Group 12

Calculator
Software Architecture Document

Version 1.3

Group 12	Version: 1.3
Software Architecture Document	Date: 21/Oct/24
Software Architecture	

Revision History

Date	Version	Description	Author
21/Oct/24	1.0	Created document	Maxfield
28/Oct/24	1.1	Formatted document	Maxfield
9/Nov/24	1.2	Finished section 1	Caleb
10/Nov/24	1.3	Finalized revisions	Maxfield

Group 12	Version: 1.3
Software Architecture Document	Date: 21/Oct/24
Software Architecture	

Table of Contents

Group 12	Version: 1.3
Software Architecture Document	Date: 21/Oct/24
Software Architecture	

Software Architecture Document

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed architectural overview of the system by using lots of different views to depict different aspects of the system. This document is used to understand the significant architectural decisions which have been made on the system.

This document's role is to define the architecture of the software project. This document will contain definitions of our architecture, our goals and constraints for the architecture, and its subparts.

1.2 Scope

This document's scope is to define high level architecture, not low level architecture, as well as defining component relationships.

1.3 Definitions, Acronyms, and Abbreviations

Glossary is external

1.4 References

Glossary is external

1.5 Overview

The rest of this document will contain information about the architecture representation, architecture goals and constraints, the logical view of the significant parts of the design model, descriptions of the user interface, and how the architecture will relate to the rest of the system.

2. Architectural Representation

The architecture for this system will be object oriented; we are using C++ so we decided early into the development cycle to implement classes. This will allow for high amounts of modularity, and allow us to easily add new features in the future. Every class we implement will execute a specific function of the calculator. We will layer our software architecture. The top layer will be user interface; this will be where the user inputs equations. The next layer will be equation processing and converting the equation into a workable form to call normal equation functions. Finally, our bottom layer will process segments of the inputted equation and return values. We will connect the different layers of our architecture using function calls. We believe this architecture style will ease development and allow for high amounts of modularity.

3. Architectural Goals and Constraints

The overall architectural goal is to make a working calculator program. As outlined by the specifications for this project, there will not be a need for off-the-shelf products, and besides the server or computer executing the program, there are no other physical requirements. There is not a significant security concern as the data a user inputs should not be confidential, and the data is not being saved. Our implementation strategy will be incremental since the requirements have already been outlined. After developing and testing the program, we will transition to streamlining the user interface. The design is going to be a simple prompt that asks the user for a problem then in a different section it answers ideally for the user to change the problem as needed and the answer will change with it. In future iterations, we may choose to expand features to include trigonometry but those types of functions overstep the bounds of this project. Finally, the team will be broken up into a few groups that will each focus on their task. Specific roles have already been outlined in prior design documents. Our schedule will

4. Logical View

4.1 Overview

The program is oriented by a main class that gathers information from the user and then feeds it into the calculator class. Within this calculator class, there will be several functions, all with a specific goal. It will range from functions whose sole goal is to interpret what math operation needs to take place. Then a function that dividends the equation so P.E.M.D.A.S. can be achieved.

4.2 Architecturally Significant Design Modules or Packages

4.2.1 Main Class

The main class will ask the user for an equation then it will take that given information a apply it to the Calculator class. As well as take the final solution and give it back to the user

4.2.2 Calculator Class

The calculator class will serve as the leader of the program. While the main class works to start the program this class will be used to take that information from the main class and distribute it into a few functions split into two main groups that will be used to solve the equation. one being the P.E.M.D.A.S. function that splits the problem so the class follows the basic rules. The second function is the order of operations functions. This are many functions that will do the basic operations such as add, subtract, multiply, divide, etc.

4.2.3 PEMDAS Function

PEMDAS is the basic math law that must be followed for the correct solution. First, the equation should look at the parentheses to evaluate what is within those, followed by the exponents, then multiplication and division which can be interchangeable, and finally, addition and subtraction, and those two are interchangeable as well. Overall the PEMDAS Function will call the functions of operations and make sure they follow the rules of PEMDAS.

4.2.4 Operation Functions

The Operation functions are the basic math functions of add, subtract, multiply, divide, etc. These functions are simple and will take the two numbers given from the PEMDAS function complete the basic operation return it to PEMDAS and send it to the next function to continue to complete the problem.

5. Interface Description

The program prompts the user for an input on the screen, asking for a math problem such as 5 + 4. The user then types their input onto their keyboard which gets displayed on the screen. Valid inputs are basic arithmetic operators such as +, -, *, /, ect, as well as “(“ and “)”. Invalid inputs are anything that's excluded from that such as letters and improper parentheses. The program will compute the answer and display it on the screen

6. Quality

The program is extremely simple to use, implement, and move. It is a simple C++ that will work on any device that can compile C++. As it is a simple file it can be replicated and distributed anywhere. And lastly, it is very easy to use as once you start the program it will prompt you for a math problem, and once you input a valid one it will give you the answer to said problem.