Problem Statement

Write a Java program to print the following pattern (where number of lines will be given by the user): -

***0**
00**
*****000**
0000****
[The above output is for 4 lines.]

Algorithm

Algorithm Pattern1
Input: No of rows given by user.
Output: Display pattern according to input row.
Step 1: Start
Step 2: num ← no. of row input from user
Step 3: If (num>0) then
    Step 3.1: For (i=1 to num) do
        Step 3.1.1: If (i%2 ≠ 0) then
            Step 3.1.1.1: For (j=0 to j<i) do
                Step 3.1.1.1.1: Print ("*")
            Step 3.1.1.2: End For
            Step 3.1.1.3: For (j=0 to j<i) do
                Step 3.1.1.3.1: Print ("0")
            Step 3.1.1.4: End For
            Step 3.1.1.5: Print (NewLine)
        Step 3.1.2: Else
            Step 3.1.2.1: For (j=0 to j<i) do
                Step 3.1.2.1.1: Print ("0")
            Step 3.1.2.2: End For
            Step 3.1.2.3: For (j=0 to j<i) do
                Step 3.1.2.3.1: Print ("*")
            Step 3.1.2.4: End For
            Step 3.1.2.5: Print (NewLine)
        Step 3.1.3: End If
    Step 3.2: End For
Step 4: Else
    Step 4.1: Print ("Enter Positive Number")
Step 5: End If
Step 6: Stop

Source Code

```
import java.util.Scanner;
class Pattern1
{
	public static void main(String[] args)
	{
		Scanner input = new Scanner(System.in);		// Create a Scanner object to read input from the user
		System.out.print("Enter the number of lines: ");	// Prompt the user to enter the number of lines
		int num = input.nextInt();			// Read the user's input as an integer
		if(num>0)					// Check if the input is positive
		{
			for (int i = 1; i <= num; i++)		// Loop through each line
			{
				if (i%2 != 0)			// Check if the current line number is odd
				{
					for (int j = 0; j < i; j++)	// Print '*' characters for the first half of the line
					{
						System.out.print("*");
					}
					for (int j = 0; j < i; j++)	// Print '0' characters for the second half of the line
					{
						System.out.print("0");
					}
					System.out.println();		// Move to the next line
				}
				else					// If the current line number is even, reverse the pattern
				{
					for (int j = 0; j < i; j++)	// Print '0' characters for the first half of the line
					{
						System.out.print("0");
					}
					for (int j = 0; j < i; j++)	// Print '*' characters for the second half of the line
					{
						System.out.print("*");
					}
					System.out.println();
				}
			}
		}
		else						// If the input is not positive, display an error message
		{
			System.out.println("!! Enter a positive number !!");
		}
	}
}
```

Result

C:\JAVA\College>*javac A14.java*

C:\JAVA\College>*java Pattern1*
Enter the number of lines: 4
**0*
*00***
****000*
*0000*****

C:\JAVA\College>*java Pattern1*
Enter the number of lines: 8
**0*
*00***
****000*
*0000*****
*****00000*
*000000*******
*******0000000*
*00000000*********

C:\JAVA\College>*java Pattern1*
Enter the number of lines: 0
!! Enter a positive number !!

C:\JAVA\College>*java Pattern1*
Enter the number of lines: -5
!! Enter a positive number !!

Discussion

- The program takes user input to determine the number of lines to be printed in a pattern.
- It first checks whether the input number is positive (greater than 0) to ensure that it's a valid input.
- If the input is positive, it enters a loop that iterates from 1 to the specified number of lines (inclusive).
- Inside the loop, it further checks if the current line number is odd (i%2 != 0). If it's odd, it prints '*' characters for the first half of the line and '0' characters for the second half, creating a pattern.
- If the current line number is even, it reverses the pattern by printing '0' characters for the first half and '*' characters for the second half.
- After completing each line, it moves to the next line by using System.out.println() to add a newline character.
- If the user enters a non-positive number, the program displays an error message.

Overall, the program generates a pattern of alternating *'*' and '0'* characters in a *triangular form*, with *odd-numbered lines starting with '**'* and *even-numbered lines starting with '0'*.

