EECS 448 – Software Engineering I Project 03 System Documentation

Group 08:

Huzaifa Zahid

Mir Shazil Faisal

Junyi Zhao

Zhenzhou Wang

Story Point Analysis of Previous EECS Projects

1	2	3	5	8	13
Lab 1 -Simple Printing Exercise	Lab 6-Array reversal, file I/O	Lab 3-Lidear Sensor	Lab 10-Intro to OOP	Lab 11- DMV Class	Lab 7 -Recursion& Backtracking
Lab 4 - Fibonacci Numbers, ASCII Conversion	Lab 1- Command-line argument file manipulation	Lab6 – Recursion exercises	Lab4 – Elevator(Queues&Stacks)	Lab9,Lab10- Binary Search Tree	EECS 448-project1
Homework Assignment 1	Lab8 – Time Complexities	Homework Assignment 2	Homework Assignment 3	Lab5-Browser Tracking (List)	EECS 448-Project2
	Homework Assignment 4	Lab 6-Unified Modeling Language		EECS 448-Project3	Homework Assignment 5

Legend:

	EECS 168
	EECS 268
	EECS 368
	EECS 388
	EECS 448

Estimate of Hours/Story Point:

1 hour/story point.

Initially we placed this project under 13 story points because we thought it would be difficult for us to familiarize ourselves with the GUI part. This is also the reason why we chose to make an arithmetic calculator for this project so that we could ease into the frontend programming aspect of things. However, when deciding which platform and language to work on, we realized that our EECS 368 knowledge of HTML and JavaScript could come in handy for this project. This allowed us to reduce the story points down to 8 and allocate 1 hour per story point. Even though the calculator was a bit more challenging, we were taught to work with Event Listeners and make buttons in EECS 368, which were an integral part of this project.

1. Introduction:

Our project 3 is a calculator program implemented in JavaScript, HTML and CSS that performs addition, subtraction, multiplication, and division. We also created a windowed interface for this calculator. After a group discussion, we are going to improve the calculator in the project 4 into a scientific calculator that can calculate slightly more complex trigonometric and calculus equations.

2. Background:

Calculator is one of the most basic features of a modern computer. Other than AI, modern calculator software reused lots of features in the past. Today, we would like to write our own, to mark our debut in the world of software engineering.

3. Code structure and detail:

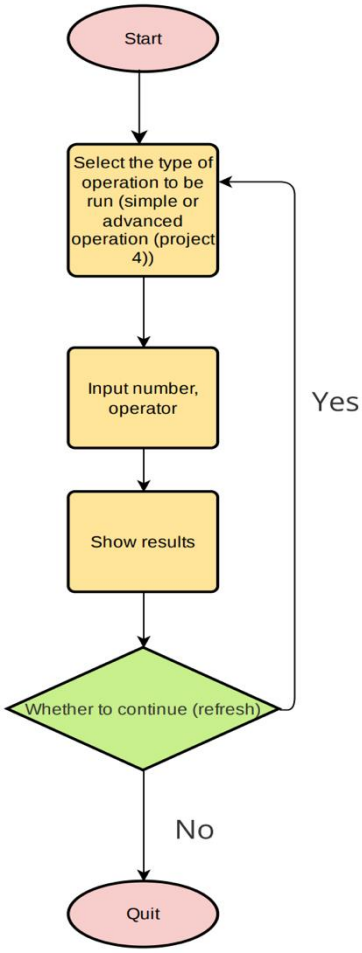
Basic design ideas:

(1) Enables four operations (addition, subtraction, multiplication, and division).

