Criterion B

Word count: 260

Table of Contents

Text-based user interface (TUI) – proposed design

```
Console

Type number of Equations
2
Type the number of Variables
2
What is the coefficient of variable 1 in equation 1
2
What is the coefficient of variable 2 in equation 1
1
What is the constant of equation 1
3
What is the coefficient of variable 1 in equation 2
3
What is the coefficient of variable 2 in equation 2
1
What is the constant of equation 2
3
```

// The user input 2 equations with 2 unkowns.
The user is reading and typing this system

$$2x_1 + 1x_2 = 3$$
$$\underbrace{3x_1 + 1x_2}_{\text{Coefficients for Equation 2}} = \underset{\text{constant}}{3} \leftarrow \text{Equation 2}$$

Figure 1: Input format

```
Console

x1=0
x2=3

Enter the number 1 to display the RRED
1
```

$$\left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 3 \end{array}\right]$$

// Displays
$x_1 = 0$
$x_2 = 3$

// Through Gaussian manipulations, the RREF looks like this in a non-augmented form

$$x_1 + 0x_2 = 0$$
$$0x_1 + x_2 = 3$$

// Augmentation Just means we remove the variables for display purposes

$$\begin{array}{cccc} \left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 3 \end{array}\right] & \rightarrow \text{constant} \\ x_1 \quad x_2 \quad = & \end{array}$$

Figure 2: Output Format

Although the client's consulted that the program's purpose should be handling large systems. The example above is just a mock illustration for the purpose of understanding the program.