Problem Statement

Write a Java program to calculate matrix addition, subtraction and multiplication using switch case. The two matrices and the choice of operation between these two matrices will be given by the user.

Algorithm

Algorithm Calculator

Input: Rows and columns with matrix from user.

Output: Display resultant matrix or warning according to user input.

Step 1: Start

Step 2: r1, c1, r2, c2 ← input rows and columns for Matrix A and Matrix B

Step 3: If (r1 < 1 or c1 < 1 or r2 < 1 or c2 < 1) then

  Step 3.1: Print ("Numbers of rows and column will be positive")

  Step 3.2: Return

Step 4: End If

Step 5: Input Matrix A and Matrix B from user

Step 6: Chose option Choice for (Addition/Subtraction/Multiplication)

Step 7: If (Choice = 1) then *//addition*

  Step 7.1: If (r1 ≠ r2 or c1 ≠ c2) then

    Step 7.1.1: Print ("Matrix addition is not possible. Matrices A and B must have the same dimensions.");

    Step 7.1.2: Return

  Step 7.2: End If

  Step 7.3: For (i = 0 to i < r1) do

    Step 7.3.1: For (j = 0 to j < c2) do

      Step 7.3.1.1: Result[i][j] ← matrixA[i][j] + matrixB[i][j]

    Step 7.3.2: End For

  Step 7.4: End For

  Step 7.5: Break

Step 8: Else If (Choice = 2) then *//subtraction*

  Step 8.1: If (r1 ≠ r2 or c1 ≠ c2) then

    Step 8.1.1: Print ("Matrix subtraction is not possible. Matrices A and B must have the same dimensions.");

    Step 8.1.2: Return

  Step 8.2: End If

  Step 8.3: For (i = 0 to i < r1) do

    Step 8.3.1: For (j = 0 to j < c2) do

      Step 8.3.1.1: Result[i][j] ← matrixA[i][j] - matrixB[i][j]

    Step 8.3.2: End For

  Step 8.4: End For

  Step 8.5: Break

Step 9: Else if (Choice = 3) then *//multiplication*

  Step 9.1: If (c1 ≠ r2)

    Step 9.1.1: Print ("Matrix multiplication is not possible. Number of columns in A must be equal to the number of rows in B.");

    Step 9.1.2: Return

  Step 9.2: End If

  Step 9.3: For (i = 0 to i < r1) do

    Step 9.3.1: For (j = 0 to j < c2) do

      Step 9.3.1.1: Result[i][j] ← 0

      Step 9.3.1.2: For (k = 0 to k < c1) do

        Step 9.3.1.2.1: Result[i][j] += matrixA[i][k] * matrixB[k][j]

      Step 9.3.1.3: End For

    Step 9.3.2: End For

  Step 9.4: End For

  Step 9.5: Break

Step 10: Else *//default*

  Step 10.1: Print ("Invalid")

