Description of the Code

The code features three parts: the Calculator, the Parser and the main class PostFixProgram.

The parser is the first part used, to read a CSV file and translate it into a grid of strings. The first possible limitation of my project is the size of the grid of strings. For the sake of the example, I chose to set the size as 3 by 3, however this would mean there would be issues if the inputted CSV file was larger than the example one. I did also contemplate doing a 10 by 10, but that would sacrifice performance, as the for loops would loop through many null elements when values aren’t present.

The main loop code is the traverseCells method which runs two for loops in the hope of translating all the cells. The final limitation here is the use of two nested four loops, which isn’t ideal aesthetically or performance wise, however at the time I couldn’t think of a better way to go through the values again to insure in examples like the B1 B2, are picked up and accounted for, as they wouldn’t be at times before the B1 and B2 cells aren’t populated, which would be the case when the code first goes through the cells.

The logic of the code is the calculateEquations method, which first ensures a string isn’t empty, null or just a blank space and then splits it into an array of smaller strings, before choosing between whether it’s Cell information, in which it will replace the cell with the cells value and place that in a stack of floats, whether it’s an operator, in which it will perform a calculation and place the value within the stack of floats, or whether it’s a number, in which it will parse the float and place It in the stack of floats. At the end there should be only one value, which will be formatted to make sure it doesn’t show a decimal unless necessary.

Finally, the writeFile method uses a path similar to the one presented as an argument, and uses a Stringbuilder to take the information from each row and store it, then a bufferwriter to save the information to a file.