Source Code:

```java
import java.io.*;
import java.util.Scanner;

/**
 * @author Joseph Siwiecki
 *         Assignment: Programming Project 1
 *         Class: CS 3010.01
 *         Date: 10/9/22
 */
public class Project1_jsiwiecki
{
    /**
     * This program gets user input for a system of linear equations
     * either through the command line or through a file,
     * and uses Gaussian Elimination with Scaled Partial Pivoting
     * to solve the system. The intermediate steps are displayed,
     * and the solution is displayed.
     *
     * Using Gaussian Elimination with Scaled Partial Pivoting will be
     * more accurate and precise than using Naive Gaussian Elimination or even
     * just standard Partial Pivoting.
     *
     * @param input             Used to gather user input.
     * @param numberOfEquations Number of equations in the augmented matrix.
     * @param augmentedMatrix   Augmented matrix that stores user input for
     *                          coefficients.
     * @throws FileNotFoundException If file is not found, an FNFE will occur.
     */
    private static Scanner input = new Scanner(System.in);
    private static int numberOfEquations;
    private static double[][] augmentedMatrix;
```

```java
/**
 * @param args
 * @throws FileNotFoundException
 */
public static void main(String[] args) throws FileNotFoundException
{
    getNumberOfEquations();
    addEquationsToArray();
    gaussianEliminationWithScaledPartialPivoting();
    printAnswerArray();

    input.close();
}

/**
 * Retrieves number of equations from user and stores the input
 * into numberOfEquations.
 */
private static void getNumberOfEquations()
{
    boolean valid = false;

    while (valid == false)
    {
        System.out.print(s: "# of linear equations to solve: ");
        int tempNum = input.nextInt();

        if (tempNum > 0 && tempNum <= 10)
        {
            numberOfEquations = tempNum;
            valid = true;
        }

        else
        {
            System.out.println(x: "Please enter a valid integer. [0 < numberOfEquations <= 10]");
        }
    }
}
```

```java
/**
 * This method gets user input for the coefficients of the linear equations,
 * and inputs them into the augmentedMatrix. The user can input the
 * coefficients from the command line, or from a file.
 * @throws FileNotFoundException If file is not found, an FNFE will occur.
 */
private static void addEquationsToArray() throws FileNotFoundException
{
    augmentedMatrix = new double[numberOfEquations][numberOfEquations + 1];

    int choice = 0, numsEntered = 0;
    boolean done = false;

    System.out.print(s: "Enter 1 to input values using CLI, enter 2 to input values using file input: ");

    /*
     * Create two more Scanners because using the standard input scanner for the whole class
     * does not operate properly. All scanners will be closed to prevent memory leaks.
     */
    Scanner eqInput = new Scanner(System.in);
    Scanner file = new Scanner(System.in);
```

```java
while (done == false)
{
    choice = input.nextInt();

    if (choice == 1)
    {
        while (augmentedMatrix.length > numsEntered)
        {
            System.out.println("Enter " + (augmentedMatrix[0].length) + " values for row " + (numsEntered + 1) + ": ");
            eqInput = new Scanner(System.in);

            String input = eqInput.nextLine();

            String[] stringArray = input.split(regex: " ");

            for (int i = 0; i < (augmentedMatrix[0].length); i++)
            {
                augmentedMatrix[numsEntered][i] = Integer.parseInt(stringArray[i]);
            }

            numsEntered++;
        }

        System.out.println(x: "\n------------ STARTING MATRIX ------------\n");
        printAugmentedMatrix();
        System.out.println(x: "-----------------------------------------");
        done = true;

    }
```

```java
/*
 * For file input, the entire file path must be specified for the file to read properly.
 * In your IDE, right click on the file to get its file path, and then use right click again
 * but on the terminal instead when prompted for the file path.