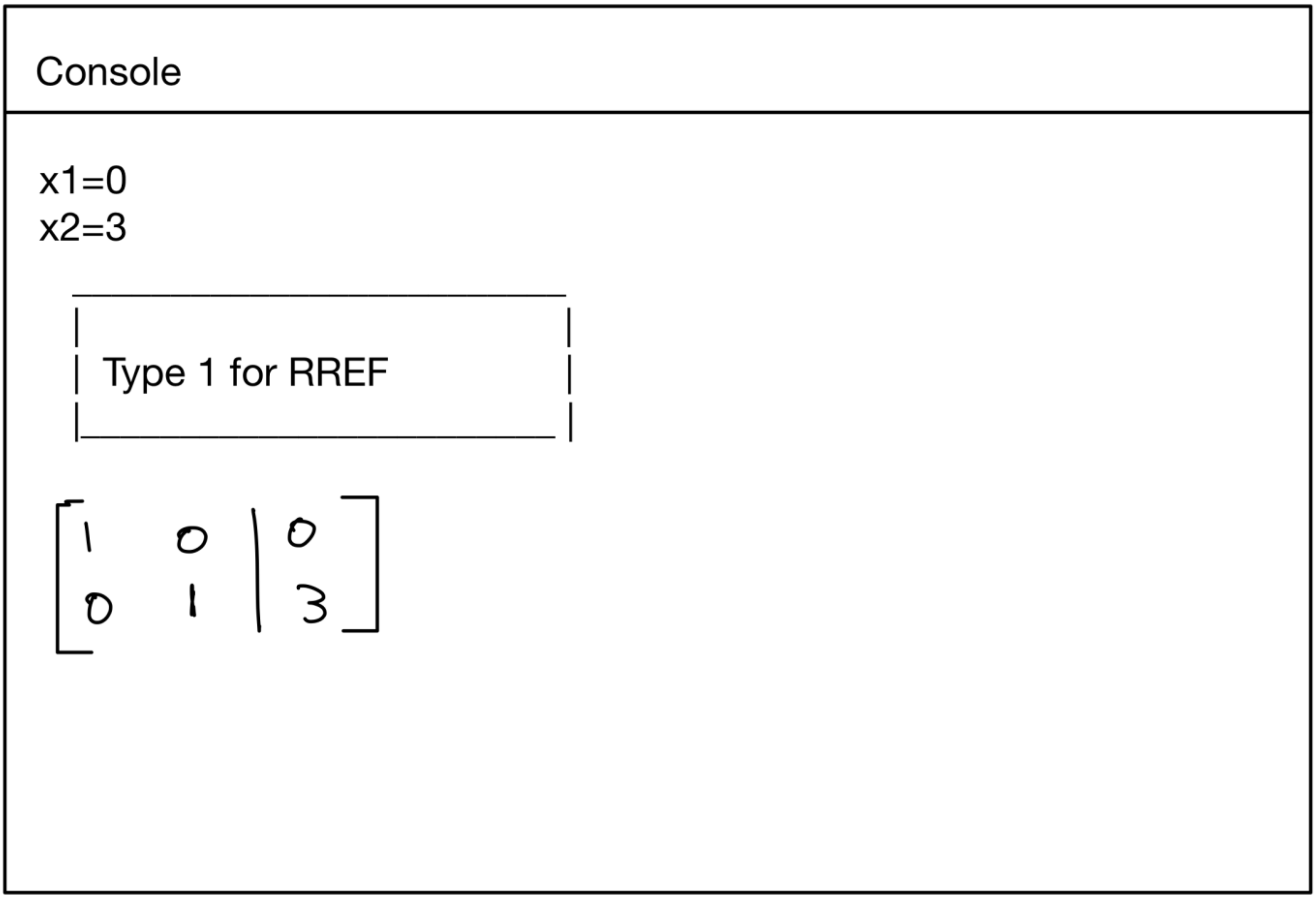

Figure 3: Output Format amended

Data Structure

I discussed with my client about the size of the system. The client requested the product to handle 3 cases which is if the system has more equations that variables and constants, more variables and constants than equations, and equal number of equations and variables. Through, this request, I used a 2d array where the rows represent the equations, and the columns represent the variables and the constant. However, as there is no specified number of rows and columns, the 2d array would need to be a data structure as the rows and columns are independent from each other.

I told my client that he would need to augment his system first, meaning that the constants would need to be and put in the right-hand side of the equations.

