Apex Software

Tech Calculator
Software Architecture Document

Version 1.3

Revision History

Date	Version	Description	Author
10/31/2024	1.0	Initial Drafting of Software Architecture Document	Gage Weaver
11/1/2024	1.1	Further Drafting of Software Architecture Document	Gage Weaver
11/7/2024	1.2	Revisions to sections 4.1 and 4.2	Gage Weaver
11/8/2024	1.3	Add diagrams and updates to 4.2	Gage Weaver

Table of Contents

Software Architecture Document

1. Introduction

The software architecture document will aim to provide a comprehensive look at the tech calculator project. Tech Calculator will be a basic calculator app with a simple user interface that will perform PEMDAS Operations along with Modulus Operator functionality which is commonly seen in programming languages. It will be able to parse user input with parentheses in order to accurately evaluate expressions. Section two will contain the architectural representation of the project, outlining the views and major elements of the software. The constraints of the project will be outlined in 3 along with goals for the layout of the software. Section four will be describing different modular parts of the project and how they fit together. Section five will provide a basic description of the interface, while six will outline the capabilities of the project, such as its reliability.

1.1 Purpose

The purpose of this document is to lay the fundamental outline for both the intricacies of the separate programs themselves, and also how each program will be required to interact with one another.

1.2 Scope

This document will refer to both the backend, frontend, and makefile pieces of programming that need to be done to produce a functional calculator. This document will also specify guidelines and goals for this code.

1.3 Definitions, Acronyms, and Abbreviations

All terms that are not clear will be defined in their respective areas of the document when referenced. Acronyms used will be PEMDAS, this acronym is for the basic order of operations in arithmetic, meaning parentheses are done first, then exponents, then multiplication and division (left to right if multiple appear), then addition and subtraction. The modulus operator is a common operator used in computer operations, when applied it returns the remainder from division of two numbers, for example 3%4 would return 4. The modulus operator is represented by the % sign and has the same precedence as multiplication and division. Ui stands for user interface.

1.4 References

A reference is made to the Software Requirements Specification document in section 3, and the specific spot where it is referenced is marked with an asterisk (*).

1.5 Overview

The rest of the Software Architecture Document contains as follows. Section two will be outlining the architecture for the project, and the included views. Section three will outline any constraints along with the ideal layout of the architecture. Sections 4 and 5 will be going over the model for the architectural layout of the system and the layout for the interface. Then lastly section 6 will contain the non functional capabilities of the system

2. Architectural Representation

Layers: Tech Calculator will have two main layers. The first will be the User Interface layer, which will be a basic command line style interface that allows the user to input their expressions, this layer will make a call to the backend (logic) layer and output the results along with any possible errors to the user. The second will be our backend (logic) layer, this will contain all the functions necessary to carry out the mathematical operations along with the logic for our expression parsing and any error codes.

Data Flow: The user inputs into the ui layer, function calls are then made to the backend layer, and then the result or an error are returned back to the ui layer for the user to see.

3. Architectural Goals and Constraints

Tech calculator has a minimal set on its software requirements and its objectives. Physical safety is not a concern with Tech Calculator, however checks will need to be put into place to avoid machine damaging

errors such as stack overflow errors, and any other potential errors that could affect the users pc. Privacy is not an issue as these are just mathematical expressions, and no user data will be collected. Tech Calculator is portable to any machine that can compile C++ applications, and it can be distributed via download link. The constraints with development are derived from the Software Requirements Specification* document. The application will need to be developed in C++ and must be able to run on most modern computers. Lastly the strategy for the design of the software is to have a minimal, text based expression evaluator without a complicated user-interface. As overcomplication can lead to unnecessary errors.

4. Logical View

Packages: Tech Calculator will have two packages that will be bundled together into one calculator app for release. These will be the core functionality package, and the user interface package. These are described further in section 4.1 and 4.2

Classes: Tech Calculator will have a user interface class running from the ui package that will contain methods within it for interacting with the Core Functionality Package and Command Handling class. Within the core functionality package there will be interaction between the operator, expression parsing, and error handling classes.

4.1 Overview

The package hierarchy is modeled with the UI package being the top package, with the Core Functionality below it. This is due to our organizational structure of having a UI method always running that will make calls to functions and classes within our core functionality package.