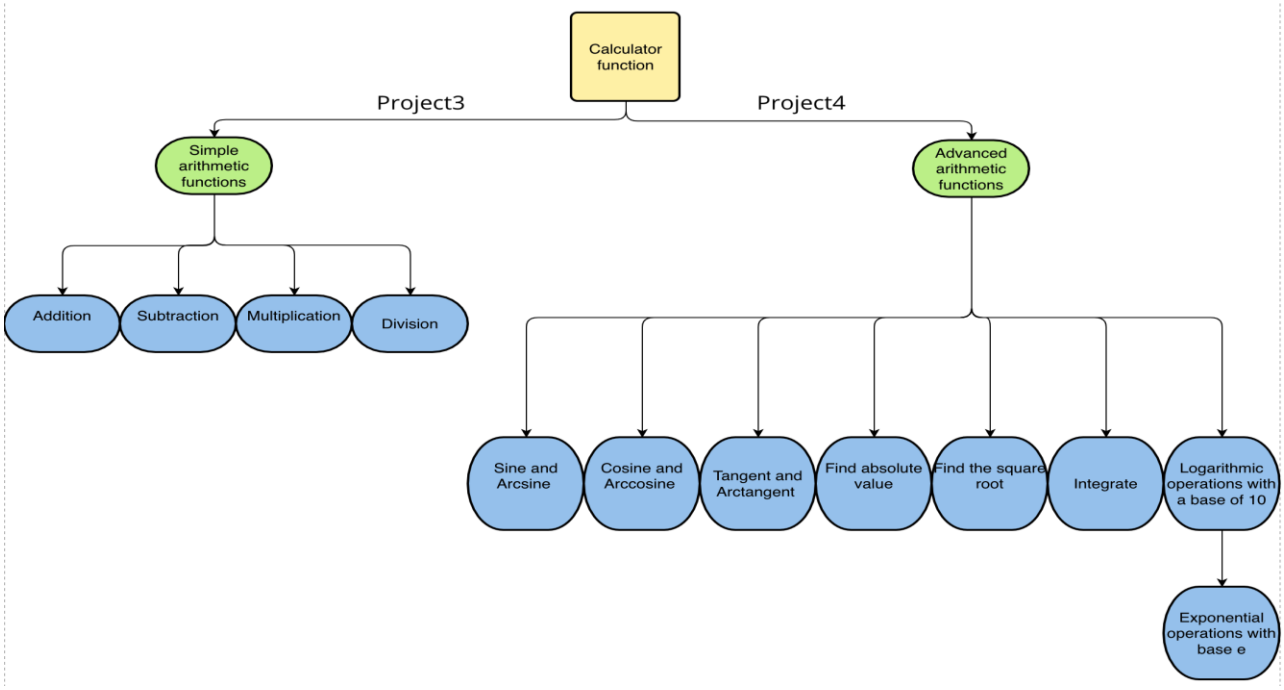
(2) Can realize multiple consecutive calculations. We have buttons for parenthesis which allows the user to determine the order of precedence of the operators.

(3) For now, the maximum length is not limited to the maximum number of digits that can be accommodated on the display, and an ERROR message would be shown only when an equation is of invalid format.

Project 3 Calculator Flow Chart



Calculator function module diagram



4. Design Paradigm:

The design paradigm we chose for this project is mainly functional-oriented because we planned on providing each component with a function that is clearly defined, so have functions that do something specifically rather than classes. We are using EventListener to check for the buttons being pressed and evaluate the expression accordingly. JavaScript is also object-oriented along with functional, and even though we did not use classes, we did employ some aspect of object-oriented programming. For example, linking index.html to index.js like how different classes are linked to each other.

Other than this, if object-oriented design is thought of as being a top-down approach to design that focuses on the real-world design containing classes that serve as objects and having methods that relate to those objects, then it can be said that our calculator has different methods to evaluate different operations. However, our project mainly uses functional-oriented design paradigm.

5. Software Architecture:

A calculator is rather a single-thread process than other patterns containing mainly buttons and button-listening events, we decided to use event driven architecture.
Different from two older projects in the class, calculator does not have a procedure: there is no “place” and “play” sections. The only thing we have here are buttons, click events, and string evaluations. Therefore, pipe and filter architecture does not apply to project 3.
We also do not have connection issues. For online software, interaction applications, contact between back end and front end is necessary. Therefore, client-server communication is a good architecture for them. To us, however, the calculator is no more than a single thread, single machine project. There is no merit to open or reserve any client-server communication.

However, a calculator can still require communication. This is when the calculation is extremely huge. For example, when calculating the ratio of a circle’s perimeter to its diameter, known as pi, to its billionth digit. Under this scale’s calculation, a single computer’s calculator can never work. Under this circumstance, cloud computing and/or connection to a supercomputer is necessary. For supercomputers, their power of calculation is distributed. Under that, a waitlist and application system are also needed. Under such circumstance, a server-client communication is needed. it could also be possible that P2P architecture is needed.

But our calculator is not that advanced. At the final section of its life, it deals no more than trigonometric functions or calculus. This makes thoughts of creating a more complex architecture lack of merit.

6. Design Pattern:

In the project3 calculator application, one of our design patterns should be Builder under the Creational Design Pattern.

The builder, as we saw in our EECS 448 lecture, allows you to create complicated things step by step and to make multiple types and representations of the same item using the same construction code.

A builder pattern is a design pattern that allows us to isolate the object building from its own class and reuse it across numerous representations. One advantage of using this pattern is that it allows us to create objects by layering operations on top of each other, so we don't have to call all of them at the same time, only the ones that are required to achieve a certain result.

In a calculator, the result of an operation is the result of superimposing numbers and various functions to obtain the result by adding, subtracting, multiplying, and dividing in a superimposed manner. In our project 3, we used the eval() algorithm to get the same effect.

Since our project requires future modification and addition of new features, a builder like that gives us ease. It would be hard for eval() function to deal with trigonometric or calculus function, so project 4 may end up with twice size of code than project 3. We have to prepare for additional functions and features. This will not only save us time, but also avoid additional changes of project 3 code. Creational design pattern is a great way to foresee and deal with such requirement.

6. Conclusion:

Trying to do a project in a different language for the first time was a new challenge, and it was also very helpful for our teamwork and innovation.