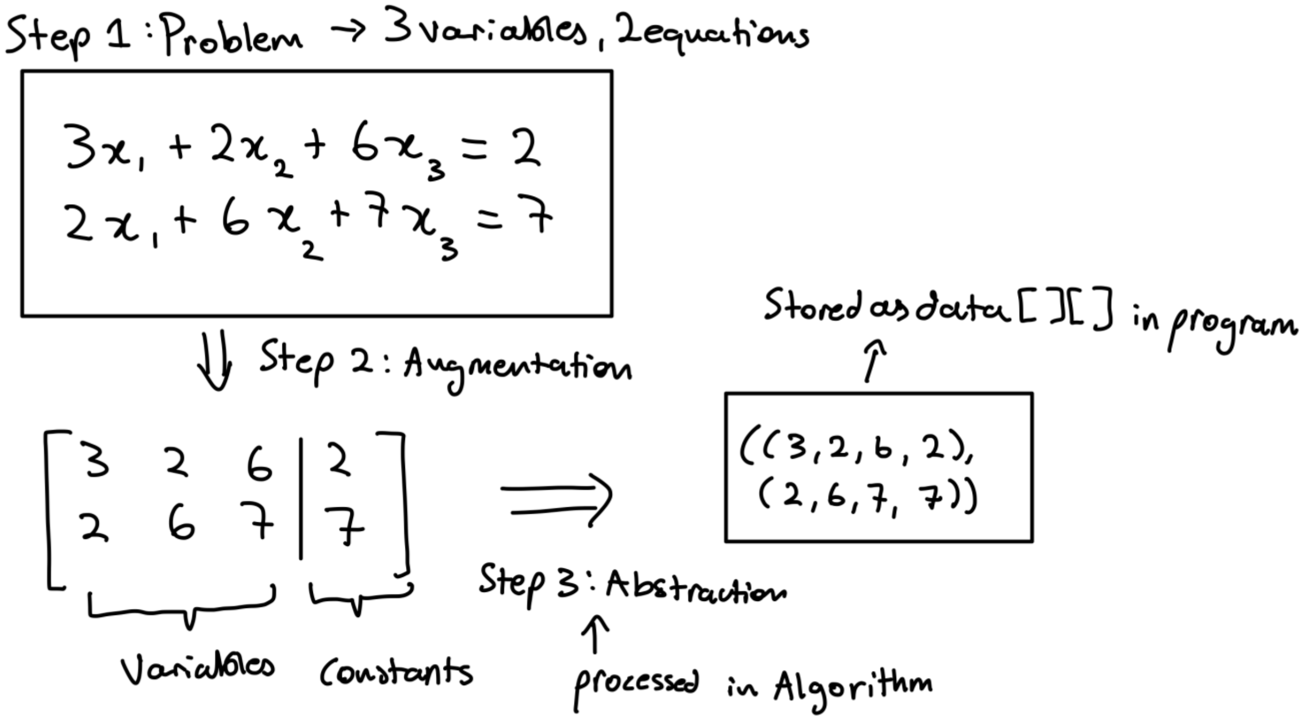

Figure 4: Steps on abstraction

Flowchart

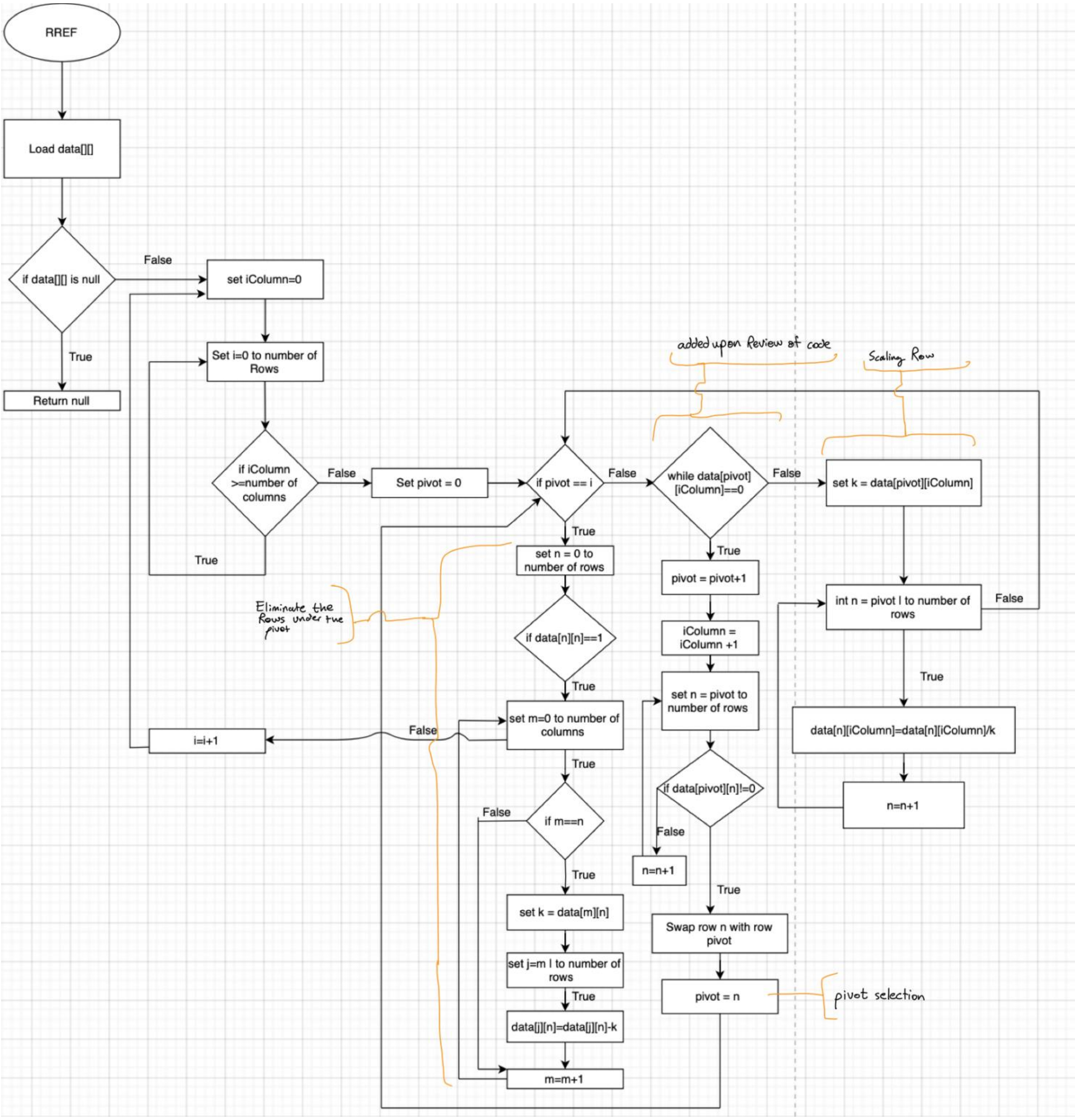

Figure 5: Finding the RREF:

