**Calculator**

**By Jobanpreet Singh**

**System Analysis and Design (INFO 1113) S10  
group Braeden Scott (100413491) , Simran (100404857), and Jobanpreet Singh (100402373)**

**November 30, 2021**

<https://sites.google.com/d/1Xle97F-o2jY9QdWeOzXDZNOqKG41wbil/p/1kNGKuEXhSvM8idFDn-A1bfehaOQPPSxi/edit> (Joban’s Website)

[jobanpreet.singh7@email.kpu.ca](mailto:%6a%6fb%61%6e%70%72ee%74%2e%73%69%6e%67%687@e%6d%61%69%6c%2ekpu%2e%63a)

**Table of Contents**

1. Project Requirements pg. 3
2. Use-Case diagram and Use-Case description pg. 6
3. Class diagram pg. 8
4. Report pg. 9
5. Websites pg. 10
6. Work cited pg. 11

**Project Requirements:**

Building a web calculator is a great project, especially if you have just started learning JavaScript. It is quite simple for people of any skill level. This project covers the interactions with UI and key JavaScript methods.

Prerequisites

* Any good Text editor.
* Basic understanding of JavaScript and HTML.

Designing the calculator

To get started, you need to consider the basic functionalities of a calculator. They include addition, subtraction, multiplication, division, delete, all-clear, and of course, the ability to use decimal numbers in performing these operations.

In your text editor, create three separate folders for your HTML, CSS and JavaScript. This just basically makes your code more organized.

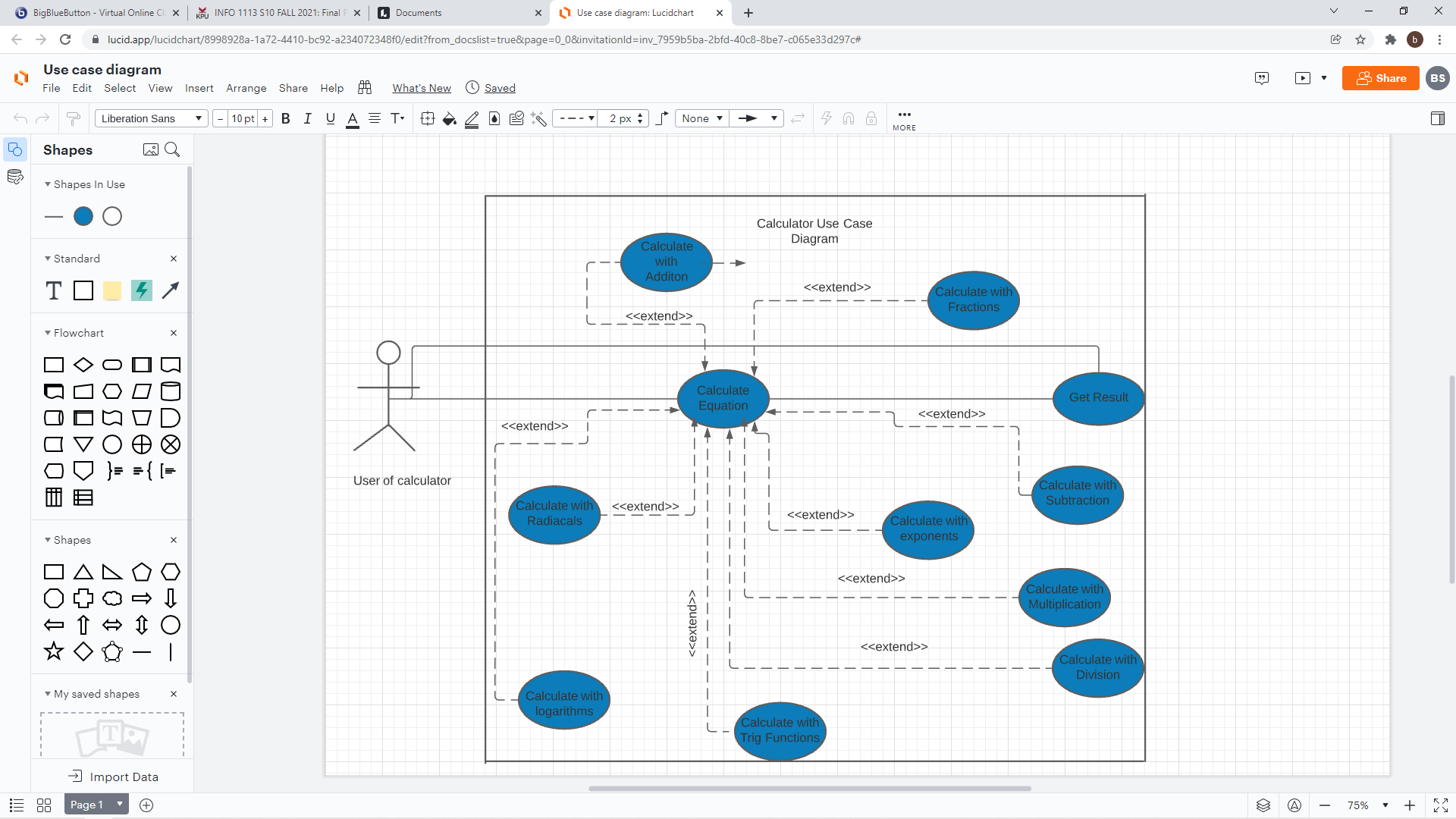
For making the structure of the project, we will use HTML and we need to style the calculator using CSS, and for organizing the project, we will use JAVASCRIPT.

