Question:

Write a Program in Java to input a 2-D array of size 'm*n' and print its boundary (border) elements.

For example:

INPUT

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

OUTPUT

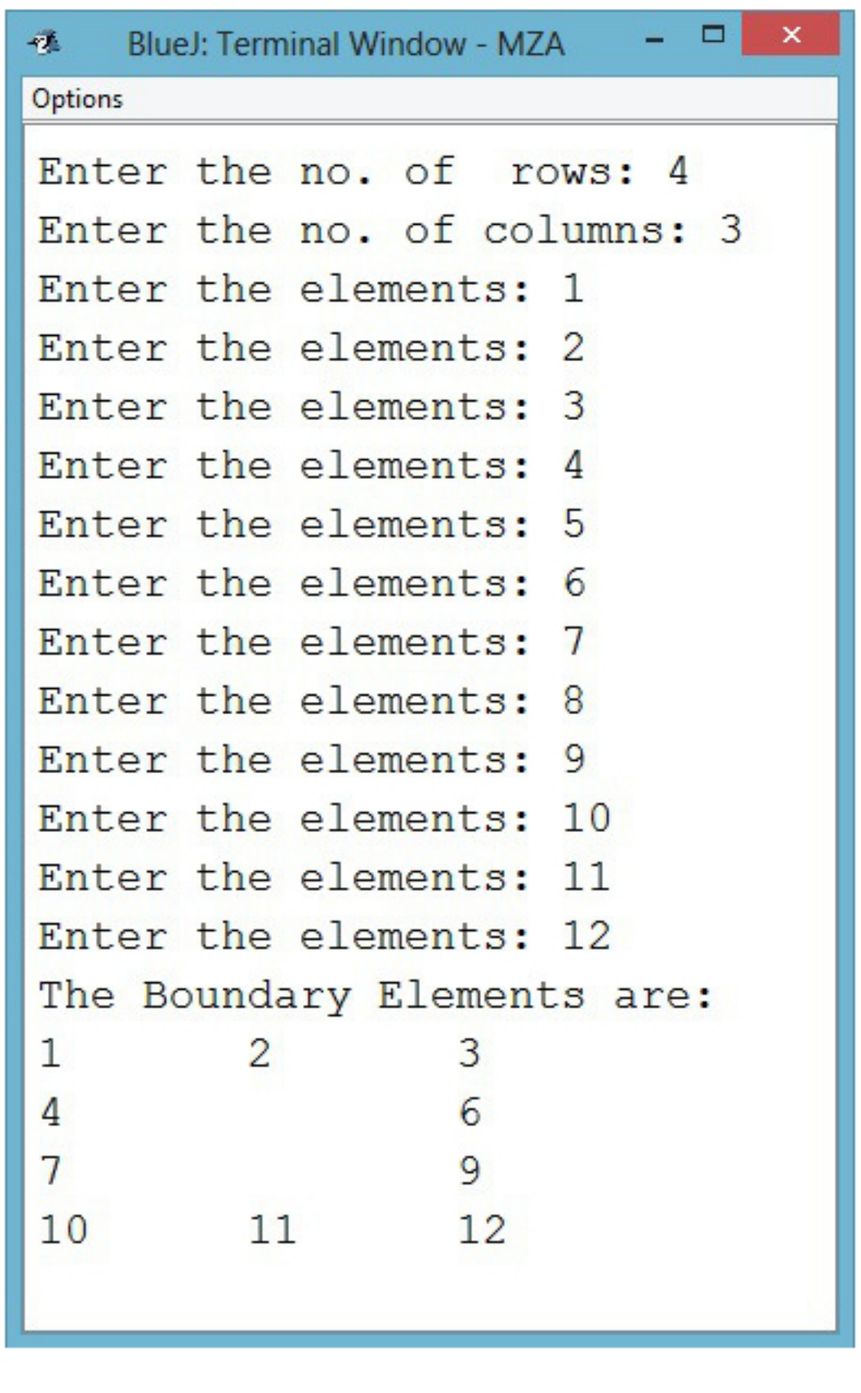
1	2	3	4	5
6				10
11				15
16	17	18	19	20