  Step 10.2: Return

Step 11: End If

Step 12: Stop

Source Code

```
import java.util.Scanner;
class Calculator
{
	public static void main(String[] args)
	{
		Scanner input = new Scanner(System.in);

		// Input dimensions of the matrices
		System.out.print("Enter the number of rows for matrix A: ");
		int m1 = input.nextInt();
		System.out.print("Enter the number of columns for matrix A: ");
		int n1 = input.nextInt();

		System.out.print("Enter the number of rows for matrix B: ");
		int m2 = input.nextInt();
		System.out.print("Enter the number of columns for matrix B: ");
		int n2 = input.nextInt();

		if (m1 < 1 || n1 < 1 || m2 < 1 || n2 < 1)
		{
			System.out.println("Please enter positive values for the number of rows and columns.");
			return;   // Exit the program
		}

		// Input matrices A and B
		int[][] matrixA = new int[m1][n1];
		int[][] matrixB = new int[m2][n2];

		System.out.println("Enter elements for matrix A:");
		for (int i = 0; i < m1; i++)
		{
			for (int j = 0; j < n1; j++)
			{
				matrixA[i][j] = input.nextInt();
			}
		}

		System.out.println("Enter elements for matrix B:");
		for (int i = 0; i < m2; i++)
		{
			for (int j = 0; j < n2; j++)
			{
				matrixB[i][j] = input.nextInt();
			}
		}

		// Choose the operation
		System.out.println("Choose operation:");
		System.out.println("1. Addition");
		System.out.println("2. Subtraction");
		System.out.println("3. Multiplication");
		System.out.print("Enter your choice (1/2/3): ");
		int choice = input.nextInt();

		int[][] result = new int[m1][n2];
```

```
// Perform the chosen operation
switch (choice)
{
        case 1:
                // Addition
                if (m1 != m2 || n1 != n2)
                {
                        System.out.println("Matrix addition is not possible. Matrices A and B must
                                                have the same dimensions.");
                        return;
                }
                for (int i = 0; i < m1; i++)
                {
                        for (int j = 0; j < n2; j++)
                        {
                                result[i][j] = matrixA[i][j] + matrixB[i][j];
                        }
                }
                break;
        case 2:
                // Subtraction
                if (m1 != m2 || n1 != n2)
                {
                        System.out.println("Matrix subtraction is not possible. Matrices A and B must
                                                have the same dimensions.");
                        return;
                }
                for (int i = 0; i < m1; i++)
                {
                        for (int j = 0; j < n2; j++)
                        {
                                result[i][j] = matrixA[i][j] - matrixB[i][j];
                        }
                }
                break;
        case 3:
                // Multiplication
                // Check if matrices can be operated on
                if (n1 != m2)
                {
                        System.out.println("Matrix multiplication is not possible. Number of columns
                                                in A must be equal to the number of rows in B.");
                        return;
                }

                for (int i = 0; i < m1; i++)
                {
                        for (int j = 0; j < n2; j++)
                        {
                                result[i][j] = 0;
                                for (int k = 0; k < n1; k++)
                                {
                                        result[i][j] += matrixA[i][k] * matrixB[k][j];
                                }
                        }
                }
                break;
```

```
                default:
                        System.out.println("Invalid choice");
                        return;
                }

                // Print the result matrix
                System.out.println("Result Matrix:");
                for (int i = 0; i < m1; i++)
                {
                        for (int j = 0; j < n2; j++)
                        {
                                System.out.print(result[i][j] + " ");
                        }
                        System.out.println();
                }
        }
}
```

Result

C:\JAVA\College>*javac A15.java*

C:\JAVA\College>*java Calculator*
Enter the number of rows for matrix A: 3
Enter the number of columns for matrix A: 3
Enter the number of rows for matrix B: 3
Enter the number of columns for matrix B: 3
Enter elements for matrix A:
2 2 1
1 5 0
0 0 1
Enter elements for matrix B:
5 7 1
0 3 0
1 0 8
Choose operation:
1. Addition
2. Subtraction
3. Multiplication
Enter your choice (1/2/3): 1
Result Matrix:
7 9 2
1 8 0
1 0 9

C:\JAVA\College>*java Calculator*
Enter the number of rows for matrix A: 3
Enter the number of columns for matrix A: 3
Enter the number of rows for matrix B: 3
Enter the number of columns for matrix B: 3
Enter elements for matrix A:
2 2 1
1 5 0
0 0 1

Enter elements for matrix B:
5 7 1
0 3 0
1 0 8
Choose operation:
1. Addition
2. Subtraction
3. Multiplication
Enter your choice (1/2/3): 2
Result Matrix:
-3 -5 0
1 2 0
-1 0 -7

C:\JAVA\College>*java Calculator*
Enter the number of rows for matrix A: 3
Enter the number of columns for matrix A: 4
Enter the number of rows for matrix B: 3
Enter the number of columns for matrix B: 3
Matrix multiplication is not possible. Number of columns in A must be equal to the number of rows in B.

C:\JAVA\College>*java Calculator*
Enter the number of rows for matrix A: 3
Enter the number of columns for matrix A: 3
Enter the number of rows for matrix B: 3
Enter the number of columns for matrix B: 3
Enter elements for matrix A:
1 2 3
3 4 -2
3 2 1
Enter elements for matrix B:
-1 1 1
3 4 -2
3 2 1
Choose operation:
1. Addition
2. Subtraction
3. Multiplication
Enter your choice (1/2/3): 3
Result Matrix:
14 15 0
3 15 -7
6 13 0

C:\JAVA\College>*java Calculator*
Enter the number of rows for matrix A: 3
Enter the number of columns for matrix A: -3
Enter the number of rows for matrix B: 3
Enter the number of columns for matrix B: 3
Please enter positive values for the number of rows and columns.

