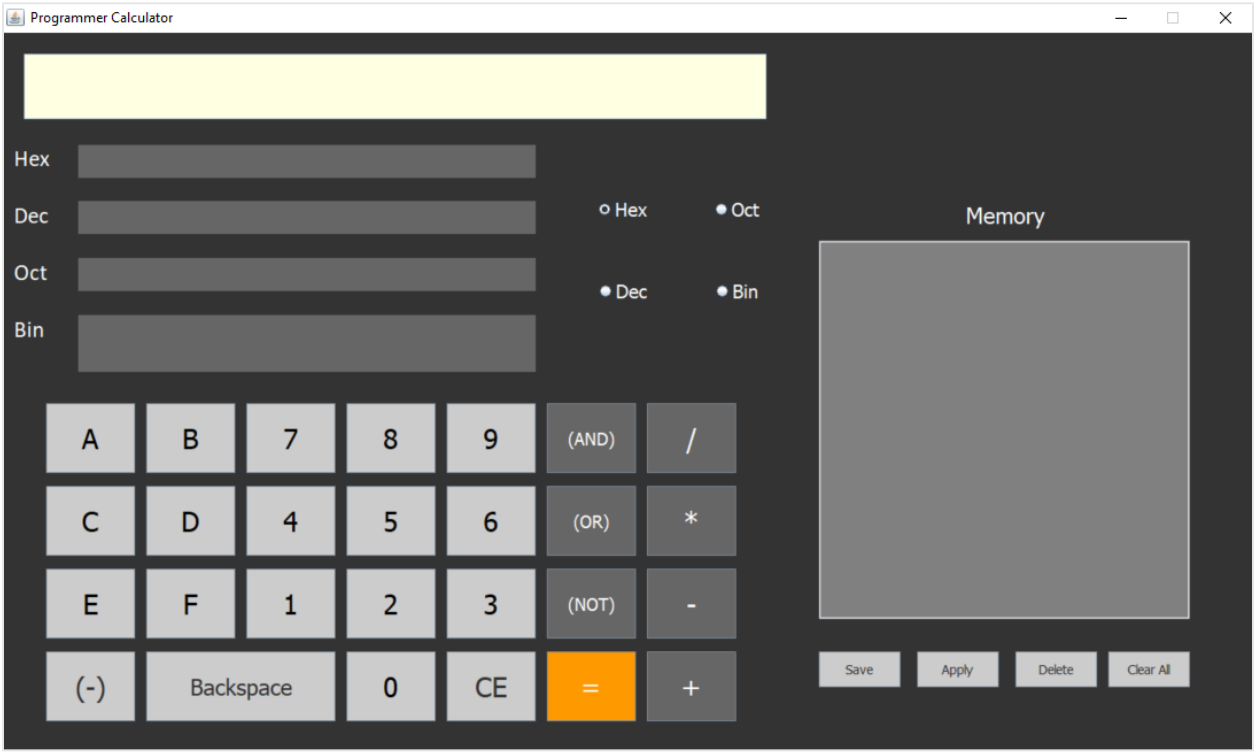
Software Requirement Specification

“Programmer Calculator”



Author(s):

Eric Olechovski

Table of Contents

Introduction

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements and functionality of the “Programmer Calculator”. This document should provide an overview of the program’s features, which includes the software application’s User Interface (UI) as well a description of the operations required by the program.

1.2 Scope

The “Programmer Calculator” is a program that displays conversions between the following numerical notations: Hexadecimal, Decimal, Octal Decimal, and Binary.

As well as performing arithmetic and logical operations between these numerical notations.

The intent of this program is to aid users in calculating various numerical notations. The program is also designed to help further the users understanding of the numerical notations along with their appropriate operations.