The flowchart above is an algorithmic method to find the Reduced Row Echelon Form of a system. When creating the flow chart, I tested it out with a 2d array but I realized that there was an error if the pivot value was a 0. This meant that k would be made to 0 in the scaling part which would cause an error as an integer divided by 0 is undefined. As a result, I created a while loop that increments the pivot and iColumn when a 0 is detected.

Pseudocodes ReadSolution method

This method relies on traversal patterns to display the solutions once the RREF form is calculated

a.Number of equations equals the number of variables

```
If numRows = numVariables THEN

//First we need to check if all of the diagonal is non zero

    UniqueSolution = true
    infiniteSolution = true

    For iRow from 0 to numRows-1 DO   pivot is just the initial diagonal entries.
        IF RREF[iRow][iRow]=0 Then    ↳ Check if the pivot elements are non zero
            uniqueSolution = false
            Break out of loop             → A zero pivot means it is impossible for a unique
        END IF                              Solution to exist
    END FOR

    If UniqueSolution is True THEN ⇒ if all pivots are non zero then output the Unique solution
    //if all pivots are non zero than a unique Solution may exist
        FOR iRow 0 to numRow Do
            Output("x" + (iRow+1)+ "=" + RREF[Row][numCols-1])
        END FOR
        infiniteSolution = false ⇒ No free Variables

    ELSE
        // If a pivot is zero, check if the row represents an infinite or no solution
        FOR iRow from = to numRows -1 DO
            IF RREF[iRow][iRow]=0 AND RREF[row][numCols] != 0 THEN
            no solution =true        ↳ check if pivot is zero and the constant term is non-zero
                FOR iCols FROM = to numCols-2 DO
                    IF RREF[iRow][iCols] != 0 THEN ↳ Verify that all coefficient entries are zero except
                        noSolution =false              Constant term
                        BREAK out of loop
                    END IF
                END FOR
                IF noSolution == TRUE THEN
                    Output("The System has no Solutions")
                    EXIT method   ↳ when inconsistent Row is found. All variable coefficients are zero and constant term
                END IF              is not zero
            END IF
        END FOR
    END IF

    IF infinteSolution == true THEN          Ex : 2x1 = 2+3x3
        //Will print in parametric form              x2 = 1 + 6x3
        FOR iRow FROM 0 to numRows-1 DO              x3 = x3
            IF RREF[iRow][iRow] !=0 THEN
                equation = "x" + (row+1) +"=" + RREF[iRow][numCols-1]
                FOR iCols FROM iRow+1 to numCols-2 DO
                    IF RREF[iRow][iCols]!=0 THEN
                        equation = equation + "(" + (-RREF[iRow][iCols]) + "x" + (iCols+1)+ ")"
                    END IF
                END FOR
                Output(equation)
            END IF
        END FOR
    END IF
```

Figure 6: Pseudocode algorithm when equal number of equations and variables

b. More Variables Than Equations

```
ELSE IF numVariables > numRows THEN
    IF (RREF[numRows-1][numVariables-1]=0) AND (RREF[numRows-1][numVariables]!=0) THEN
        Output "System has no Solution"  → Check for inconsistency in the last Row.
        EXIT method                        As if a zero coefficient variable equal zero then
    END IF                                 this is a error, as 0 ≠ k
        Process the rows for printing              ↑
    FOR iRow FROM 0 to numRows-1 DO              constant
        IF RREF[iRow][iRow]==1 THEN
            equation = "x" + (row+1)  → only when pivot is 1.
            firstOutput = false
            For iCols From iRow+1 To numVariables DO ← collects free variables for infinite solution.
                IF iCols = numVariables THEN ⇒ if no free variables
                    constant term = RREF[iRow][iCols]       One variable is equal in terms
                    IF firstOutput ==false THEN             of another variable
                        equation = equation + "=" + constantTerm
                        firstOutput = true
                    ELSE
                        equation = equation + "+" + constantTerm
                    END IF
                ELSE IF RREF[iRow][iCols] !=0 THEN ← process the coefficient as a free variable
                    coefficient = -1*RREF[iRow][iCols]
                    IF firstOutput == false THEN
                        equation = equation + "=" + coefficient + "x" + (iCols+1)
                        firstOutput = true
                    ELSE
                        equation = equation + "+" + coefficient + "x" +(iCols+1)
                    END IF
                END IF
            END FOR
            Output(equation)
        END IF
    END FOR
```