 */
else if (choice == 2)
{
    String filePath = "";

    System.out.println(x: "Enter the full path of the file, along with the file extension [Use right click to paste file path]: ");
    eqInput = new Scanner(System.in);

    filePath = eqInput.nextLine();
    File inputFile = new File(filePath);
    file = new Scanner(inputFile);

    while (augmentedMatrix.length > numsEntered && file.hasNextLine())
    {
        String input = file.nextLine();
        String[] stringArray = input.split(regex: " ");

        for (int i = 0; i < (augmentedMatrix[0].length); i++)
        {
            augmentedMatrix[numsEntered][i] = Integer.parseInt(stringArray[i]);
        }
        numsEntered++;
    }

    System.out.println(x: "\n------------ STARTING MATRIX ------------\n");
    printAugmentedMatrix();
    System.out.println(x: "-----------------------------------------");
    done = true;

}

else
{
    System.out.print(s: "Invalid choice, please enter 1 or 2: ");
}
```

```
        eqInput.close();
        file.close();
    }

    /**
     * This function prints the augmented matrix in a neat format.
     */
    private static void printAugmentedMatrix()
    {
        for (double[] linearEquation : augmentedMatrix)
        {
            for (double coefficient : linearEquation)
            {
                System.out.print("[" + coefficient +"] ");
            }
            System.out.println();
        }
        System.out.println();
    }
```

```java
/**
 * Uses Gaussian Elimination with Scaled Partial Pivoting to solve augmented matrices
 * holding up to 10 linear equations. Prints intermediate steps during solving process.
 */
private static void gaussianEliminationWithScaledPartialPivoting()
{
    /*
     * Find highest pivot value to prepare for swapping a row to work with partial pivoting.
     */
    int maxLength = (Math.min(augmentedMatrix.length, augmentedMatrix[0].length));
    for (int pivot = 0; pivot < maxLength; pivot++)
    {
        int maxCoefficient = pivot;

        for (int i = (pivot + 1); i < augmentedMatrix.length; i++)
        {
            if (Math.abs(augmentedMatrix[i][pivot]) > Math.abs(augmentedMatrix[maxCoefficient][pivot]))
            {
                maxCoefficient = i;
            }
        }

        /*
         * Swap array rows based on pivot values.
         */
        double[] swapArray = augmentedMatrix[pivot];
        augmentedMatrix[pivot] = augmentedMatrix[maxCoefficient];
        augmentedMatrix[maxCoefficient] = swapArray;
```

```java
            /*
             * Calculate the scaled value for each array and convert augmentedMatrix' values
             * to be consistent with the scaled value.
             */
            for (int i = (pivot + 1); i < augmentedMatrix.length; i++)
            {
                double scaledCoefficient = augmentedMatrix[i][pivot] / augmentedMatrix[pivot][pivot];

                for (int j = pivot; j < augmentedMatrix[0].length; j++)
                {
                    augmentedMatrix[i][j] = augmentedMatrix[i][j] - (scaledCoefficient * augmentedMatrix[pivot][j]);
                }

                System.out.println(x: "\n------------- INTERMEDIATE STEP -------------\n");
                printAugmentedMatrix();
                System.out.println(x: "-------------------------------------------\n");

            }
        }
    }
```

```java
    /**
     * Creates an array holding the solved variables and prints them.
     */
    private static void printAnswerArray()
    {
        /*
         * Calculate the value of the variables and place them in the answer array.
         */
        double[] answerArray = new double[augmentedMatrix[0].length];

        int minLength = (Math.min(augmentedMatrix[0].length - 1, augmentedMatrix.length - 1));

        for (int i = minLength; i >= 0; i--)
        {
            double sum = 0.0;

            for (int j = (i + 1); j < augmentedMatrix[0].length; j++)
            {
                sum += augmentedMatrix[i][j] * answerArray[j];
            }

            answerArray[i] = (augmentedMatrix[i][augmentedMatrix.length] - sum) / augmentedMatrix[i][i];
        }

        /*
         * Print the answer array with up to 10 variables.