UI package: This package contains one layer containing all of the necessary user-interface code

Core Functionality Package:

Expression Parsing Layer: This layer contains the code for breaking down the user input and handling any violations with expression notation such as too many parentheses or bad operator usage and will make calls to the operator layer

Operator Layer: This layer contains the functions to be called by the expression parsing layer

4.2 Architecturally Significant Design Modules or Packages

Core Functionality Package :

4.2.1: Operator Layer:

Operator Class contains the following methods:

- Addition(float a, float b): Adds two numbers. (a+b) and returns the answer
- Subtraction(float a, float b): Subtracts the second number from the first. (a-b) and returns the answer
- Multiplication(float a, float b): Multiplies two numbers. (a*b) and returns the answer
- Modulus(float a, float b): Returns the remainder of dividing the first number by the second number and returns the answer
- Divide(float a, float b): Returns the result of division, first number divided by second (a/b) and returns the answer with safeguards for division by 0
- Power(float a, float b): Computes the first integer to the power of the second and returns the answer

4.2.2: Expression Parsing Layer:

Expression Parsing class contains the following methods:

- parseExpression(string expression): Takes the expression as a string from the user and will break it down into numbers and operators
- validateExpression(string expression): This method will ensure proper usage of parentheses and operators

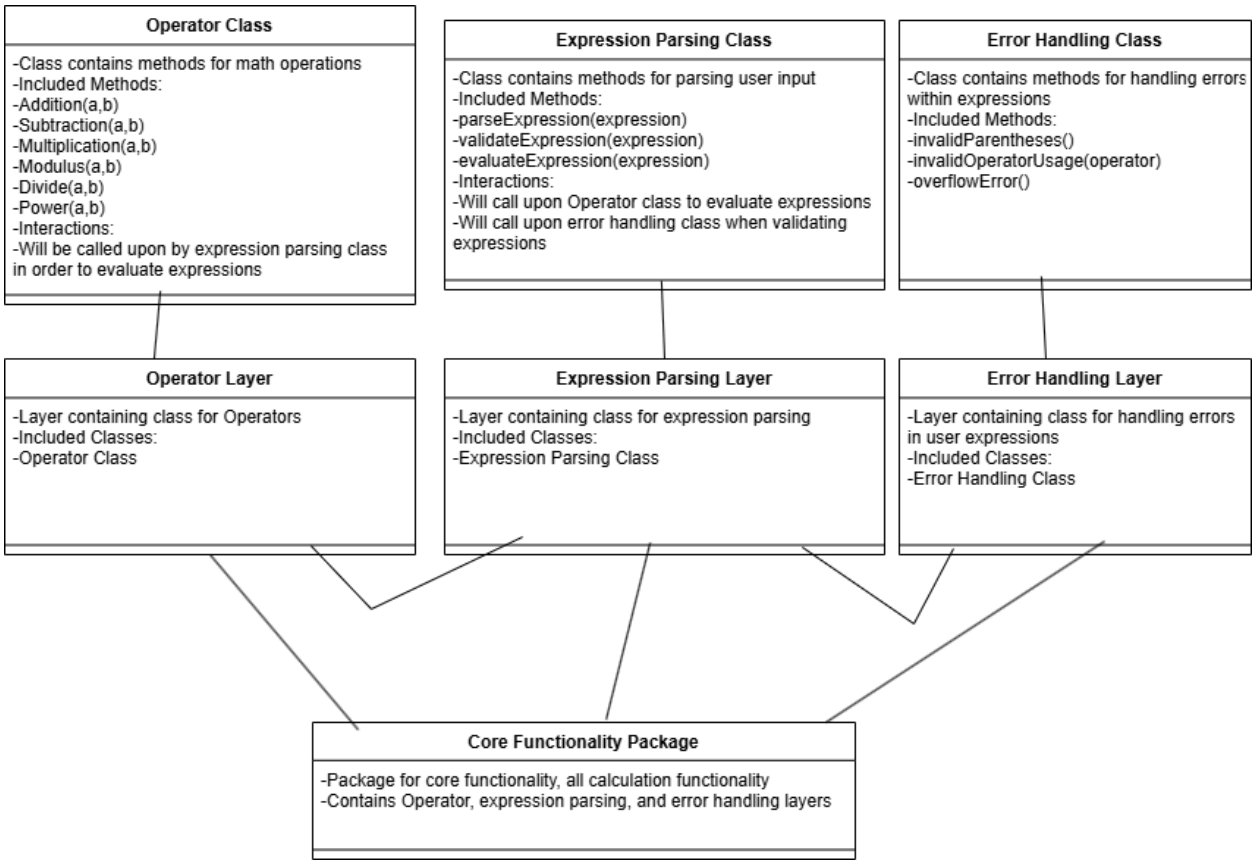
- evaluateExpression(string expression): This will be called upon by parseExpression after splitting input and ensuring its validity to make calls to our operators following PEMDAS