C:\JAVA\College>*java Calculator*
Enter the number of rows for matrix A: 3
Enter the number of columns for matrix A: 4
Enter the number of rows for matrix B: 4
Enter the number of columns for matrix B: 3
Enter elements for matrix A:
7 5 3 2
1 5 9 3
1 4 5 6
Enter elements for matrix B:
4 5 6
7 8 9
1 2 3
1 4 7
Choose operation:
1. Addition
2. Subtraction
3. Multiplication
Enter your choice (1/2/3): 1
Matrix addition is not possible. Matrices A and B must have the same dimensions.

C:\JAVA\College>*java Calculator*
Enter the number of rows for matrix A: 2
Enter the number of columns for matrix A: 2
Enter the number of rows for matrix B: 2
Enter the number of columns for matrix B: 2
Enter elements for matrix A:
1 2
3 4
Enter elements for matrix B:
7 8
6 5
Choose operation:
1. Addition
2. Subtraction
3. Multiplication
Enter your choice (1/2/3): 4
Invalid choice

C:\JAVA\College>*java Calculator*
Enter the number of rows for matrix A: 2
Enter the number of columns for matrix A: 2
Enter the number of rows for matrix B: 2
Enter the number of columns for matrix B: 2
Enter elements for matrix A:
1 2 5
3 4 4
Enter elements for matrix B:
7 8
6 5
Choose operation:
1. Addition
2. Subtraction
3. Multiplication
Enter your choice (1/2/3): 2
Matrix subtraction is not possible. Matrices A and B must have the same dimensions.

Discussion

- ***User Input for Matrix Dimensions:*** The program begins by prompting the user to input dimensions (number of rows and columns) for two matrices, A and B. It also performs checks to ensure that these dimensions are positive, displaying a warning message and exiting if they are not.

- ***Input of Matrices A and B:*** After obtaining the dimensions, the program creates two 2D arrays, *matrixA* and *matrixB*, to store the elements of the matrices. It then uses nested loops to populate these matrices with user-provided values.

- ***Operation Selection:*** The program allows the user to choose from three matrix operations: addition, subtraction, and multiplication. It displays options and prompts the user for their choice using a *switch* statement.

- ***Matrix Operations:*** Depending on the user's choice, the program performs the selected matrix operation. For addition and subtraction, it checks if the matrices A and B have the same dimensions before performing the operation. For multiplication, it checks if the number of columns in matrix A is equal to the number of rows in matrix B, as required for matrix multiplication. The results of the operations are stored in the *result* matrix.

- ***Result Display:*** Finally, the program prints the resulting matrix based on the chosen operation. It displays the elements of the result matrix in the standard matrix format.

- ***Error Handling:*** The program includes error handling to notify the user of invalid input or operations that cannot be performed due to incompatible matrix dimensions.

Problem Statement

Write a program in java that accepts a 2D matrix (m X n) and prints the matrix with row minimum and column maximum values in the following format: -

Example: - Input:

4	**3**	**5**
1	**0**	**7**
8	**4**	**6**

Output:

4	**3**	**5**	*3*
1	**0**	**7**	*0*
8	**4**	**6**	*4*
8	*4*	*7*	

Algorithm

Algorithm RMCM

Input: No. of row and column and the matrix.

Output: Display the resultant matrix with row minimum and column maximum.