Here are some use cases (abilities your project needs to have):

1. Your calculator is going to contain functions for all of the basic math operators you typically find on simple calculators, so start by creating functions for the following items and testing them in your browser’s console.
   1. add
   2. subtract
   3. multiply
   4. divide
2. Create a new function operate that takes an operator and 2 numbers and then calls one of the above functions on the numbers.
3. Create a basic HTML calculator with buttons for each digit, each of the above functions and an “Equals” key.
   1. Do not worry about wiring up the JS just yet.
   2. There should also be a display for the calculator, go ahead and fill it with some dummy numbers so you can get it looking right.
   3. Add a “clear” button.
4. Create the functions that populate the display when you click the number buttons… you should be storing the ‘display value’ in a variable somewhere for use in the next step.
5. Make the calculator work! You’ll need to store the first number that is input into the calculator when a user presses an operator, and also save which operation has been chosen and then operate () on them when the user presses the “=” key.
   1. You should already have the code that can populate the display, so once operate () has been called, update the display with the ‘solution’ to the operation.
   2. This is the hardest part of the project. You need to figure out how to store all the values and call the operate function with them. Don’t feel bad if it takes you a while to figure out the logic.
6. Gotchas: watch out for and fix these bugs if they show up in your code:
   1. Users should be able to string together several operations and get the right answer, with each pair of numbers being evaluated at a time. For example, 12 + 7 - 5 \* 3 = should yield 42. An example we’re looking for would be like- **Your calculator should not evaluate more than a single pair of numbers at a time. If you enter a number, then an operator and another number that calculation should be displayed if your next input is an operator. The result of the calculation should be used as the first number in your new calculation.**
   2. You should round answers with long decimals so that they don’t overflow the screen.
   3. Pressing = before entering all of the numbers or an operator could cause problems!
   4. Pressing “clear” should wipe out any existing data. make sure the user is really starting fresh after pressing “clear”
   5. Display a snarky error message if the user tries to divide by 0… don’t let it crash your calculator!
7. EXTRA CREDIT: Users can get floating point numbers if they do the math required to get one, but they can’t type them in yet. Add a. button and let users input decimals! Make sure you don’t let them type more than one though: 12.3.56.5. It is hard to do math on these numbers. (disable the decimal button if there’s already one in the display)
8. EXTRA CREDIT: Make it look nice! This can be a good portfolio project… but not if it’s UGLY. At least make the operations a different colour from the keypad buttons.
9. EXTRA CREDIT: Add a “backspace” button, so the user can undo if they click the wrong number.
10. EXTRA CREDIT: Add keyboard support!