4.2.3 Error Handling Layer:

Error Handling class contains the following methods:

- invalidParentheses(): Checks for mismatched or otherwise invalid parentheses
- invalidOperatorUsage(operator):Checks for correct operator usage
- overflowError(): Checks that input/output would be in a safe calculation range with respect to magnitude

Core Functionality Package Diagram:



User Interface Package:

4.2.4 UI Interaction Layer:

UI Interaction class contains the following methods:

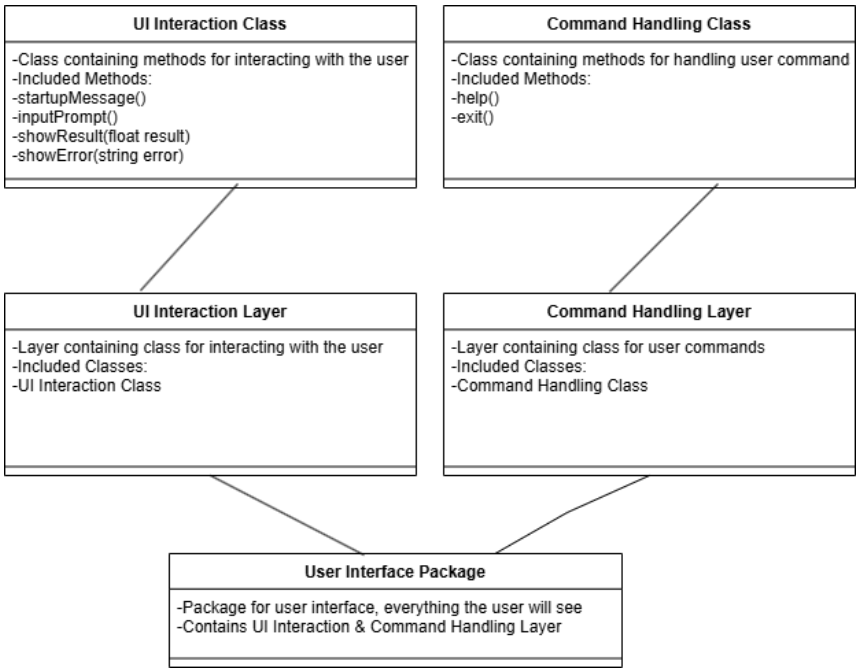
- startupMessage(): Shows the user a message upon startup informing them about the calculator and brief instructions on how to use it
- inputPrompt(): Prompt the user to enter expression or type help to see user manual/instructions
- showResult(float result): Show the user the result of their calculation from core functionality package
- showError(string error): Show the user the error with their expression from core functionality package

4.2.5 Command Handling Layer:

Command Handling class contains the following methods:

- help(): Shows the user a menu with brief notes about using the calculator with the option to see the user manual
- exit(): Exits the calculator

UI Package Diagram:



5. Interface Description

Users' expressions will be input into a text-based interface in a single line. Valid inputs include multiple numerical values as well as arithmetic operator symbols such as: +, -, *, /, %.

In order to get numerical inputs as well as the desired arithmetic operator, the user will be prompted with messages instructing them on what input they need to enter at a given time, and provide them the ability to input their values using the keyboard on their device.

The output to the user will be the numerical result of the chosen arithmetic operation on the numerical values they entered or a relevant error message. The output will display the parsed values in the terminal, or alternatively, any applicable instructions or error messages based on syntactical errors. E.g., the user inputs an expression with a divisor of zero, the terminal outputs an explanation of valid input and examples of invalid expressions. Additionally, the user can request a full list of instructions and tips by typing 'help'.

6. Quality

Reliability: Our Tech Calculator will be programmed in such a way that promotes reliability. User input will be restricted to only valid inputs, limits will be placed on how large inputs/outputs can be, and so on. Extensive debugging will take place during the development process to ensure a fully operational product. Consistent output is also expected, as equivalent expressions should deliver the same result every time they are evaluated.

Simplicity: The Tech Calculator will be simple enough such that the controls are intuitive to any potential user. A user encountering errors will be prompted with either the user manual, or specific error messages

that provide guidance. Alongside this, starting the program will ideally inform the user towards where to find the user manual (e.g. “Use the ‘help’ command to receive guidance.).