Programming Code:

```java
/**
* The class Boundary_Element accesses and prints the boundary elements of a 2D array
* @author : www.javaforschool.com
* @Program Type : BlueJ Program - Java
*/

import java.io.*;
class Boundary_Element
{
    public static void main(String args[])throws IOException
    {
        int i,j,m,n;
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

        System.out.print("Enter the no. of  rows: "); //Inputting the number of rows
        m=Integer.parseInt(br.readLine());
        System.out.print("Enter the no. of columns: "); //Inputting the number of columns
        n=Integer.parseInt(br.readLine());

        int A[][]=new int[m][n]; //Creating the array

        /* Inputting the array */
        for(i=0;i<m;i++)
        {
            for(j=0;j<n;j++)
            {
                System.out.print("Enter the elements: ");
                A[i][j]=Integer.parseInt(br.readLine());
            }
        }

        System.out.println("The Boundary Elements are:");
        for(i=0;i<m;i++)
        {
            for(j=0;j<n;j++)
            {
                if(i==0 || j==0 || i == m-1 || j == n-1) //condition for accessing boundary elements
                    System.out.print(A[i][j]+"\t");
                else
                    System.out.print(" \t");
            }
            System.out.println();
        }
    }
}
```

Note: If you are asked to input a square matrix of size 'n*n' then just input the value of 'n' and replace 'm' and 'n' in the above program with 'n'.

Output:



Question:

Write a Program in Java to fill a 2-D array with the first 'm*n' prime numbers, where 'm' is the number of rows and 'n' is the number of columns.

For example: If rows = 4 and columns = 5, then the result should be:

2	3	5	7	11
13	17	19	23	29
31	37	41	43	47
53	59	61	67	71

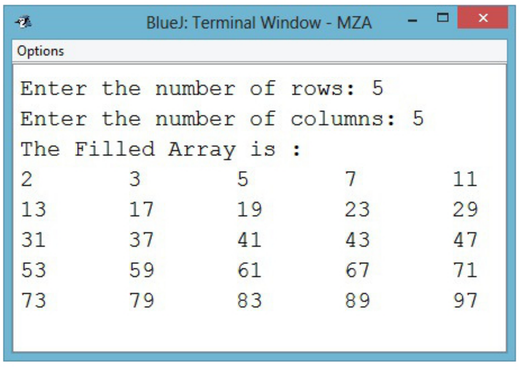
Solution:

```
/**
* The class FillPrime fills a 2D array with 'm*n' Prime numbers
* @author : www.javaforschool.com
* @Program Type : BlueJ Program - Java
*/

import java.io.*;
class FillPrime
{

    boolean isPrime(int n) // Function for checking whether a number is prime or not
    {
        int c = 0;
        for(int i = 1; i<=n; i++)
        {
            if(n%i == 0)
                c++;
        }
        if(c == 2)
            return true;
        else
            return false;
    }

   public static void main(String args[])throws IOException
   {
       FillPrime ob = new FillPrime();
       BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

       System.out.print("Enter the number of rows: ");
       int m=Integer.parseInt(br.readLine());
       System.out.print("Enter the number of columns: ");
       int n=Integer.parseInt(br.readLine());

       int A[][]=new int[m][n]; // 2D array for storing 'm*n' prime numbers
       int B[] = new int [m*n]; // 1D array for storing 'm*n' prime numbers

       int i = 0, j;
       int k = 1; // For generating natural numbers

       /* First saving the 'm*n' prime numbers into a 1D Array */
       while(i < m*n)
       {
           if(ob.isPrime(k)==true)
           {
                   B[i] = k;
                   i++;
           }
           k++;
       }

       /* Saving the 'm*n' prime numbers from 1D array into the 2D Array */
       int x = 0;
       for(i=0; i<m; i++)
        {
           for(j=0; j<n; j++)
           {
               A[i][j] = B[x];
               x++;
           }
        }

       /* Printing the resultant 2D array */
       System.out.println("The Filled Array is :");
       for(i=0; i<m; i++)
        {
            for(j=0; j<n; j++)
                {
                    System.out.print(A[i][j]+"\t");
                }
            System.out.println();
        }
   }
}
```

Note: If you are asked to input a square matrix of size 'n*n' then just input the value of 'n' and replace 'r' and 'c' in the above program with 'n'.

Similarly, you can fill a 2D array with any type of number. Just replace the function isPrime() in the above program with the appropriate function.