Figure 7: Pseudocode algorithm when more variables than equations

c. More equations than variables

```
ELSE IF numVariables < numRows THEN
   FOR iRow = 0 TO numRows - 1 DO ⇒ Check for inconsistent rows
      allZeros = true
      FOR iCols = 0 TO numCols - 2 DO
         IF RREF[iRow][iCols] != 0 THEN
            allZeros = false
            BREAK
         END IF
      END FOR                    → Case when inconsistent row is verified
      IF allZeros == true AND RREF[iRow][numCols - 1] != 0 THEN
         PRINT "System has no solution"
         EXIT Procedure
      END IF
   END FOR

   uniqueSolution = true       → Check if unique solution exist by checking non-zero pivot
   FOR iRow = 0 TO numRows - 1 DO
      IF iRow < numVariables AND RREF[iRow][iRow] == 0 THEN
         uniqueSolution = false
         BREAK
      END IF
   END FOR
                               → if unique solution exist then output each variable value
   IF uniqueSolution == true THEN
      FOR iRow = 0 TO MIN(numRows, numVariables) - 1 DO
         OUTPUT( "x" + (iRow+1) + " = " + RREF[iRow][numCols - 1])
      END FOR
   ELSE
      FOR iRow = 0 TO numVariables - 1 DO ⇒ otherwise print in parametric form
         IF RREF[iRow][iRow] != 0 THEN
            equation = "x" + (iRow+1) + " = " + RREF[iRow][numCols - 1]
            FOR iCols = iRow + 1 TO numVariables - 1 DO
               IF RREF[iRow][iCols] != 0 THEN
                  equation = equation + " + (" + (-RREF[iRow][iCols]) + "x" + (iCols+1) + ")"
               END IF
            END FOR
            OUTPUT(equation)
         END IF
      END FOR
   END IF
                         → Additional consistency check
   FOR iRow = 0 TO numRows - 1 DO
      FOR iCols = 0 TO numCols - 2 DO
         IF RREF[iRow][iCols] == 0 AND RREF[iRow][iCols+1] != 0 THEN
            OUTPUT( "System has no solution")
            EXIT Procedure
         END IF
      END FOR
   END FOR
END IF
```