         */
        char[] variables = { 'x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f', 'g' };

        System.out.println(x: "Solved Varibles: ");

        for (int i = 0; i < augmentedMatrix.length; i++)
        {
            System.out.print(variables[i] + " = " + answerArray[i] + "\n");
        }

        System.out.println();
    }
}
```

Test Run 1 (Command Line Input):

```
# of linear equations to solve: 3
Enter 1 to input values using CLI, enter 2 to input values using file input: 1
Enter 4 values for row 1:
2 3 0 8
Enter 4 values for row 2:
-1 2 -1 0
Enter 4 values for row 3:
3 0 2 9

------------- STARTING MATRIX -------------

[2.0] [3.0] [0.0] [8.0]
[-1.0] [2.0] [-1.0] [0.0]
[3.0] [0.0] [2.0] [9.0]

-------------------------------------------

------------- INTERMEDIATE STEP -------------

[3.0] [0.0] [2.0] [9.0]
[0.0] [2.0] [-0.33333333333333337] [3.0]
[2.0] [3.0] [0.0] [8.0]

-------------------------------------------


------------- INTERMEDIATE STEP -------------

[3.0] [0.0] [2.0] [9.0]
[0.0] [2.0] [-0.33333333333333337] [3.0]
[0.0] [3.0] [-1.3333333333333333] [2.0]

-------------------------------------------


------------- INTERMEDIATE STEP -------------

[3.0] [0.0] [2.0] [9.0]
[0.0] [3.0] [-1.3333333333333333] [2.0]
[0.0] [0.0] [0.5555555555555555] [1.6666666666666667]

-------------------------------------------

Solved Varibles:
x = 0.9999999999999997
y = 2.0
z = 3.0000000000000004
```

Test Run 2 (File Input):

```
# of linear equations to solve: 6
Enter 1 to input values using CLI, enter 2 to input values using file input: 2
Enter the full path of the file, along with the file extension [Use right click to paste file path]:
C:\Users\JRSiw\Desktop\Programming\School Workspace\CS3010\Project 1\test.txt

------------- STARTING MATRIX -------------

[2.0] [3.0] [0.0] [8.0] [5.0] [2.0] [4.0]
[-1.0] [2.0] [-1.0] [0.0] [2.0] [7.0] [1.0]
[3.0] [0.0] [2.0] [9.0] [9.0] [10.0] [-4.0]
[4.0] [11.0] [-3.0] [2.0] [9.0] [8.0] [2.0]
[0.0] [3.0] [21.0] [-10.0] [1.0] [6.0] [8.0]
[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]

-------------------------------------------

------------- INTERMEDIATE STEP -------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [2.1333333333333333] [-0.9333333333333333] [0.2] [2.066666666666667] [7.533333333333333] [1.2666666666666666]
[3.0] [0.0] [2.0] [9.0] [9.0] [10.0] [-4.0]
[4.0] [11.0] [-3.0] [2.0] [9.0] [8.0] [2.0]
[0.0] [3.0] [21.0] [-10.0] [1.0] [6.0] [8.0]
[2.0] [3.0] [0.0] [8.0] [5.0] [2.0] [4.0]

-------------------------------------------


------------- INTERMEDIATE STEP -------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [2.1333333333333333] [-0.9333333333333333] [0.2] [2.066666666666667] [7.533333333333333] [1.2666666666666666]
[0.0] [-0.4] [1.8] [8.4] [8.8] [8.4] [-4.8]
[4.0] [11.0] [-3.0] [2.0] [9.0] [8.0] [2.0]
[0.0] [3.0] [21.0] [-10.0] [1.0] [6.0] [8.0]
[2.0] [3.0] [0.0] [8.0] [5.0] [2.0] [4.0]

-------------------------------------------
```

```
------------ INTERMEDIATE STEP ------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [2.1333333333333333] [-0.9333333333333333] [0.2] [2.066666666666667] [7.533333333333333] [1.2666666666666666]
[0.0] [-0.4] [1.8] [8.4] [8.8] [8.4] [-4.8]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [3.0] [21.0] [-10.0] [1.0] [6.0] [8.0]
[2.0] [3.0] [0.0] [8.0] [5.0] [2.0] [4.0]

-------------------------------------------


------------ INTERMEDIATE STEP ------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [2.1333333333333333] [-0.9333333333333333] [0.2] [2.066666666666667] [7.533333333333333] [1.2666666666666666]
