Description of the Code

The code features three classes: A Calculator, a Parser and the main class PostFixProgram. This was done just for the sake of not making the main class look too bloated, it would have been just as easy to keep the two methods within the PostFixProgram, but I felt it just was a tad more aesthetically pleasing.

The CSVParser class is the first that is used, in order 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.

Next the code uses the calculate method to go through each cell and run the prepareString method. This method exists in the case of equations that use A1, B2 etc and handles swapping out the values. Another limitation is the if statement used to determine whether an expression is of “Letter/Number” format, as it sticks to a length of 2 and checks for numbers and letters. If the size of the csv went into double digit letters of numbers, this would cause a problem, so in future refactoring that to account for such case would be good.

Finally the main code is the calculate 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.