1.3 Overview

The Software Requirement Specification document will cover the functional and nonfunctional requirements for the “Programmer Calculator”. The requirements will entail how the program will operate. They will also discuss the visual aspects of the calculator as well as other features. This document also contains architectural designs for the “Programmer Calculator”, to give the reader a better understanding of the process and functionality of the calculator.

2.1 Nonfunctional Requirements

Usability	• User must be knowledgeable on what some numerical notations on the calculator are. • The programmer calculator has a similar apperarnce to a traditional calculator. • Does not require any type of login system.
Reliability	• Calculations **must** be precise and accurate! • No room for ambiguity. • Large inputs/outputs must be handled by either exceptions or using contraints to create limitations. • Storing previous entries must be reachable and accurate, until user ends the program.
Performance	• Calculations need to be rapid, Everything must be sub-second. • Non-webbased calulator, no need to support concurrent users • User should be able to continuously do calculations from the product of last operation.
Supportability	• Program requires a Java Virtual Machine (JVM) or some version of a Java Optimized Processor (JOP).
Implementation	• All users should be able to download the calculator program which will be a JAR file. • The user constraints are the supportability contraints listed above. • Users must be able to check for updates, if there is a newer version on the web.
Legal	• Software Engineering (SE) & Association for Computing Macinery (ACM) codes

2.2 Functional Requirements

Input	• User selects a numerical notation they desire as their input. • Numerical Notations (Hexadecimal, Decimal, Octaldecimal, and binary). • User creates a valid expression based on the numerical notation selected to convert.
Output	• Takes input expression and convert it into binary. Then perform the necessary operators if they exist in the expression. • Convert the binary into other numerical notations and display them accordingly. • Take the output from the calculation and store it in recent history.
Contraints	• Charcater limitation, to avoid computational overflow. • Disable unecessary buttons according to the selected numerical notation. • Validate the expression entered in by the user before performing calculations.

User Interface	• Consist of Buttons and a frame to display users input/output.
	• Does not require keyboard input entry.
	• Buttons:
	  ○ A – F
	  ○ 0 – 9
	  ○ Arithmetic Operators (+)(-)(*)(/)
	  ○ Logical Operators (AND)(OR)(NOT)
	  ○ Numerical Notation Selection (HEX)(DEC)(OCT)(BIN)