[0.0] [-0.4] [1.8] [8.4] [8.8] [8.4] [-4.8]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [3.0] [21.0] [-10.0] [1.0] [6.0] [8.0]
[2.0] [3.0] [0.0] [8.0] [5.0] [2.0] [4.0]

-------------------------------------------


------------ INTERMEDIATE STEP ------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [2.1333333333333333] [-0.9333333333333333] [0.2] [2.066666666666667] [7.533333333333333] [1.2666666666666666]
[0.0] [-0.4] [1.8] [8.4] [8.8] [8.4] [-4.8]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [3.0] [21.0] [-10.0] [1.0] [6.0] [8.0]
[0.0] [2.7333333333333334] [-0.13333333333333333] [7.6] [4.866666666666666] [0.9333333333333333] [3.466666666666667]

-------------------------------------------


------------ INTERMEDIATE STEP ------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [-5.551115123125783E-17] [1.6751592356687899] [8.445859872611466] [9.13375796178344] [8.624203821656051] [-4.764331210191083]
[0.0] [2.1333333333333333] [-0.9333333333333333] [0.2] [2.066666666666667] [7.533333333333333] [1.2666666666666666]
[0.0] [3.0] [21.0] [-10.0] [1.0] [6.0] [8.0]
[0.0] [2.7333333333333334] [-0.13333333333333333] [7.6] [4.866666666666666] [0.9333333333333333] [3.466666666666667]

-------------------------------------------
```

```
------------ INTERMEDIATE STEP ------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [-5.551115123125783E-17] [1.6751592356687899] [8.445859872611466] [9.13375796178344] [8.624203821656051] [-4.764331210191083]
[0.0] [0.0] [-0.26751592356687903] [-0.04458598726114646] [0.28662420382165665] [6.337579617834395] [1.0764331210191083]
[0.0] [3.0] [21.0] [-10.0] [1.0] [6.0] [8.0]
[0.0] [2.7333333333333334] [-0.13333333333333333] [7.6] [4.866666666666666] [0.9333333333333333] [3.466666666666667]

-----------------------------------------------


------------ INTERMEDIATE STEP ------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [-5.551115123125783E-17] [1.6751592356687899] [8.445859872611466] [9.13375796178344] [8.624203821656051] [-4.764331210191083]
[0.0] [0.0] [-0.26751592356687903] [-0.04458598726114646] [0.28662420382165665] [6.337579617834395] [1.0764331210191083]
[0.0] [0.0] [21.936305732484076] [-10.343949044585987] [-1.5031847133757958] [4.318471337579618] [7.732484076433121]
[0.0] [2.7333333333333334] [-0.13333333333333333] [7.6] [4.866666666666666] [0.9333333333333333] [3.466666666666667]

-----------------------------------------------


------------ INTERMEDIATE STEP ------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [-5.551115123125783E-17] [1.6751592356687899] [8.445859872611466] [9.13375796178344] [8.624203821656051] [-4.764331210191083]
[0.0] [0.0] [-0.26751592356687903] [-0.04458598726114646] [0.28662420382165665] [6.337579617834395] [1.0764331210191083]
[0.0] [0.0] [21.936305732484076] [-10.343949044585987] [-1.5031847133757958] [4.318471337579618] [7.732484076433121]
[0.0] [0.0] [0.7197452229299363] [7.286624203821655] [2.5859872611464967] [-0.5987261146496816] [3.2229299363057327]

-----------------------------------------------


------------ INTERMEDIATE STEP ------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [0.0] [21.936305732484076] [-10.343949044585987] [-1.5031847133757958] [4.318471337579618] [7.732484076433121]
[0.0] [0.0] [0.0] [0.0] [-0.17073170731707316] [0.2682926829268299] [6.390243902439025] [1.170731707317073]