Figure 8: Pseudocode algorithm when equal number of equations and variables

UML diagram

Figure 6: UML diagram in program

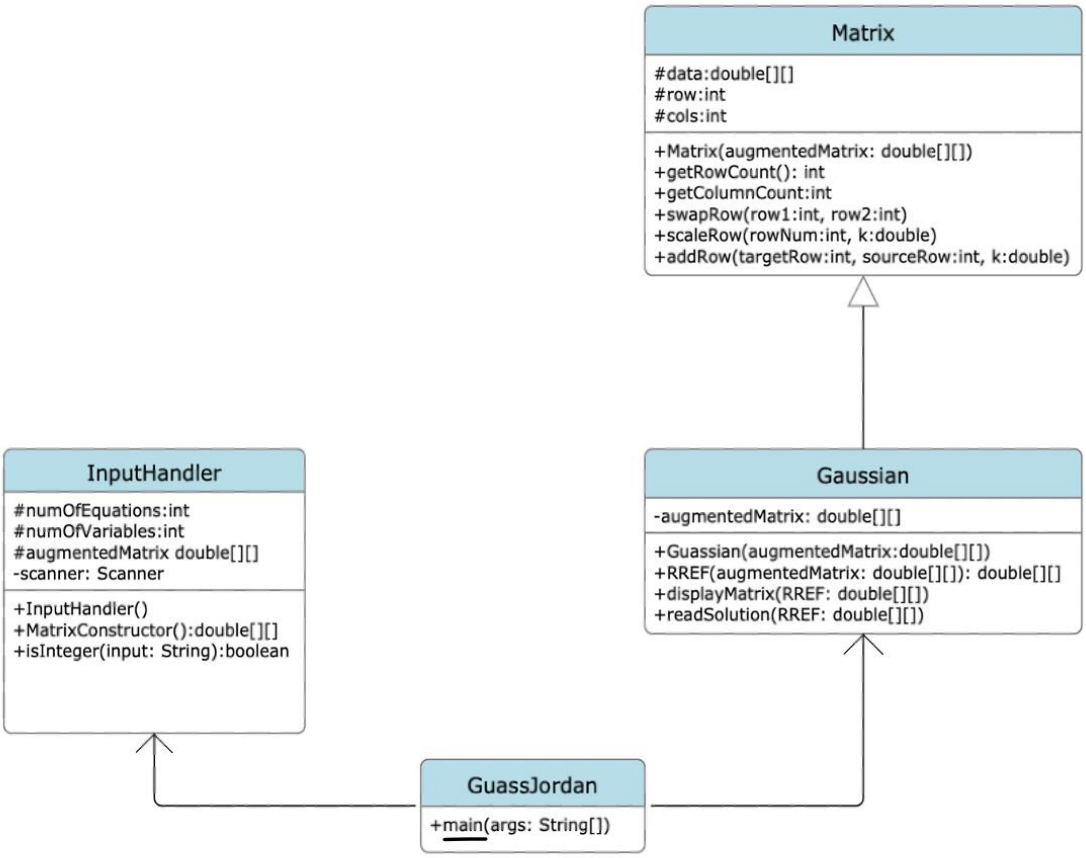

Figure 9: UML diagram

Test Plan

Test number	Success Criteria	Description of test	Test method/ Expected outcome
1	Client can specify number of equations and variables for the system	The client is promoted to enter the number of equations and number of variables. The program must accept valid integer inputs (minimum of 2 for each as required by the algorithm)	Run the MatrixConstructor via the InputHandler Class. The system accepts the input and initialized a 2d augmented matrix of size $equation \times variable + 1$
2	Enter Coefficients and constants	The client enters each coefficient and constant for every equation. The order should be intuitive.	Use the MatrixConstructor method to input each number which valid numeric outcomes.
3	Exception handling when there are invalid inputs	The program must detect and handle invalid input types (e.g., decimal when an integer is excepted or a string when a number is expected)	Manually enter invalid data when prompted. The program catches the InputMismatchException and prompts the client to re-enter the value without crashing.
4	MatrixConstruction Accuracy	Verify that all inputs are stored correctly.	After entering the test values use a nested for loop to print the matrix. This will give me confidence that the client's provided data us accurately captured.
5	Matrix operations (swap, scale, and add) work correctly on their own	Independently test the basic matrix operations using small 2d arrays.	Preform the matrix operation individually on a test array and print them using a nested for loop.
6	Correct RREF Calculation	Test the Gaussian elimination algorithm with pre-solved system of linear equations. This includes cases with unique, infinite, or no solution.	Supply predetermined matrices to RREF method. The outcome should exactly match the RREF.

7	Accurate solution interpretation	After the RREF is computed, the program interprets the augmented matrix to display the solutions using the readSolution method.	Use known test cases. The output should correctly show each variable's value, provide parametrized answers for infinite solution, clearly indicate when there is no solution.
8	Display the RREF form	After processing, the client should have the option to display the RREF from. There should be a visual divider between the coefficient and constant in each equation.	Run the main method, choose to display option and observe the formatted output from displayMatrix. The outcome should be the RREF in a augmented form.
9	Test the algorithm for edge cases	Test the system that have more equations that variables or more variables than equations.	Provide specific input cases for each scenario and observe the RREF and the output solution.
10	Handling negative and decimal numbers	Verify that the program can correctly compute the RREF and final solution when coefficients are constants and negative numbers	Input a test system that has negative coefficients and decimal values. The RREF should be interpreted accurately.

The test table above provides a structured roadmap that benefits development and testing whilst building clarity for the clients need.