Scrum Blasters

Big Ole' Calculator
Software Architecture Document

Version <1.01>

Revision History

Date	Version	Description	Author
11/6/2024	1.0.1	Proposed Completed Software Architecture Document	Everyone

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

The purpose of the SAD is to lay out the plans for the implementation of the BOC. The SAD will look at how we plan to build the BOC architecturally, logically, and graphically.

1.2 Scope

The SAD will apply to the BOC. This document will set out how we plan to build the BOC and what methods we will use to implement it.

1.3 Definitions, Acronyms, and Abbreviations

- SAD: Software Architecture Design
- BOC: Big Ole’ Calculator
- PEMDAS: Parenthesis, Exponent, Multiplication, Division, Addition, Subtraction

1.4 References

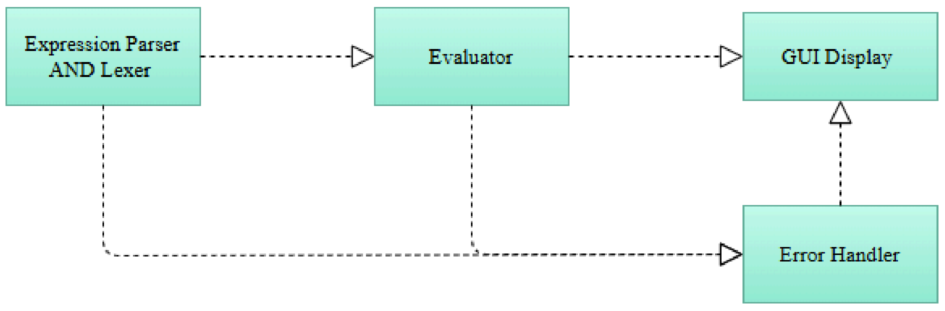
There are no references in the Software Architecture Design document.

1.5 Overview

The SAD will contain an overview of how our team plans to build and implement the BOC into a larger system.

2. Architectural Representation

The calculator will have a pipe and filter architecture. There will be a linear flow of independent components. It will always start with an expression parser, followed by an arithmetic module, then finally ending with an GUI display.



3. Architectural Goals and Constraints

The goals and objectives of this architecture will be to safely and sustainably operate our software. It will be important to manage memory in a way that won’t risk any segmentation in user hardware and it will also be important that the program will be sustainable enough for prolonged usage. With those goals in mind, it will be important to create a modal and distributable resource for foreign hardware to properly adapt. Security or privacy should not be a concern (no sensitive information could be provided). In addition to these goals, the project will keep in mind design goals, implementation strategies, and architectural component size to give all team members a balanced workload.

4. Logical View

The lexer will work with the expression parser in getting the user's input into workable data for the rest of the program. The evaluator will perform the arithmetic operations and follow operation priorities. Output and GUI display will take any errors produced by the error handler or take the final number from the arithmetic module and display it to the end user. The error handler will work with each component to detect any invalid data.

4.1 Architecturally Significant Design Modules or Packages

<u>Expression Parser</u>

- Lexer: Determines the input to be numbers, operators, or parentheses.
- Parentheses Handler: Checks to see if there are the correct amount of parentheses in the correct order.
- Error Handler: Checks for invalid input sequences (i.e. garbage input).

<u>Arithmetic Module</u>

- Operation Priority: This part should perform PEMDAS math operations in the correct sequence.
- Math Evaluation: This part performs the given operation and returns value.

<u>GUI Display</u>

- Output Handling: Formats and provides visual answer to given input with necessary details (e.g. sign). Might also output an error in execution.

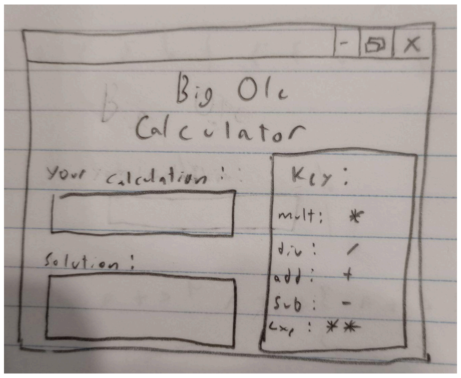
5. Interface Description

Description: Our calculator will have a box for input, a box for output, and a key explaining what a valid input looks like.

Valid Inputs: +,-,*,/,%,**, as well as numeric constants. The input must be in a valid mathematical format, aka: Complete parenthesis, no hanging operators, etc.

Output: Either the solution to the equation, or an error message communicating what went wrong.

Screen Format:



6. Quality

Because we use a pipe and filter system our code will be more extensible, as in it will be easier to add functionality with minimal impact on existing components.