2D Array Exercises: Your Name: Shyaan Khan

Create one class TwoDArrays which includes a main method , and the static methods listed below. You may use example arrarys provided below as well as additional ones to ensure you have run sufficient test cases. Copy/paste your method code under each corresponding question (1 -7).

basic = new int[][] { {1,2,3}, {4,5,6}, {7,8,9} };

allneg = new int[][] { {-10,-12,-3}, {-4,-5,-6,-8}, {-7,-8} }; //all neg and ragged

nonsquare = new int[][] { {1,2,3}, {4,5}, {6,7,8,9} };

negatives = new int[][] { {1,-2,3}, {4,5,6}, {-7,8,-9} };

rowmagic = new int[][] { {1,2,3}, {-1,5,2}, {4,0,2} };

colmagic = new int[][] { {1,-1,4,10}, {3,5,0,-6} };

magic = new int[][] { {2,2,2}, {2,2,2}, {2,2,2} };

notmagic1 = new int[][] { {1,2,3}, {4,5,6}, {6,8,9} }; //diag sums are not equal

notmagic2 = new int[][] { {1,5,3}, {4,5,6}, {7,8,9} }; //diag sums //are equal but rows are not

1. Write a method public static int max(int[][] a) that returns the maximum value in the 2d parameter array a.

```java
public static int max(int [][] a)
	{
		int [][] tempArray = a;
		int maxValue = tempArray[0][0];
		for(int row = 0; row<tempArray.length; row++)
		{
			for (int col = 0; col<tempArray[0].length; col++)
			{
				if (tempArray[row][col]> maxValue)
				{
					maxValue = tempArray[row][col];
				}
			}
		}

		return maxValue;
	}
```

2. Write a method public static int rowSum(int[][] a, int x) that returns the sum of the elements in Row x of a.

```
public static int rowSum(int[][] a, int x)
{
        int [][] tempArray = a;
        int sum = 0;

        for(int row = x;row==x;row++)
        {
                for(int col = 0; col < tempArray[0].length; col++)
                {
                        int element = tempArray[row][col];
                        sum+=element;
                }
        }
        return sum;
}
```

3. Write a method public static int columnSum(int[][] a, int x)

```
public static int columnSum(int[][] a, int x)
{
        int [][] tempArray =  a;
        int sum = 0;

        for(int row = 0; row <tempArray.length; row++)
        {
                for(int col = x; col == x; col++)
                {
                        int element = tempArray[row][col];
                        sum += element;
                }
        }

        return sum;
}
```

4. Write a method public static boolean isRowMagic(int[][] a) that checks if the array is row-magic (this means that every row has the same row sum).

```
public static boolean isRowMagic(int[][] a )
        {
                int [][] tempArray = a;
                boolean magic = false;
                for(int i = 0; i < tempArray[0].length -1; i++)
                {
                        if(rowSum(tempArray, i) == rowSum(tempArray, i++))
                        {
                                magic = true;
                        }
                        else
                        {
```

```
                magic = false;
            }
        }
        return magic;
    }
```

5. *Write a method public static boolean isColumnMagic(int[][] a) that checks if the array is column-magic (this means that every column has the same column sum).

```
    public static boolean isColumnMagic(int[][] a)
    {
        boolean columnMagic = true;
        int maxLength = a[0].length;
        for(int i = 0; i <a.length; i++)
        {
            if (a[i].length > maxLength)
            {
                maxLength = a[i].length;
            }
        }

        int val = columnSum(a, 0);
        for(int i = 0; i<maxLength; i++)
        {
            if (val != columnSum(a,i))
                {
                    columnMagic= false;
                }

        }
        return columnMagic;
    }
```

6. Write a method public static boolean isSquare(int[][] a) that checks if the array is square (i.e. every row has the same length as a itself).

```
    public static boolean isSquare(int[][]a)
    {
        int [][] tempArray = a;
        boolean square = true;
        int length = tempArray.length;

        for(int i = 0; i < length; i++)
        {
            if(length == tempArray[i].length)
            {
                square = true;
            }
            else
            {
                square = false;
            }
        }

        return square;
    }
```

7. Write a method public static boolean isMagic(int[][] a)that checks if the array is a magic square. This means that it must be square, and that all row sums, all column sums, and the two diagonal-sums must all be equal.

```
public static boolean isMagic(int [][] a)
    {
        boolean magicSquare = false;

        int [][]tempArray = a;
        for(int row = 0; row < tempArray.length; row++)
        {
            int rowSum = rowSum(tempArray, row);

            for(int col = 0; col < tempArray[0].length; col++ )
            {
                int colSum = columnSum(tempArray, col);
                if(rowSum == colSum)
                {
                    boolean square = isSquare(tempArray);
                    if (square = true)
                    {
                        magicSquare = true;
                    }
                    else
                    {
                        magicSquare = false;
                    }
                }
                else
                {
                    magicSquare = false;
                }
            }
        }

        return magicSquare;
    }
```