Step 1: Start

Step 2: m ← no. of row input, n ← no. of column input

Step 3: If (m<1 or n<1) then

- **Step 3.1:** Print ("Please enter positive values for the number of rows and columns.")
- **Step 3.2:** Return

Step 4: End If

Step 5: For (i=0 to i<m) do

- **Step 5.1:** min ← matrix[i][0]
- **Step 5.2:** For (j=1 to j<n) do
  - **Step 5.2.1:** If (matrix[i][j] < min) then
    - **Step 5.2.1.1:** min ← matrix[i][j]
  - **Step 5.2.2:** End If
- **Step 5.3:** End For
- **Step 5.4:** rowMin[i] ← min

Step 6: End For

Step 7: For (i=0 to i<n) do

- **Step 7.1:** max ← matrix[0][i]
- **Step 7.2:** For (j=1 to j<m) do
  - **Step 7.2.1:** If (matrix[i][j] < max) then
    - **Step 7.2.1.1:** max ← matrix[i][j]
  - **Step 7.2.2:** End If
- **Step 7.3:** End For
- **Step 7.4:** colMax[j] ← max

Step 8: End For

Step 9: For (i=0 to i<m) do

- **Step 9.1:** For (j=0 to j<n) do
  - **Step 9.1.1:** Print (matrix[i][j])
- **Step 9.2:** End For
- **Step 9.3:** Print (rowMin[i])
- **Step 9.4:** Print (NewLine)

Step 10: End For

Step 11: For (j=0 to j<n) do

- **Step 11.1**: Print (colMax[j])

Step 12: End For

Step 13: Stop

Source Code

```
import java.util.Scanner;
class RMCM
{
	public static void main(String[] args)
	{
		Scanner input = new Scanner(System.in);

		// Input the dimensions of the matrix
		System.out.print("Enter the number of rows (m): ");
		int m = input.nextInt();
		System.out.print("Enter the number of columns (n): ");
		int n = input.nextInt();

		if (m < 1 || n < 1)
		{
			System.out.println("Please enter positive values for the number of rows and columns.");
			return;			// Exit the program
		}

		// Input the matrix
		int[][] matrix = new int[m][n];
		System.out.println("Enter the elements of the matrix:");
		for (int i = 0; i < m; i++)
		{
			for (int j = 0; j < n; j++)
			{
				matrix[i][j] = input.nextInt();
			}
		}

		// Find row minimums and column maximums
		int[] rowMin = new int[m];
		int[] colMax = new int[n];

		for (int i = 0; i < m; i++)
		{
			int min = matrix[i][0];
			for (int j = 1; j < n; j++)
			{
				if (matrix[i][j] < min)
				{
					min = matrix[i][j];
				}
			}
			rowMin[i] = min;
		}

		for (int j = 0; j < n; j++)
		{
			int max = matrix[0][j];
			for (int i = 1; i < m; i++)
			{
				if (matrix[i][j] > max)
				{
					max = matrix[i][j];
```

```
                                        }
                                }
                                colMax[j] = max;
                        }
                        System.out.print("\n");

                        // Print the modified matrix
                        System.out.println("Modified Matrix:");
                        for (int i = 0; i < m; i++)
                        {
                                for (int j = 0; j < n; j++)
                                {
                                        System.out.print(matrix[i][j] + "\t");
                                }
                                System.out.print(rowMin[i]);
                                System.out.println();
                        }

                        for (int j = 0; j < n; j++)
                        {
                                System.out.print(colMax[j] + "\t");
                        }
                }
}
```

Result

C:\JAVA\College>*javac A16.java*

C:\JAVA\College>*java RMCM*
Enter the number of rows (m): 3
Enter the number of columns (n): 3
Enter the elements of the matrix:
4 3 5
1 0 7
8 4 6

Modified Matrix:
4 3 5 3
1 0 7 0
8 4 6 4
8 4 7

C:\JAVA\College>*java RMCM*
Enter the number of rows (m): 3
Enter the number of columns (n): 4
Enter the elements of the matrix:
5 6 7 -8
6 0 -4 6
7 -10 50 8

Modified Matrix:
5 6 7 -8 -8
6 0 -4 6 -4
7 -10 50 8 -10
7 6 50 8

C:\JAVA\College>*java RMCM*
Enter the number of rows (m): 3
Enter the number of columns (n): 3
Enter the elements of the matrix:
-4 -6 -7
-8 -2 -4
0 -5 -1

Modified Matrix:
-4 -6 -7 -7
-8 -2 -4 -8
0 -5 -1 -5
0 -2 -1

C:\JAVA\College>*java RMCM*
Enter the number of rows (m): -3
Enter the number of columns (n): 3
Please enter positive values for the number of rows and columns.

