can choose to parse additional function or get answer)
* Display History (Displays calculator history of the current user since they started using calculator)
* Addition (Simple addition function)
* Subtraction (Simple subtraction function)
* Multiplication (Simple multiplication function)
* Division (Simple division function)
* History (History is an array of Functions, which are separate classes for special ones and strings for simple arithmetic)

Exponential Function is a class

* Calculate Answer (calculates value of exponential function after user enter values for a, b, and x – currently x can only be an integer)
* Get Superscript (takes in string and outputs it as an exponent)
* Set Multiplier Number
* Set Base Number
* Set Exponent
* Multiplier Number (the value of a for the function)
* Base Number (the value of b for the function)
* Exponent (the value of x for the function)
* Answer (the final answer of the function after values entered)

OR

Function class

* exponential\_function (calculates exponential function from input)
* arccos (calculates arrcos function)
* sinh (calculates sinh function)
* log (calculates log function)
* mad (calculates MAD function)
* standard\_deviation (calculates standard deviation)
* special\_exponential\_function (calculates special exponential function)

Pseudo Code for Exponential Function

‘\*\*’ Denotes the exponential arithmetic in the Python language.

function exponential\_function (mult\_num, base\_num, exp\_num, answer)

answer = (mult\_num) \* (base\_num \*\* exp\_num)

return answer

function set\_mult\_num(String mNum)

if ‘/’ in mNum

try

num = contents of mNum before ‘/’

den = contents of mNum after ‘/’

mult\_num = float(num)/float(den)

return mult\_num

catch Division By Zero Exception

return None

else

mult\_num = float(mNum)

Task

Compute the sum of the square root of 2, the cube root of 3 and the fourth root of 4 using the Exponential Function (√2 + 3√3 + 4√4). Hint: use fractions in the exponent.

Persona

Marco is a 2nd year Concordia Student in the Pure and Applied Math Undergraduate program. As a student, he uses a calculator almost every day, mostly for trigonometric functions or simple arithmetic. He prefers the practical aspect of math, specifically algebra and statistics, and enjoys solving problems. While not experienced with computers, he has used an online scientific calculator before and has some experience with command line. A hard worker who values simplicity and ease of use, he hopes to become a TA for an algebra class in the future.

Use Case

This use case is a simple use of our Eternity System. The user is a student, and the purpose of their use is to retrieve an answer to a problem. The Choose Function use case is a generalized use case, which branches into Choose Special Functions or Choose Simple Arithmetic. The initial use case also has the preconditions that the student has analyzed the problem and identified the desired function and needed values. The user makes a choice, and is then asked to Enter Values, for their chosen functions. Once values have been entered, the system gives the student the chance to Parse Secondary Function. If they want to, then Parse Secondary Function becomes an extension of Choose Function as the student must now choose a second function to parse. Once the secondary function is parsed, or if none was chosen, then student moves on to Perform Calculation, which is a generalization of the Receive Final Answer use case and displays the answer to their function. The student can then view the calculator history, as at least one function must be performed for the history to not be empty.

Macro Architecture Design

The **macro architecture design** for the Eternity System can be explained using a Layered View. The architecture begins with User Input, which is how the user can interact with the system. The user is asked to input through the command line, the function they wish to work with, the necessary values, and whether they wish to parse a secondary function as well. The user can continue to input values, until they finally decide to stop, by requesting no further function be parsed. This moves the system to the second layer, which is the Computation Layer.

In the Computation layer, the input from the user for any single function is used immediately. Once a function is chosen and the appropriate values are inputted, the Computation layer performs the necessary calculations. The computation is divided between the Calculator class, which is responsible for calling the appropriate method from the function classes and performing the simple arithmetic operations of addition, subtraction, multiplication, and division. While the calculations for the special functions are handled in the function classes, one for each special function. The Calculator class determines which function is used by the user input, generates an object of the specified functions class, and calls the appropriate methods to perform the computation. Once the result is determined, the Calculator class stores the result, and wait for the next user input, whether that be to display the answer or parse a secondary function. If the choice is the latter, the result is kept, and eventually added/subtracted/multiplied/divided to the new function call result. If the choice is the former, the Calculator class can display the final answer to the user through the command line.

Micro Architecture Design

The **micro architecture design** for the Eternity System is done using **Object-Oriented Design**. The work is divided among separate **classes**. One Calculator class, which acts as the main class and how the user interacts with the Calculator, and the Function classes, a class for each of the special functions that includes any subordinate functions or variables they may need. This does result in high **coupling**, since the Calculator class collaborates with many others, but it offers better independency and allows the individual classes to define their own **instance variables** and subordinate functions. Each of the function classes have the responsibility of performing the computation, while the Calculator class is responsible for user input and parsing of functions. The Calculator also provides the work for simple arithmetic, both for individual computation and for parsing of functions. The Calculator has only one instance variable which is the History, represented as an array, which stores the information of all calculations done from the latest startup of the system. The History can also be displayed through the Calculator class.

