Arithmetic Expression Evaluator

Software Architecture Document

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 11/10/2023 | 1.0 | Update Introduction | Elizabeth Channel |
| 11/11/2023 | 1.1 | Update Architectural. Goals and Constaints | Aryan Kevat |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Architectural Representation 4

3. Architectural Goals and Constraints 4

4. Use-Case View 4

4.1 Use-Case Realizations 5

5. Logical View 5

5.1 Overview 5

5.2 Architecturally Significant Design Packages 5

6. Interface Description 5

7. Size and Performance 5

8. Quality 5

**Software Architecture Document**

# Introduction

The following document will detail the software architecture of the *Arithmetic Expression Evaluator (AEE)*. It will be referred to throughout the project.

## Purpose

The purpose of this document is to provide an overview of the software architecture. Using a series of subtopics such as the *Architectural Representation*, this document will act as a resource for the CAVJAC group and its programmers to refer to when regarding the architecture the *AEE*. It will detail specific architectural decisions made by the group.

## Scope

This document applies to the *AEE*. It will influence any future discussion and decisions regarding the architecture of the evaluator.

## Definitions, Acronyms, and Abbreviations

AEE – Arithmetic Expression Evaluator

## References

* Software Development Plan – 9/22/2023, CAVJAC Group
* Use Case Specifications Document – 10/13/2023, CAVJAC Group
* Software Requirements Document – 10/14/2023, CAVJAC Group

## Overview

The following document will detail the software architecture of the *Arithmetic Expression Evaluator (AEE)*. It will explain components such as goals and constraints of the project, a description of the interface, and performance of the software.

# Architectural Representation

[This section describes what software architecture is for the current system, and how it is represented. It enumerates the views that are necessary, and for each view, explains what types of model elements it contains.]

# Architectural Goals and Constraints

Referring to our Software Requirements Specification document, our requirements include being able to parse arbitrary user input, evaluate valid mathetical operations with correct operator precendence, and notify the user of errors in their input. The architecture must allow for the user to have a seamless experience within the program itself. The user should not need to rely on external programs or manual control flow in order to operate the program. Because the program is purely text based, the architecture will be portable and not rely on any platform specific user interface libraries.  
  
In addition to fulfilling our software requirements, there are additional constaints for our software architecture. It needs to follow object-oriented principles, including a class based design. The design should not be overly complex and should allow for completion before our (tentative) period for writing documentation, November 30th. Each team member should participate in the design and implementation of our program, including writing test cases. Finally, the project must both be open source and easily buildable using a copy of our repository hosted on GitHub.

# 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.]

## 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.]

# Logical View

[This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

## Overview

[This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.]

## Architecturally Significant Design Modules or Packages

[For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

For each significant class in the package, include its name, brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.]

# Interface Description

[A description of the major entity interfaces, including screen formats, valid inputs, and resulting outputs. If a User-Interface Prototype Document is available, refer to it in this section]

# Size and Performance

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

# Quality

[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.]