Discussion

- ***User Input:*** The program starts by taking user input for the number of rows (m) and columns (n) of a matrix using the Scanner class. It also includes a check to ensure that the entered dimensions are positive, providing a warning message if not.
- ***Matrix Input:*** After obtaining the matrix dimensions, the program creates a 2D array called matrix to store the elements of the matrix. It then uses nested loops to populate the matrix with user-provided values.
- ***Row Minimums and Column Maximums:*** The program calculates the minimum value for each row (*rowMin*) and the maximum value for each column (*colMax*) in the matrix. It uses nested loops to iterate through the elements of the matrix and updates these arrays accordingly.
- ***Printing the Modified Matrix:*** The program prints the modified matrix, including the original matrix values and the row minimums. Each row is followed by its respective row minimum value. The column maximums are printed in a separate line.
- ***Program Structure:*** This program showcases a structured approach to matrix manipulation and demonstrates the use of loops and arrays to perform calculations on a matrix. It also provides user-friendly input prompts and checks for negative or zero matrix dimensions.
- ***Overall Purpose:*** The program's primary purpose is to take user input for a matrix, find the minimum value for each row and the maximum value for each column, and then display the modified matrix with these additional values. It's a practical example of working with matrices in Java and can be a useful tool for basic matrix analysis tasks.

Problem Statement

Write a program in java that accepts a 2D matrix (m X n) and prints the matrix with row minimum, column maximum and the total (sum) of all elements of the matrix [in the (m, n) position of resultant matrix] in the following format: -

Example: - Input:

4	**3**	**5**
1	**0**	**7**
8	**4**	**6**

Output:

4	**3**	**5**	*3*
1	**0**	**7**	*0*
8	**4**	**6**	*4*
8	*4*	*7*	*38*

Algorithm

Algorithm RMCM2

Input: No. of row and column and the matrix.

Output: Display the resultant matrix with row minimum and column maximum.

Step 1: Start

Step 2: m ← no. of row input, n ← no. of column input, sum ← 0

Step 3: If (m<1 or n<1) then

- **Step 3.1:** Print ("Please enter positive values for the number of rows and columns.")
- **Step 3.2:** Return

Step 4: End If

Step 5: For (i=0 to i<m) do

- **Step 5.1:** min ← matrix[i][0]
- **Step 5.2:** For (j=1 to j<n) do
  - **Step 5.2.1:** If (matrix[i][j] < min) then
    - **Step 5.2.1.1:** min ← matrix[i][j]
  - **Step 5.2.2:** End If
- **Step 5.3:** End For
- **Step 5.4:** rowMin[i] ← min

Step 6: End For

Step 7: For (i=0 to i<n) do

- **Step 7.1:** max ← matrix[0][i]
- **Step 7.2:** For (j=1 to j<m) do
  - **Step 7.2.1:** If (matrix[i][j] < max) then
    - **Step 7.2.1.1:** max ← matrix[i][j]
  - **Step 7.2.2:** End If
- **Step 7.3:** End For
- **Step 7.4:** colMax[j] ← max

Step 8: End For

Step 9: For (i=0 to i<m) do

- **Step 9.1:** For (j=0 to j<n) do
  - **Step 9.1.1:** sum += matrix[i][j]
- **Step 9.2:** End For

Step 10: End For

Step 11: For (i=0 to i<m) do

- **Step 11.1:** For (j=0 to j<n) do
  - **Step 11.1.1:** Print (matrix[i][j])
- **Step 11.2:** End For
- **Step 11.3:** Print (rowMin[i])
- **Step 11.4:** Print (NewLine)

Step 12: End For

Step 13: For (j=0 to j<n) do

- **Step 13.1**: Print (colMax[j])

Step 14: End For

Step 15: Print (sum)