Output:



2D- ARRAY

Write a program to declare a square matrix A[ ][ ] of order N (N<20). Allow the user to input positive integers into this matrix. Perform the following tasks on the matrix:

a) Output the original matrix.
b) Find the SADDLE POINT for the matrix. A saddle point is an element of the matrix such that it is the minimum element from the row to which it belongs and the maximum element for the column to which it belongs. Saddle point for a given matrix is always unique. If the matrix has no saddle point, out the message "NO SADDLE POINT"
c) Sort the elements along principle diagonal in ascending order using insertion sort technique. All other elements should remain unchanged.

Test your program for the following data and some random data:

SAMPLE DATA :

INPUT : N = 4

Matrix A[ ][ ] =
2 5 6 9
8 4 12 3
6 7 3 1
12 24 2 11

OUTPUT :
2 5 6 9
8 4 12 3
6 7 3 1
12 24 2 11

NO SADDLE POINT

2D- ARRAY

MATRIX AFTER SORTING THE PRINCIPLE DIAGONAL
2 5 6 9
8 3 12 3
6 7 4 1
12 24 2 11

INPUT : N = 3

Matrix A[ ][ ] =
4 16 12
2 8 14
1 3 6

OUTPUT :
4 16 12
2 8 14
1 3 6

SADDLE POINT = 4

MATRIX AFTER SORTING THE PRINCIPLE DIAGONAL
4 16 12
2 6 14
1 3 8

SOLUTION

D K Singh https://sites.google.com/site/javaprogramsisc/

```
import java.io.*;
class ISCsaddlepoint
{
    public static void main(String arg[]) throws IOException
    {
    BufferedReader br = new BufferedReader(new
                                        InputStreamReader(System.in));
    int n,i,j,k,p, max,min;
    int a[ ][ ]=new int[20][20];
    double y;
    System.out.println("Enter size of an array");
    n=Integer.parseInt(br.readLine());                    //Enter value of n
     System.out.println("Enter matrix");
    for(i=0;i<n;++i)            //Enter array
          for(j=0;j<n;++j)
                a[i][j]=Integer.parseInt(br.readLine());
    int r=0;
     int c=0;
     for (i=0;i<n;i++)
          for (j=0;j<n;j++)
          {
            r=1;
            c=1;
            p=a[i][j];
            max=min=p;
            for(k=0;k<n;k++)
              {
                if(a[i][k]<min)  r=0;
                if (r==1)
                {   for(k=0;k<n;k++)
                            if(a[k][j]>max)  c=0;
                    if (c==1)      System.out.println(a[i][j]+"saddle point");
                    }
                }
          }
    }
}
```

2D- ARRAY

Given a square matrix list[ ][ ] of order 'n'. The maximum value possible for 'n' is 20. Input the value for 'n' and the positive integers in the matrix and perform the following tasks:

a) Display the original matrix
b) Print the row and column position of the largest element of the matrix
c) Print the row the column position of the second largest element of the matrix
d) Sort the elements of the rows in the ascending order and display the new matrix

Sample data:

INPUT: N=3

List [ ][ ]

5	1	3
7	4	6
9	8	2

OUTPUT:

5	1	3
7	4	6
9	8	2

The largest element 9 is in row 3 and column 1
The second largest element 8 is in row 3 and column 2

Sorted List

1	3	5
4	6	7
2	8	9

2008

SOLUTION

D K Singh https://sites.google.com/site/javaprogramsisc/

```
import java.io.*;
class ISC2008PR_03_matrix
{
public int n,i,j;
public int a[ ][ ]=new int[100][100];
public void input() throws IOException
{
   BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
   System.out.println(" Input the order of the matrix");
   n=Integer.parseInt(br.readLine());
   for(i=0;i<n;i++)
      for(j=0;j<n;j++)
         a[i][j]=Integer.parseInt(br.readLine());;
   for(i=0;i<n;i++)
      {
        for(j=0;j<n;j++)
              System.out.print(a[i][j]+"   ");
          System.out.println();
      }
}
public void calculate()
{
int t, k, p, q, max1=a[0][0], max2=a[0][0];
p = q = 0;
for(i=0;i<n;i++)
   for(j=0;j<n;j++)
      if(a[i][j]>max1)
         {          max1=a[i][j];
                    p=i+1;
                    q=j+1;
          }
 System.out.println("largest no. "+max1+" is in row "+p+" and column "+ q);
for(i=0;i<n;i++)
   for(j=0;j<n;j++)
      if(a[i][j]>max2 && a[i][j]!=max1)
         {      max2=a[i][j];
                p=i+1;
                q=j+1;
         }
System.out.println("second largest no. "+ max2 + " row no. "+ p +" column no. "+ q);
for(i=0;i<n;i++)
   for(j=0;j<n;j++)
      for(k=0;k<n-1;k++)
         {
           if(a[i][k]>a[i][k+1])
           {
              t=a[i][k];
              a[i][k]=a[i][k+1];
              a[i][k+1]=t;
           }
         }
}
```

