Programming Assignment 2 Grading Rubric

The rubric below will be applied to each of the programs your submit for Programming Assignment 2. Each program will be assigned a score between 0 and 100. The overall score for programming assignment 2 will be the average of the three program scores.

Criteria	Notes	MaxPts	Earned Pts
Author Comments	no author comments	6	6
Informational Comment	Need to comment classes	6	4
Program Formatting	formatting ok	6	6
Program Meets Assignment Input and Output Specification	Accepts appropriate inputs and produces required output	12	12

Program Produce Correct Answers	Should not report true with neg values for positivemarkove. . In fixedpoint the goal is to not use floating point.	40	35
Program Compilation	Compiles ok.	30	30
Total Points per program		100	93

```
package progassgn2;

//Author: Raja Pragnesh Reddy Nandyala
//Assignment: Programming Assignment 2
//Instructor: Prof Dave Pitts
//Date: <due 2/21/2016

import java.util.Scanner;

public class PositiveMarkov {
        static int j = 0;
        static float total = 0;
        static float[] result = new float[7];

        public static void main(String[] args) {
                int noOfRows = 0;
                Scanner input = new Scanner(System.in);

                do {
                        System.out.println("Enter matrix input between 2 & 7");
                        noOfRows = input.nextInt();
                } while (!(noOfRows >= 2 && noOfRows < 8));

                System.out.println("You have choosen " + noOfRows + "*" + noOfRows + "
matrix. Begin entering numbers.");
                float[][] matrix = new float[noOfRows][noOfRows];
                // reading a matrix one at a time
                for (int i = 0; i < matrix.length; i++) {
                        for (int j = 0; j < matrix[i].length; j++) {
                                System.out.print("Enter next number:" + " ");

                                matrix[i][j] = input.nextFloat();
                        }
                }

                // out of a  read matrix
                for (int i = 0; i < matrix.length; i++) {

                        for (int j = 0; j < matrix[i].length; j++) {
                                if (i == 0 && j == 0) {
                                        System.out.println("The matrix entered was ");
                                }
                                System.out.print(matrix[i][j] + " ");
                        }
                        System.out.println();
                }
                //
                boolean returnValue = Validate(matrix);
                if (returnValue == true)
                        System.out.println("Positive Markov");
                else
                        System.out.println("Negetive Markov");
                input.close();
        }
// calling a method to check condition
        static boolean Validate(float[][] matrix) {
// loop to add row values
                for (int row = 0; row < matrix.length; row++) {
                        float total = 0;
                        for (int column = 0; column < matrix.length; column++)
                                total = total + matrix[row][column];
                        // when sum is not equal to 1 then return false
                        if (total != 1.0)
```

can't have negative values

```
                                return false;
                }
                return true;

        }

}
```

```
package progassgn2;

//Author: Raja Pragnesh Reddy Nandyala
//Assignment: Programming Assignment 2
//Instructor: Prof Dave Pitts
//Date: <due 2/21/2016
import java.util.Scanner;

public class SelectColumn {
        static int select = 0;

        public static void main(String[] args) {

                Scanner input = new Scanner(System.in);
                System.out.print("Enter number of rows: ");
                int rows = input.nextInt();
                System.out.print("Enter number of columns: ");
                int columns = input.nextInt();
                // creating two dimensional array from give rows and columns
                int[][] twoDimArray = new int[rows][columns];
                // reading one after other
                for (int row = 0; row < twoDimArray.length; row++) {
                        for (int column = 0; column < twoDimArray[row].length; column++
) {
                                System.out.print("Enter values in a matrix row " + row
+ " " + "Enter column " + column + " : ");
                                twoDimArray[row][column] = input.nextInt();
                        }
                }

                System.out.println("Selectort column? ");
                int a = input.nextInt();

                int[] returnColumns = SelectColumns(twoDimArray, a);
// validating entered or selected column
                if (returnColumns == null) {
                        System.out.println("The column selector was invalid");
                } else {
                        System.out.println("The select column was");
                        for (int i = 0; i < returnColumns.length; i++)
                                System.out.println(returnColumns[i]);

                }
                input.close();
        }

        static int[] SelectColumns(int[][] tempArray, int columnNumber) {
                int[] column = new int[tempArray.length];
                int index = 0;
                System.out.println("The array was");
                // getting out the given input array
                for (int row = 0; row < tempArray.length; row++) {
                        for (int columns = 0; columns < tempArray[row].length; columns+
+) {
                                System.out.print(tempArray[row][columns] + " ");

                                if (columns == columnNumber) {
                                        column[index++] = tempArray[row][columns];
                                }
                        }
                        System.out.println();
                }
```

```
                if (index == 0)
                        return null;
                else
                        return column;
        }
}
```

```
package progassgn2;

import java.math.BigDecimal;
import java.text.DecimalFormat;

public class FixedPointDecimal {
	public static void main(String[] args) {
		FixedPoint f1 = new FixedPoint(2, 44, 30);
		System.out.println("value is: " + f1.value);
		FixedPoint f2 = new FixedPoint(2, "71.56");
		System.out.println("value is: " + f2.value);// printing second value of

		// a constructor
		System.out.println("value is: " + f1.add(f2));// adding f2 to f1

	}
}

class FixedPoint {
	int d;
	int w;
	int f;
	String floatString;
	double value;

	// using constructor converting into a required float value
	public FixedPoint(int d, int w, int f) {
		this.d = d;
		this.w = w;
		String m;
		// using decimal format to set max & minimum decimal points
		DecimalFormat df = new DecimalFormat();
		df.setMaximumFractionDigits(d);
		df.setMinimumFractionDigits(d);
		m = (String) (w + "." + f);
		// converting and formating the string value to big decimal
		BigDecimal tempValue = new BigDecimal(m);

		this.value = Double.parseDouble(df.format(tempValue));
	}

	public FixedPoint(int d, String f) {
		this.d = d;
		DecimalFormat df = new DecimalFormat();
		df.setMaximumFractionDigits(d);
		df.setMinimumFractionDigits(d);
		BigDecimal tempValue = new BigDecimal(f);
		this.value = Double.parseDouble(df.format(tempValue));

	}

	public FixedPoint add(FixedPoint f) {
		// checking two constructor decimal values
		if (this.d == f.d) {
			double floatTotal = this.value + f.value;
			FixedPoint sumF = new FixedPoint(this.d, String.valueOf(floatTo
tal));
			return sumF;

		} else {

			return null;
```

goal is not to use Floating point

```
                }

        }

        public String toString() {

                return String.valueOf(this.value);
        }

}
```