```java
import java.util.*;

/**
 * מחלקת שירות לטיפול במערך דו-ממדי
 * @author Hila Kadman
 *
 *
 *      : עמודות N -ו שורות N הגדרת מטריצה בגודל
 *
 *      int [][] mat = new int [N][M] ;
 *
 */

public class Matrix {

    /**
     * פעולה המאפסת את כל איברי המטריצה
     */
    public static void initMat(int [][]m)
    {
        for (int i = 0 ; i < m.length ; i++)
            for (int j = 0 ; j < m[i].length ; j++)
                m[i][j] = 0;
    }


    /**
     * פעולה הקולטת איברים למטריצה
     */
    public static void matKelet (int [][]m)
    {
        Scanner in = new Scanner (System.in);
        for (int i = 0 ; i < m.length ; i++)
            for (int j = 0 ; j < m[i].length ; j++)
            {
                System.out.print(j + " : עמודה " + i
                                 + " : הקש ערך לאיבר בשורה --> ");
                m[i][j] = in.nextInt();
            }
    }


    /**
     *  1 או 0 פעולה הממלאה מטריצה במספרים אקראיים
     */
    public static void matFill (int [][]m)
    {
        Random rnd = new Random ();
        for (int i = 0 ; i < m.length ; i++)
            for (int j = 0 ; j < m[i].length ; j++)
                m[i][j] = rnd.nextInt(2);
    }


    /**
```

```
56       * פעולה המציגה את המטריצה כטבלה
57       */
58      public static void matPelet (int [][]m)
59      {
60
61          for (int i = 0 ; i < m.length ; i++)
62          {
63              System.out.print(i + ":");
64              for (int j = 0 ; j < m[i].length ; j++)
65                  System.out.print("\t" + m[i][j]);
66              System.out.println();
67          }
68      }
69
70
71      /**
72       *             m טענת כניסה: מטריצה
73       *   טענת יציאה: מוצר סכום המטרצה
74       */
75      public static int matSum (int [][]m)
76      {
77          int sum = 0;
78          for (int i = 0 ; i < m.length ; i++)
79              for (int j = 0 ; j < m[i].length ; j++)
80                  sum = sum + m[i][j];
81          return sum;
82      }
83
84
85      /**
86       *   line ושורה m טענת כניסה: מטריצה
87       *   טענת יציאה: מוצר סכום השורה
88       */
89      public static int sumOfLine (int [][]m, int line)
90      {
91          int sum = 0;
92          for (int i = 0 ; i < m[line].length ; i++)
93              sum = sum + m[line][i];
94          return sum;
95      }
96
97
98      /**
99       * col ועמודה  m טענת כניסה:  מטריצה
100      *   col -ה סכום העמודה :טענת יציאה
101      */
102     public static int sumOfColumn (int [][]m, int col)
103     {
104         int sum = 0;
105         for (int i = 0 ; i < m.length ; i++)
106             sum = sum + m[i][col];
107         return sum;
108     }
109
110
```

```
111     /**
112      *           טענת כניסה: מטריצה ריבועית m
113      *   טענת יציאה: מוצר סכום האלכסון הראשי
114      */
115     public static int sumOfFirstDiagonal (int [][]m)
116     {
117         int sum = 0;
118         for (int i = 0 ; i < m.length ; i++)
119             sum = sum + m[i][i];
120         return sum;
121     }
122
123
124     /**
125      *           טענת כניסה: מטריצה ריבועית m
126      *   טענת יציאה: מוחזר סכום האלכסון המשני
127      */
128     public static int sumOfSecondDiagonal (int [][]m)
129     {
130         int j = m[0].length - 1 , sum = 0;
131         for (int i = 0 ; i < m.length ; i++)
132         {
133             sum = sum + m[i][j];
134             j-- ;
135         }
136         return sum;
137     }
138
139
140     /**
141      *                 טענת כניסה: מטריצה m בגודל NxN כך ש ת- N איזוגי
142      * טענת יציאה: מוחזר "אמת" אם סכום פינות המטריצה שוות לאיבר האמצעי
143      *                                         ו-"שקר" אחרת.
144      */
145     public static boolean matCorners (int [][]m)
146     {
147         int N = m.length;
148         int sum = m[0][0] + m[0][N-1] +
149                   m[N-1][0] + m[N-1][N-1];
150         if (m[N/2][N/2] == sum)
151             return true;
152         return false;
153     }
154
155
156
157     /**
158      *               טענת כניסה: מטריצה m , מספר שורה i , מספר עמודה j
159      *   טענת יציאה: מוצר סכום האיברים שנמצאים ב- 4 פינות האיבר הנתון
160      *    1 <= i < m.length-1  ,  1 <= j < m[i].length-1         הנחה:
161      *
162      *          X     X
```

```
163      *              O
164      *           X     X
165      */
166     public static int sumOfCorners (int [][]m, int i , int j)
167     {
168         int sum = m[i-1][j-1] + m[i-1][j+1] +
169                   m[i+1][j-1] + m[i+1][j+1];
170         return sum;
171     }
172
173
174     /**
175      *            j מספר עמודה , i מספר שורה , m מטריצה :טענת כניסה
176      * טענת יציאה: מוצזר סכום האיברים שנמצאים ב- 4 צידי האיבר הנתון
177      * 1 <= i < m.length-1  ,  1 <= j < m[i].length-1          :הנחה
178      *
179      *               X
180      *            X  O  X
181      *               X
182      */
183     public static int sumOfSides (int [][]m, int i , int j)
184     {
185         int sum = m[i-1][j] + m[i][j-1] + m[i][j+1] + m[i+1][j];
186         return sum;
187     }
188
189
190     /**
191      *            y מספר עמודה , x מספר שורה , m מטריצה :טענת כניסה
192      *     טענת יציאה: מוחזר סכום האיברים שנמצאים מסביב לאיבר הנתון
193      * 1 <= i < m.length-1  ,  1 <= j < m[i].length-1          :הנחה
194      *
195      *            X  X  X
196      *            X  O  X
197      *            X  X  X
198      */
199     public static int sumOfNeighbours (int [][]m, int x , int y)
200     {
201         int sum = 0;
202         for (int i = x-1 ; i <= x + 1 ; i++)
203             for (int j = y-1 ; j <= y + 1 ; j++)
204                 sum = sum + m[i][j]; // הלולאה סיכמה גם את האיבר הנתון
205         return sum - m[x][y];        //     לכן נחסר אותו מהערך המוחזר
206     }
207
208 }
209
210
211
212
```