EECS 348 Group 14

Arithmetic Expression Evaluator in C++
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
11/9/23	1.0	First draft	Vincent Dick

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. Logical View **5**

4.1 Overview 5

4.2 Architecturally Significant Design Modules or Packages 5

5. Interface Description **7**

6. Quality **7**

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides an overview of the architectural structure of the system for the AEEC, using different views of each part to discern all of the different aspects in the system. It will capture each of the fundamental architectural decisions for the system and convey the intent of each part.

1.2 Scope

This Software Architecture Document provides an architectural overview for the AEEC. The AEEC will evaluate a mathematical expression provided by the user, as long as that expression is valid. The AEEC will evaluate the aforementioned expression accurately by following PEMDAS method of operations.

1.3 Definitions, Acronyms, and Abbreviations

- AEEC (Arithmetic Expression Evaluator in C++)
- PEMDAS (Parentheses, exponents, multiplication, division, addition, subtraction; method of evaluating mathematical expressions)
- UML (Unified Modeling Language)

1.4 References

- AEEC Software Requirements Specification

1.5 Overview

The rest of this document contains an overall description of the architectural overview of the AEEC software, providing the architectural representation and the goals and constraints of said architecture, the logical view of the design model, how users will interface with the AEEC, as well as the quality assurance of the software overall.

2. Architectural Representation

This document presents the object-oriented architecture of the AEEC system as an architecture, limited to the logical, development, and process view models. These views are modeled with the UML class and package diagrams.

3. Architectural Goals and Constraints

Safety: There are no potential risks or hazards that this software package can bring to the user(s).

Security and Privacy: There are no security or privacy concerns with this software package, as the user(s) is able to run the software within their own self-contained terminal/compiler. No data given by the user(s) is being transmitted to a database/server.

Off-the-Shelf Product: This software package is capable of being given to the user(s) for immediate use and required applications are not implemented.

Portability: The AEEC software package is capable of running on all operating systems (OS) that are capable of compiling and running C++ code.

Distribution: No data from the user is being distributed to databases/servers, the data is self contained within the terminal.

Reuse: This software package is designed to be a stand-alone tool that should not be integrated within other software, but can serve as a template for software packages.

Constraints: The software package is only limited to running on host machines/compilers that are capable of running C++ software.

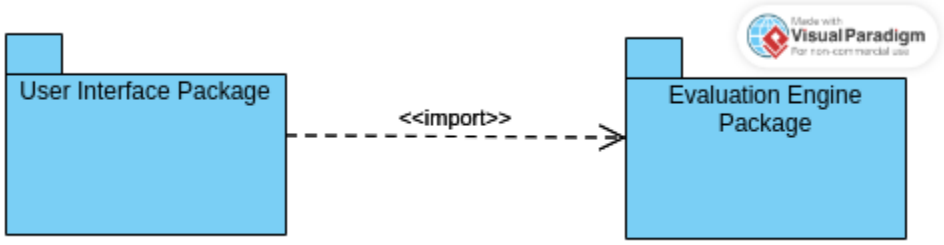
4. Logical View

4.1 Overview

This subsection describes the packages and classes within the AEEC through UML class and package diagrams with tables containing detailed specifications.

4.2 Architecturally Significant Design Modules or Packages

4.2.1 Package Diagram

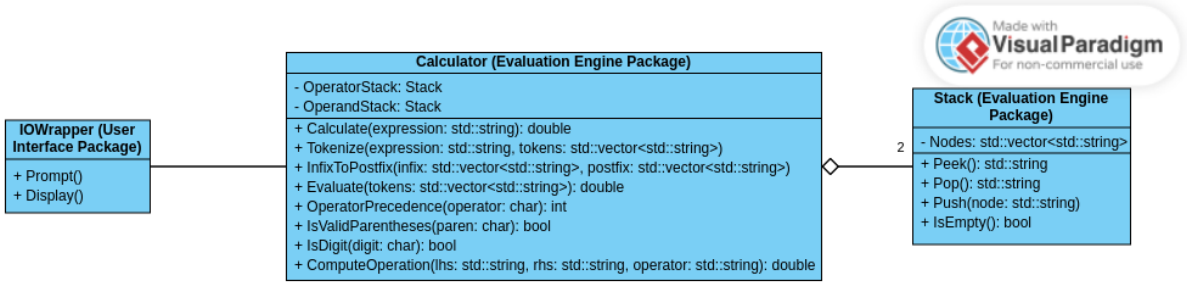


4.2.2 Package Descriptions

User Interface Package	
Description	The User Interface Package encapsulates the I/O functionality for the Arithmetic Expression Evaluator. This package is responsible for all communication with users.
Classes	IOWrapper
Relationships	Dependency of Evaluation Engine Package

Evaluation Engine Package	
Description	The Evaluation Engine Package encapsulates the calculation functionality for the Arithmetic Expression Evaluator. This package is responsible for taking an expression from the User Interface Package and returning a result.
Classes	Calculator Stack
Relationships	Dependent on User Interface Package

4.2.3 Class Diagram



4.2.4 Class Descriptions

Property	**Description**
Name	IOWrapper
Description	The user interface; how the user will interact with the calculator and view results
Responsibilities	Read user input and display an accurate evaluation of the user's equation
Relations	Association with Calculator
Methods	Prompt(): prompts the user for an equation to evaluate. Display(): display the evaluation of the equation submitted by the user.
Attributes	None

Property	**Description**
Name	Calculator
Description	The evaluation engine
Responsibilities	Perform an accurate calculation of an equation
Relations	Association with IOWrapper, aggregation of two stacks
Methods	double Calculate(std::string expression): Manage calculations at the highest-level, distributing the work to other functions within the Calculator class. void Tokenize(std::string expression, std::vector<std::string> tokens): Splits the expression into individual tokens (operators and operands. void InfixToPostfix(std::vector<std::string> infix, std::vector<std::string> postfix): Reorders tokens from infix (e.g., 5 + 3) to postfix (e.g., 5 3 +). double Evaluate(std::vector<std::string> tokens): Returns the evaluated expression, using the tokens in postfix order. int OperatorPrecedence(char operator): Returns the precedence of an operator based on PEMDAS.

	bool IsValidParentheses(paren: char): Returns whether or not the given character is a parenthesis. bool IsDigit(digit: char): Returns whether or not the given character is a digit. double ComputeOperation(lhs: std::string, rhs: std::string, operator: std::string): Returns the computation of an operation between two operands.
Attributes	Stack OperatorStack Stack OperandStack

Property	**Description**
Name	Stack
Description	Implementation of the "Stack" data structure to control storage of and access to expression tokens (operators and operands).
Responsibilities	Store and provide access to expression tokens in a first in last out (FILO) order.
Relations	Aggregate component of Calculator.
Methods	std::string Peek(void): Returns the top node without removing it from the Stack. std::string Pop(void): Returns the top node and removes it from the Stack. void Push(std::string token): Adds a node to the top of the Stack. bool IsEmpty(): Returns whether the Stack is empty.
Attributes	std::vector<std::string> Nodes

5. Interface Description

The main interface for the AEEC is the IOWrapper, which will prompt the user in the system's terminal for an equation to evaluate. If the user's input is a valid expression the resulting output will be a mathematical calculation of the user's expression, which will be displayed in the same terminal.

6. Quality

The AEEC architecture will follow the quality requirements that are described in the Software Requirement Specification.