Arithmetic Expression Evaluator

Version <1.0>

[Note: The following template is provided for use with the Unified Process for EDUcation. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.] *Marked (shaded) areas: items that are OK to leave out.*

Revision History

Date	Version	Description	Author
<11/09/2023>	<1.0>		<Michael Merino>

Table of Contents

1. Introduction

1.1 Purpose

Covers the most important components (or modules or classes). Software architecture plays a crucial role in software development by providing a high-level structural framework for a system, allowing for efficient organization, scalability, and maintainability. It facilitates the abstraction and separation of concerns, fosters reusability, and ensures flexibility when choosing components and technologies. A well-designed architecture optimizes performance, addresses quality attributes, and aids in risk management.

1.2 Scope

What's affected by this document: Our teams choices on the oncoming software coices pertaining its architecture

1.3 Definitions, Acronyms, and Abbreviations

SDLC - Software Development Life Cycle

API - Application Programming Interface

SDK - Software Development Kit

GUI - Graphical User Interface

TDD - Test Driven Development

DB - Database

Derivable - Tangible or intangible item that is delivered to a client or stakeholder as part of a project.

Sprint - A "sprint" is a set period during which specific work has to be completed and made ready for review. It's essentially a short, focused phase of development.

Bug: An error, flaw, or unintended result in software.

Deployment: The process of releasing a new feature, app, or update to the available infrastructure.

Push/Pull: Sending or receiving changes to/from a remote repository in a version control system.

Commit: A set of changes or edits saved to a version control system.

Branch: A separate line of development created in a version control system, which can later be merged back with the main line.

Framework: A platform for developing software applications. It provides a foundation on which software developers can build programs for a specific platform.

1.4 References

00-Project-Description.pdf

01-Project-Plan-marked.pdf

01-Project-Plan.dot

02-Software-Requirements-Spec.docx

03-Software-Architecture

1.5 Overview

The remaining portion of the Software Architecture document contains the bulk of the architectural choices made for the group:

- Architectural Representation

- Architectural Goals and Constraints

- Use-Case View

- Use-Case Realizations

- Logical View

- Overview

- Architecturally Significant Design Modules or Packages

- Interface Description

- Size and Performance

- Quality

2. Architectural Representation

We are going to use an object-oriented programming software architecture where we are representing input from the user.

3. Architectural Goals and Constraints

Constraint - Object Orientated, In C, We all work so the schedule is tight

Goal - Functional Code with the object for all unique expressions and callable functions

4. Use-Case View

[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements or if they stress or illustrate a specific, delicate point of the architecture.]

4.1 Use-Case Realizations

[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations and explains how the various design model elements contribute to their functionality. If a Use-Case Realization Document is available, refer to it in this section.]

5 Logical View

User Handler Class:

Handles user input and turns it into something usable

Input Verifying Class:

Verifying user input, making sure no errors

Parenthesis Handler Class:

The class that handles the inputs parenthesis, partitioning the individual math expressions in the correct order to the next class

Arithmetic Evaluator Class:

Takes in arithmetic expressions and does necessary math to find solution

5.1 Overview

Pemdas = Classes

Main - takes user input and checks if valid (no letters)

Using a stack filter through the equation turning each operation into a number removing one operation at a time till there is just one number left

Recursive function to check for pemdas in order, when no operators left return value

P(parentheses) - if any parentheses left return equation inside of parentheses by putting it through the recursive function

E(exponents) - if any exponents do math then return the value

M(multiplication) - if any multiplication do math then return the value

D(division) - if any exponents do math then return the value

A(addition) - if any exponents do math then return the value

S(subtraction) - if any exponents do math then return the value

5.2 Architecturally Significant Design Modules or Packages

Module: Calculator

Description: Main Object that houses sub objects for different functionality within the calculator

Sub-Module: Operators

Description: class that represents the operator and tells the evaluator how to use it to evaluate the expression

Sub-Module: Numbers

Description: class that represents numerical input.

Sub-Module: Stack

Description: holds operator and number classes put on by the user input class

Sub-Module: User Input

Description: takes input from a file and turns it into list of operators and numbers and sends them off to be verified

Sub-Module: Verifier

Description: verifies the list of operators and numbers to be correct and if correct puts the list on the stack.

Sub-Module: Evaluator

Description: takes off portions of the stack and evaluates them

Sub-Module: Output

Description: takes values from the evaluator and completes the operations and outputs the values

6. Interface Description

The user will input the arithmetic expression into input, and after clicking the button to start the process the solution will be displayed below in a separate box.

7. /Size and Performance

[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

8 Quality

Extensibility:

The architecture's modularity and clear interfaces enable seamless integration of new features, adapting to evolving requirements.

Reliability:

Robust fault-tolerance mechanisms and effective error handling ensure system resilience, crucial for mission-critical applications.

Portability:

Abstraction of hardware dependencies and use of platform-independent elements make the system adaptable to diverse environments.

Performance:

Efficient design choices and scalability planning optimize response times, throughput, and resource utilization.

Scalability:

The architecture supports scalable solutions, incorporating features like load balancing and distributed computing.

Security:

Access controls, encryption, and secure communication protocols safeguard against unauthorized access and protect sensitive data.

Maintainability:

Modularity, coding standards, and comprehensive documentation enhance system maintainability, reducing the total cost of ownership.

Safety:

Safety-critical systems include error-checking, fail-safes, and redundancy features to prevent and mitigate potential hazards.

Privacy:

Privacy measures like data encryption and strict access controls are integrated into the architecture to protect sensitive information and comply with regulations.