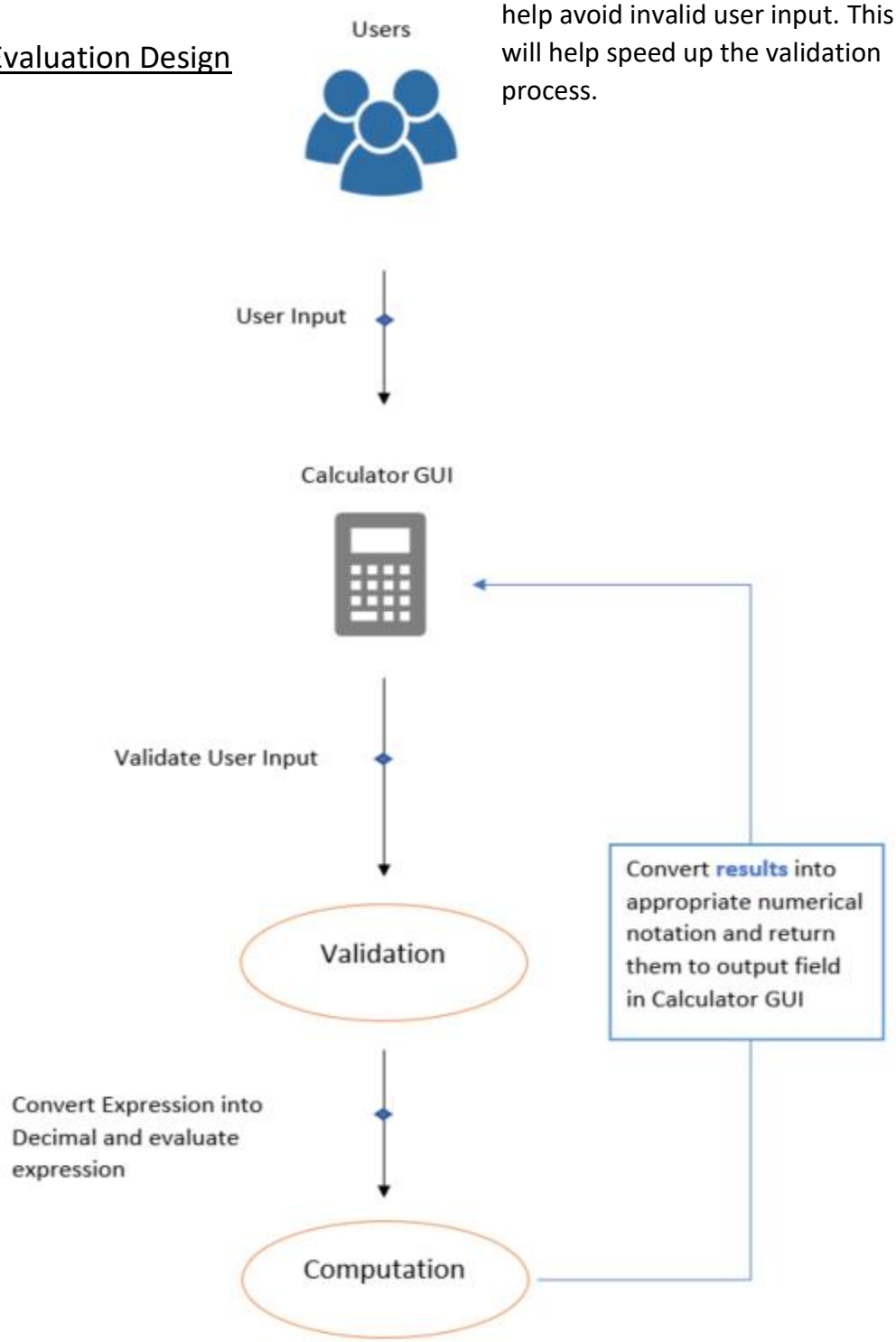
Visual Appeal/Usability	• Obtain basic appearance of traditional calculator so it is easy to operate. • Large buttons and fonts, easy to find your way around the calculator. • Window will be fixed to a specified size
Scalability	• Will not display decimals to avoid irrational numbers, (round down). • Will create a limit/contraint to avoid numbers too large or too small for optimal computation, and to fit in the output window.