Step 13: Stop

Source Code

```
import java.util.Scanner;
class RMCM2
{
	public static void main(String[] args)
	{
		Scanner input = new Scanner(System.in);

		// Input the dimensions of the matrix
		System.out.print("Enter the number of rows (m): ");
		int m = input.nextInt();
		System.out.print("Enter the number of columns (n): ");
		int n = input.nextInt();

		if (m < 1 || n < 1)
		{
			System.out.println("Please enter positive values for the number of rows and columns.");
			return;			// Exit the program
		}

		// Input the matrix
		int[][] matrix = new int[m][n];
		System.out.println("Enter the elements of the matrix:");
		for (int i = 0; i < m; i++)
		{
			for (int j = 0; j < n; j++)
			{
				matrix[i][j] = input.nextInt();
			}
		}

		// Find row minimums and column maximums
		int[] rowMin = new int[m];
		int[] colMax = new int[n];

		for (int i = 0; i < m; i++)
		{
			int min = matrix[i][0];
			for (int j = 1; j < n; j++)
			{
				if (matrix[i][j] < min)
				{
					min = matrix[i][j];
				}
			}
			rowMin[i] = min;
		}

		for (int j = 0; j < n; j++)
		{
			int max = matrix[0][j];
			for (int i = 1; i < m; i++)
			{
				if (matrix[i][j] > max)
				{
					max = matrix[i][j];
```

```
                        }
                    }
                    colMax[j] = max;
                }

                // Calculate the total sum of all elements
                int totalSum = 0;
                for (int i = 0; i < m; i++)
                {
                    for (int j = 0; j < n; j++)
                    {
                        totalSum += matrix[i][j];
                    }
                }

                // Print the modified matrix with row minima, column maxima, and total sum
                System.out.println("Modified Matrix:");
                for (int i = 0; i < m; i++)
                {
                    for (int j = 0; j < n; j++)
                    {
                        System.out.print(matrix[i][j] + "\t");
                    }
                    System.out.print(rowMin[i] + "\t");
                    System.out.println();
                }

                for (int j = 0; j < n; j++)
                {
                    System.out.print(colMax[j] + "\t");
                }

                // Print the total sum
                System.out.println(totalSum);
            }
}
```

Result

C:\JAVA\College>*javac A17.java*

C:\JAVA\College>*java RMCM2*
Enter the number of rows (m): 3
Enter the number of columns (n): 3
Enter the elements of the matrix:
4 3 5
1 0 7
8 4 6
Modified Matrix:
4 3 5 3
1 0 7 0
8 4 6 4
8 4 7 38

C:\JAVA\College>*java RMCM2*
Enter the number of rows (m): 3

Enter the number of columns (n): 4
Enter the elements of the matrix:
5 6 7 -8
6 0 -4 6
7 -10 50 8
Modified Matrix:
5 6 7 -8 -8
6 0 -4 6 -4
7 -10 50 8 -10
7 6 50 8 73

C:\JAVA\College>*java RMCM2*
Enter the number of rows (m): 3
Enter the number of columns (n): 3
Enter the elements of the matrix:
-4 -6 -7
-8 -2 -4
0 -5 -1
Modified Matrix:
-4 -6 -7 -7
-8 -2 -4 -8
0 -5 -1 -5
0 -2 -1 -37

C:\JAVA\College>*java RMCM2*
Enter the number of rows (m): -4
Enter the number of columns (n): 4
Please enter positive values for the number of rows and columns.

Discussion

- ***User Input:*** The program starts by using the Scanner class to take user input for the number of rows (m) and columns (n) of a matrix. It checks if the values entered are positive, and if not, it displays an error message and exits the program.
- ***Matrix Input:*** After obtaining the dimensions of the matrix, the program creates a 2D array matrix to store the elements of the matrix. It then uses nested loops to fill in the matrix with user-provided values.
- ***Row Minimums and Column Maximums:*** Next, the program calculates the minimum value for each row (*rowMin*) and the maximum value for each column (*colMax*) in the matrix. It uses nested loops to iterate through the elements of the matrix and updates these arrays accordingly.
- ***Total Sum Calculation:*** The program also calculates the total sum of all elements in the matrix by iterating through it with nested loops and accumulating the values in the *totalSum* variable.
- ***Printing the Modified Matrix:*** The program then prints the modified matrix. It displays the original matrix along with the row minimums and column maximums. Each row is followed by its respective row minimum, and each column maximum is displayed in a separate row.
- ***Printing Total Sum:*** Finally, the program prints the total sum of all elements in the matrix.