The ExponentialFunction class has several functions, such as the set methods to set the values for the member variables: multiplier, base, and exponent. The ExponentialFunction class also has an answer member variable, which is used to store the answer of the computation, after the calculate\_answer function is called. The set functions are also used to handle any division by zero errors, since the user can enter fractions as input. An example of which is given in the pseudocode for the set\_mult\_num function. If the user enters a fraction with 0 as the denominator, the set functions return None, so the function is never evaluated. The get super function is just used to display the exponent as a subscript to the screen, instead of using the ^ sign. This function was retrieved from GeeksforGeeks and makes use of python string functions. The calculate\_answer function implements the pseudocode algorithm for exponential function. Since any division by zero errors were caught in the set functions, the algorithm only calculates the answer using the values inputted. And finally, the \_\_str\_\_ method allows the information from the function to be displayed as a string.

The **micro architecture design** for the Eternity System is done using **Object-Oriented Design**. The work is divided among separate **classes**. One Calculator class, which acts as the main class and how the user interacts with the Calculator, and the Function class, a class that contains the method for each of the special functions, including any subordinate functions or variables they may need. This helps reduce the coupling between classes, since the Calculator class collaborates with only one other class. While providing better independency since all the implementations and methods for the functions are separate from the Calculator class. The function classes have the responsibility of performing the calculations for the special functions and returning the result, while the Calculator class is responsible for receiving user input, calling the appropriate method in the Function class. The Calculator also provides the work for simple arithmetic, both for individual computation and for parsing of functions. The Calculator has only 1 data member which is the History, represented as an array of strings, which stores the information of all calculations done from the latest startup of the system. The History can also be displayed through the Calculator class.

The Function class includes several methods, as it is responsible for the computation of the special functions. It includes the exponential\_function method which takes as input a multiplier number, base\_number and exponent, performs the exponential operation and then returns the result. The method makes use of the pseudocode described above, using the exponential arithmetic operator provided by Python, to calculate the final answer and return it. The input is received from the Calculator class, which also handles catching any exceptions, if the user inputs a fraction with a denominator of zero, like the set\_mult\_num pseudocode defined above.

Algorithm

The algorithm made use of the exponential arithmetic operator for python. The \*\* does not work as efficiently as the math library power function but returns the result with a very small percent difference. The arithmetic operator was chosen over designing a new exponential function both for simplicity, as the code is easier to understand and smaller. It was also chosen for decimal and fractional exponents, as when using decimal exponents, the Exponential Function becomes more complex, leaving more room for uncertainty and difference in the final answer. Originally, the root function was going to be used to change the base value and exponent values for decimal and fractional exponents, since #1/a = a√#, however, using the built-in arithmetic operator was determined to be simpler and more efficient. It minimizes the number of functions used, as well as decreases the running time of the function.

Test Result

Calculate the cube root of 3 using the Exponential Function.

A screenshot of a computer

Description automatically generatedText

Description automatically generatedUsing percent difference, we will see the error percentage from the cube root value calculated using the Eternity System. For simplicity, we will use up to 11 decimal points each, for both values. In this instance V1 is the exact answer for cube root of 3, while V2 is the answer from Eternity.

Percent Difference = [|V1 – V2|/([V1+V2]/2)] x 100

V1 = 1.44224957031

V2 = 1.44224957031

Percent Difference = 0.0 %

This margin of error is 0% when dealing with up to 11 decimal points.

Glossary (Ext.)

Class: an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods). [1]

Coupling: the degree of interdependence between software modules (class or package or microservice). [2]

Instance: specific object created from a particular class. [3]

Instance Variable: tied to the instance of the class; contents of instance variable are independent from one object instance to another. [4]

Layered Pattern/View: used to separate higher-level from lower-level responsibilities. [5]

Method: a specification that defines a procedure or behavior of an object, also referred to as functions of a class. [1]

Macro-Architecture Design: the aspect of software design that describes how software is decomposed and organized into components, and the interfaces between those components. Independent of any paradigm. [5]

Micro-Architecture Design: the aspect of software design that describes the components at a level of detail that enable their construction. Usually dependent on a paradigm, such as object-oriented or procedure oriented. [5]

Object-Oriented Design: process for defining classes that constitute a software system and the relationships among them, defining the structure and behavior of instances of those classes. [5]

Works Cited

[1] D. Braunschweig, “Objects and Classes,” Dec. 2018, Accessed: Jul. 18, 2021. [Online]. Available: https://press.rebus.community/programmingfundamentals/chapter/objects-and-classes/

[2] G. Pagade, “Difference Between Cohesion and Coupling | Baeldung on Computer Science,” May 25, 2021. https://www.baeldung.com/cs/cohesion-vs-coupling (accessed Jul. 25, 2021).

[3] “Classes (OOP) | Brilliant Math & Science Wiki.” https://brilliant.org/wiki/classes-oop/ (accessed Jul. 18, 2021).

[4] R. Tayal, “Class vs Instance Variables,” *Medium*, Jun. 11, 2019. https://medium.com/python-features/class-vs-instance-variables-8d452e9abcbd (accessed Jul. 25, 2021).

[5] P. Kamthan, “Introduction to Software Architecture.” Pankaj Kamthan.