D K Singh https://sites.google.com/site/javaprogramsisc/

```
public void display()
{
for(i=0;i<n;i++)
      {
        for(j=0;j<n;j++)
            System.out.print(a[i][j]+"   ");
         System.out.println();
      }
}

public static void main(String arg[]) throws IOException
{
  ISC2008PR_03_matrix ob=new ISC2008PR_03_matrix();
  ob.input();
  ob.calculate();
  ob.display();
}
}
```

2D- ARRAY

Write a program to declare a square matrix A[ ][ ] of order (M x M) where 'M' is the number of rows and number of columns such that M must be greater than 2 and less than 20. Allow the user to input integers into this matrix. Display appropriate error message for an invalid input.
Perform the following tasks:

a) Display the input matrix.
b) Create a mirror image of the inputted matrix.
c) Display the mirror image matrix

Test your program with the sample data and some random data.

Example 1 :

INPUT: m = 3

4	16	12
8	2	14
6	1	3

OUTPUT: ORIGINAL MATRIX

4	16	12
8	2	14
6	1	3

MIRROR IMAGE MATRX

12	16	4
14	1	8
3	1	6

Example 2 :

INPUT: m = 22

OUTPUT: SIZE OUT OF RANGE

SOLUTION

```
import java.util.*;
class MatrixImage
{
        int a[ ][ ];
        int i, j, m;
        Scanner ob=new Scanner(System.in);
        public void show( )
        {
                System.out.println("Enter size of an Array from 2 to 20 ");
                m=ob.nextInt( );
                if(m>1 && m<=20)
                {
                        a=new int[m][m];
                for(i=0;i<m;i++)
                  {
                    for(j=0;j<m;j++)
                       {
                          System.out.print("Enter value for a[i][j] ");
                          a[i][j]=ob.nextInt( );
                       }
                  }
                System.out.println("Original Matrix is ");
                System.out.println( );
                for(i=0;i<m;i++)
                  {
                    for(j=0;j<m;j++)
                          System.out.print(a[i][j]+"   ");
                    System.out.println( );
                  }
                System.out.println( );
                System.out.println("Image Matrix is ");
                System.out.println( );
                for(i=0;i<m;i++)
                   {
                     for(j=m-1;j>=0;j--)
                        System.out.print(a[i][j]+"   ");
                     System.out.println( );
                   }
                 }
                else System.out.println("SIZE OUT OFF RANGE");
        }
        public static void main(String arg[])
        {
                MatrixImage obj = new MatrixImage( );
                obj.show();
        }
}
```

Question:

Write a program to declare a square matrix A[ ][ ] of order MxM where 'M' is the number of rows and the number of columns, such that M must be greater than 2 and less than 10. Accept the value of M as user input. Display an appropriate message for an invalid input. Allow the user to input integers into this matrix. Perform the following tasks:

(a) Display the original matrix.
(b) Rotate the matrix 90° clockwise as shown below:

Original Matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ Rotated Matrix $\begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$

(c) Find the sum of the elements of the four corners of the matrix.

Test your program for the following data and some random data:

Example 1

INPUT :

M = 3

```
3  4  9
2  5  8
1  6  7
```

OUTPUT :

ORIGINAL MATRIX

```
3  4  9
2  5  8
1  6  7
```

MATRIX AFTER ROTATION

```
1  2  3
6  5  4
7  8  9
```

Sum of the corner elements = 20

Example 2

INPUT :

M = 4

```
1  2  4  9
2  5  8  3
1  6  7  4
3  7  6  5
```

OUTPUT :