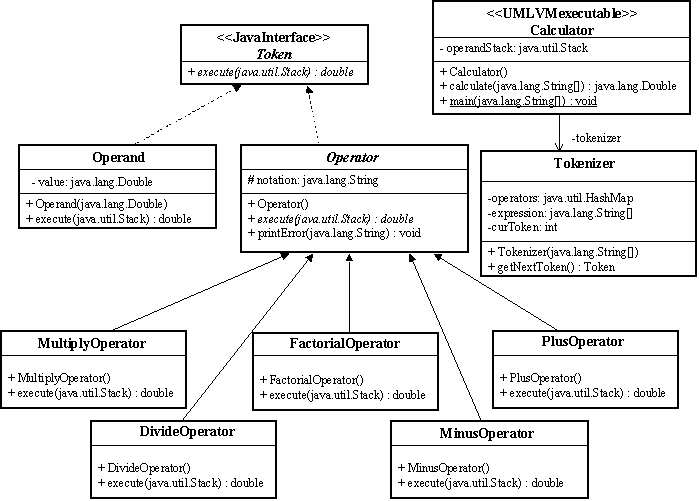
**USE CASE DIAGRAM DESCRIPTION**

Use Case Diagram:



|  |
| --- |
| Use Case Title: Calculator |
| Primary Actor: User of the calculator |
| Level: White |
| Stakeholders: User of the calculators, Programmers |
| Precondition: User opens up the calculator program and enter the data |
| Minimal Guarantee: Recalculate the error calculation |
| Success Guarantee: The result of the given data instructions |
| Trigger: Starting to analyse the given data instructions |
| Main Success Scenario:   1. User gives the data instructions 2. User presses the enter button or clicks the equal button. 3. Calculator analyse the given data instructions and calculates the data instructions. 4. Calculator shows the result of the given data instructions. |
| Extension:  1a. Error message appears because invalid data instruction was inputted.  1a1. User presses the clear button and starts over.  1a2. User exit from the program.  2a. Too big of a number error appears.  2a1. User clicks the clear button and starts over.  2a2. User closes program.  2b. Too small of a number error appears.  2b1. User clicks the clear button and starts over.  2b2. User closes program.  3a. Calculation is interrupted.  3a1. Calculation is rolled back to the beginning. User starts over  3a2. Calculation is rolled back to the beginning. User closes  program. |

**Class Diagram:**

****

**Personal Report:**

In the group project, I really learnt a lot through working in a group. We choose calculator our main project, as it’s the fundamental need of any work in the modern era. We have to calculate a lot of data equations in day to day tasks. In this project, like my other group mates, I make a website for my project. In which, I have discussed the whole structure of the group. Apart from this, I have my own and group tasks, which I did. I have researched out the project requirements and make a word document file for it. I did use- case diagram description and class-diagram during making the project.

**Websites**

Trello: <https://trello.com/invite/b/obJed7rq/ca5ed56804526f8fa8ef333be4a9fa09/final-project>

GitHub group: <https://github.com/DarkShadowGithub/Group-Cs-Group->

Group’s Website: <https://sites.google.com/view/system-analysis-s10/home>

Braeden’s GitHub: <https://github.com/DarkShadowGithub/Group-C>

Simran’s GitHub: <https://github.com/Sim-09/Use-case-title>

Joban’s GitHub: <https://github.com/Aulakhjoban/Final-Project>

**Work Cited**

Petersen, C. (n.d.). *What is a software calculator?* What Is a Software Calculator? (with picture). Retrieved September 30, 2021, from <https://www.wise-geek.com/what-is-a-software-calculator.htm>.

Anonymous. (2021, August 16). *What is a calculator?* What is a Calculator? Retrieved October 1, 2021, from <https://www.computerhope.com/jargon/c/calc.htm>

Contributor, T. T. (2005, September 21). *What is calculator? - definition from whatis.com*. WhatIs.com. Retrieved December 2, 2021, from <https://whatis.techtarget.com/definition/calculator>.