Architecture

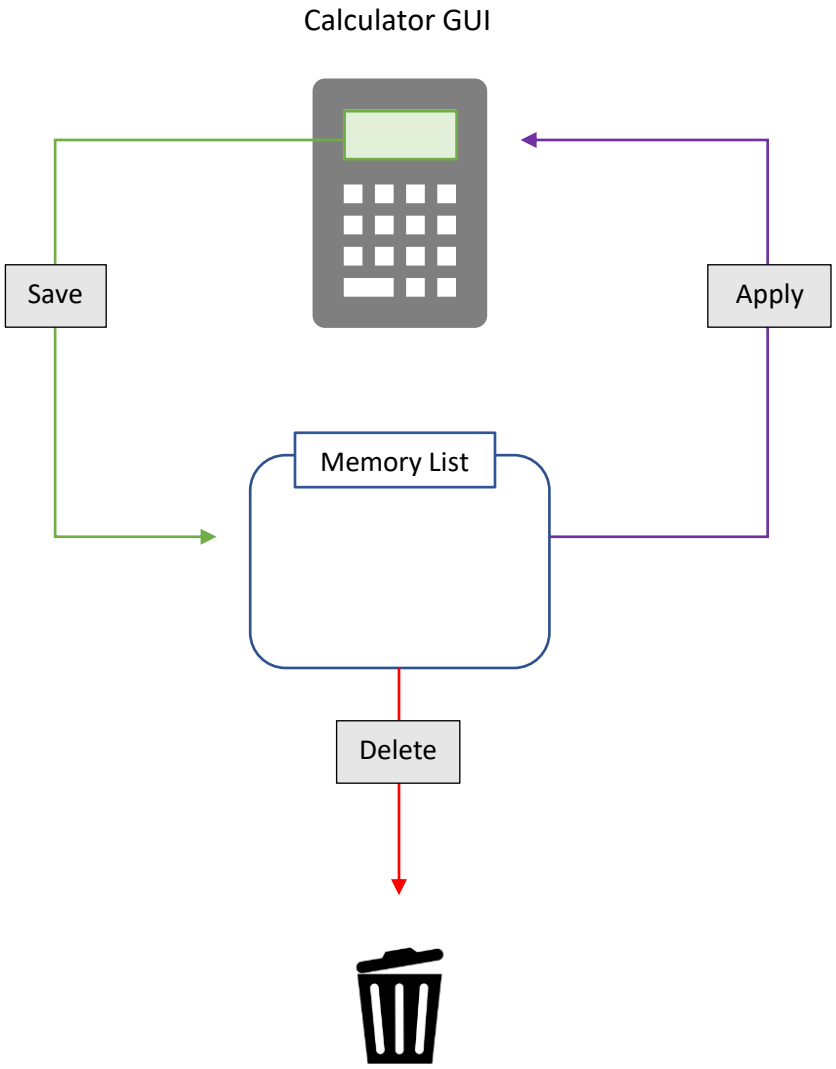
3.1 Design & Layout

Note: there may constraints that will help avoid invalid user input. This will help speed up the validation process.

Expression Evaluation Design



Memory Design



Note: The above picture shows the process of the memory for the calculator, according to the buttons pressed (Save; Apply, Delete).

3.2 Class Structure

Calculator GUI	
Object	*Description*
Input Text Field	“ Displays the Users input expression ”
Output Text Fields	“ Displays conversions of the performed operation “
Arithmetic Buttons	Arithmetic operators: (+) (-) (*) (/)
Alphabetic and Numerical Buttons	Letters: (A – F); Numbers: (0 – 9)
Logical Buttons	Logical operators: (AND) (OR) (NOT)
Notation Radio Buttons	Numerical Notation Selection: (HEX) (DEC) (OCT) (BIN)
Clearing Buttons	Backspace, Clear Entry : “deletes input made from user”
Equal Button	“ Severs as a submission button to carry out the expression inputted by the user ”
Memory Buttons	Save, Apply, Delete, Clear All : “Allow user to save expressions”
Memory List	“ Will Display the expressions and their results that were saved by the user “

Button Handler	
Object	*Action handling*
Arithmetic Buttons	" Will send the text of the button pressed to the Input Text Field so user will be able to view their input entry before submitting the expression "
Alphabetic Buttons	
Numerical Buttons	
Logical Buttons	
Memory Buttons	" Will add, remove, or copy the expression in the Memory List according to which memory button is pressed "
Notation Radio Buttons	" Will disable buttons that are deemed unnecessary provided by the selected numerical notation "
Equal Button	" Will call validation class to ensure the expression is valid, if so the expression will be executed, and the result will be converted into the other numerical notations and displayed in the according Output Text Fields "

Validation		
Function	*Description*	*Return*
Operator Validity	“ Will first check if the expression starts/ finishes with an operator. “ “ Secondly will check if an operator has a consecutive operator ”	True / False “ According to the validity of the expression “

Computation		
Function	*Description*	*Return*
Compute Input	“ Will prepare the expression to be evaluated“	Expression
Evaluate	“ Will execute the expression until there is a result “	Decimal result
Convert	“ Convert decimal result from Compute function to Decimal “	Decimal Result
To Decimal	“ Convert number into decimal notation“	Decimal