ORIGINAL MATRIX

```
1  2  4  9
2  5  8  3
1  6  7  4
3  7  6  5
```

MATRIX AFTER ROTATION

```
3  1  2  1
7  6  5  2
6  7  8  4
5  4  3  9
```

Sum of the corner elements = 18

Example 3

INPUT :
M = 14

OUTPUT :
SIZE OUT OF RANGE

Example 4

INPUT :
M = 112
N = 130

OUTPUT :
INVALID INPUT

Programming Code:

```
/**
* The class Q2_ISC2015 inputs a square matrix and rotates it 90° clockwise
* It also finds and displays the sum of the corner elements
* @author : www.guideforschool.com
* @Program Type : Blue3 Program - Java
* @Question Year : ISC Practical 2015 Question 2
*/

import java.util.*;
class Q2_ISC2015
{
    public static void main(String args[])throws Exception
    {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the size of the matrix : ");
        int m=sc.nextInt();

        if(m<3 || m>9)
            System.out.println("Size Out Of Range");
        else
        {
            int A[][]=new int[m][m];

            /* Inputting the matrix */
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<m;j++)
                {
                    System.out.print("Enter an element : ");
                    A[i][j]=sc.nextInt();
                }
            }

            /* Printing the original matrix */
            System.out.println("*************************");
            System.out.println("The Original Matrix is : ");
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<m;j++)
                {
                    System.out.print(A[i][j]+"\t");
                }
                System.out.println();
            }
            System.out.println("*************************");

            /*Rotation of matrix begins here */
            System.out.println("Matrix After Rotation is : ");
            for(int i=0;i<m;i++)
            {
                for(int j=m-1;j>=0;j--)
                {
                    System.out.print(A[j][i]+"\t");
                }
                System.out.println();
            }
            System.out.println("*************************");

            int sum = A[0][0]+A[0][m-1]+A[m-1][0]+A[m-1][m-1]; // Finding sum of corner elements
            System.out.println("Sum of the corner elements = "+sum);
        }
    }
}
```

Alternate Way (Creating a new Matrix for storing rotated matrix):

If you want, you can also save the rotated matrix in a separate array and then print it.

Programming Code:

```
/**
* The class Q2_ISC2015 inputs a square matrix and rotates it 90° clockwise
* It also finds and displays the sum of the corner elements
* @author : www.guideforschool.com
* @Program Type : Blue3 Program - Java
* @Question Year : ISC Practical 2015 Question 2
*/

import java.util.*;
class Q2_ISC2015
{
    public static void main(String args[])throws Exception
    {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the size of the matrix : ");
        int m=sc.nextInt();

        if(m<3 || m>9)
            System.out.println("Size Out Of Range");
        else
        {
            int A[][]=new int[m][m];

            /* Inputting the matrix */
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<m;j++)
                {
                    System.out.print("Enter an element : ");
                    A[i][j]=sc.nextInt();
                }
            }

            /* Printing the original matrix */
            System.out.println("*************************");
            System.out.println("The Original Matrix is : ");
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<m;j++)
                {
                    System.out.print(A[i][j]+"\t");
                }
                System.out.println();
            }
            System.out.println("*************************");

            int B[][]=new int[m][m];
            int x;

            /*Rotation of matrix begins here */
            for(int i=0;i<m;i++)
            {
                x = m-1;
                for(int j=0;j<m;j++)
                {
                    B[i][j]=A[x][i];
                    x--;
                }
            }

            /* Printing the rotated matrix */
            System.out.println("Matrix After Rotation is : ");
            for(int i=0;i<m;i++)
            {
                for(int j=0;j<m;j++)
                {
                    System.out.print(B[i][j]+"\t");
                }
                System.out.println();
            }
            System.out.println("*************************");

            int sum = A[0][0]+A[0][m-1]+A[m-1][0]+A[m-1][m-1]; // Finding sum of corner elements
            System.out.println("Sum of the corner elements = "+sum);
        }
    }
}
```

Output:

```
Enter the size of the matrix : 4
Enter an element : 1
Enter an element : 2
Enter an element : 4
Enter an element : 9
Enter an element : 2
Enter an element : 5
Enter an element : 8
Enter an element : 3
Enter an element : 1
Enter an element : 6
Enter an element : 7
Enter an element : 4
Enter an element : 3
Enter an element : 7
Enter an element : 6
Enter an element : 5
*************************
The Original Matrix is :
1 2 4 9
2 5 8 3
1 6 7 4
3 7 6 5
*************************
Matrix After Rotation is :
3 1 2 1
7 6 5 2
6 7 8 4
5 4 3 9
*************************
Sum of the corner elements = 18
```