[0.0] [-5.551115123125783E-17] [1.6751592356687899] [8.445859872611466] [9.13375796178344] [8.624203821656051] [-4.764331210191083]
[0.0] [0.0] [0.7197452229299363] [7.286624203821655] [2.5859872611464967] [-0.5987261146496816] [3.2229299363057327]

-----------------------------------------------
```

```
------------- INTERMEDIATE STEP -------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [0.0] [21.936305732484076] [-10.343949044585987] [-1.5031847133757958] [4.318471337579618] [7.732484076433121]
[0.0] [0.0] [0.0] [-0.17073170731707316] [0.2682926829268299] [6.390243902439025] [1.170731707317073]
[0.0] [-5.551115123125783E-17] [-2.220446049250313E-16] [9.235772357723578] [9.248548199767711] [8.294425087108014] [-5.354819976771196]
[0.0] [0.0] [0.7197452229299363] [7.286624203821655] [2.5859872611464967] [-0.5987261146496816] [3.2229299363057327]

-----------------------------------------------


------------- INTERMEDIATE STEP -------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [0.0] [21.936305732484076] [-10.343949044585987] [-1.5031847133757958] [4.318471337579618] [7.732484076433121]
[0.0] [0.0] [0.0] [-0.17073170731707316] [0.2682926829268299] [6.390243902439025] [1.170731707317073]
[0.0] [-5.551115123125783E-17] [-2.220446049250313E-16] [9.235772357723578] [9.248548199767711] [8.294425087108014] [-5.354819976771196]
[0.0] [0.0] [-1.1102230246251565E-16] [7.626016260162601] [2.635307781649245] [-0.7404181184668991] [2.9692218350754938]

-----------------------------------------------


------------- INTERMEDIATE STEP -------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [0.0] [21.936305732484076] [-10.343949044585987] [-1.5031847133757958] [4.318471337579618] [7.732484076433121]
[0.0] [-5.551115123125783E-17] [-2.220446049250313E-16] [9.235772357723578] [9.248548199767711] [8.294425087108014] [-5.354819976771196]
[0.0] [0.0] [0.0] [0.0] [0.4392605633802823] [6.543573943661972] [1.0717429577464788]
[0.0] [0.0] [-1.1102230246251565E-16] [7.626016260162601] [2.635307781649245] [-0.7404181184668991] [2.9692218350754938]

-----------------------------------------------


------------- INTERMEDIATE STEP -------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [0.0] [21.936305732484076] [-10.343949044585987] [-1.5031847133757958] [4.318471337579618] [7.732484076433121]
[0.0] [-5.551115123125783E-17] [-2.220446049250313E-16] [9.235772357723578] [9.248548199767711] [8.294425087108014] [-5.354819976771196]
[0.0] [0.0] [0.0] [0.0] [0.4392605633802823] [6.543573943661972] [1.0717429577464788]
[0.0] [0.0] [-1.1102230246251565E-16] [0.0] [-5.001257545271628] [-7.58915995975855] [7.390719315895371]

-----------------------------------------------
```

```
------------- INTERMEDIATE STEP -------------

[15.0] [2.0] [1.0] [3.0] [1.0] [8.0] [4.0]
[0.0] [10.466666666666667] [-3.2666666666666666] [1.2] [8.733333333333333] [5.866666666666667] [0.9333333333333333]
[0.0] [0.0] [21.936305732484076] [-10.343949044585987] [-1.5031847133757958] [4.318471337579618] [7.732484076433121]
[0.0] [-5.551115123125783E-17] [-2.220446049250313E-16] [9.235772357723578] [9.248548199767711] [8.294425087108014] [-5.354819976771196]
[0.0] [0.0] [-1.1102230246251565E-16] [0.0] [-5.001257545271628] [-7.58915995975855] [7.390719315895371]
[0.0] [0.0] [0.0] [0.0] [0.0] [5.877017852652752] [1.720870002514459]

-----------------------------------------------

Solved Varibles:
x = -0.2379764599172547 2
y = 1.614942946746018
z = 0.6733524723076726
a = 1.0820009498179517
b = -1.9221020925944154
c = 0.29281347201245905
```