2. Write a Java program to generate a magic square of order n (all row, column, and diagonal sums are equal).From Wikipedia, In recreational mathematics and combinatorial design, a magic square is a n x n square grid (where n is the number of cells on each side) filled with distinct positive integers in the range 1, 2, ..., n2 such that each cell contains a different integer and the sum of the integers in each row, column and diagonal is equal. The sum is called the magic constant or magic sum of the magic square. A square grid with n cells on each side is said to have order n.

Analysis and research

• N must be odd

•The sum of the elements of all rows, columns and diagonal result in the same value, which is given by the formula: numSum = n(n^2+1 / 2).
•There is a pattern to generate magic squares that consists of placing the first number (1) in the row that is in the middle of the square, and in the last column, that is, (1) will be in the position (n/2, n-1) since it gives me half of the square.
•Then, to place the next element, the row is decreased by 1 and the column is increased by 1, i.e. (i-1, j+1). But by doing this, because the first element starts in the row that is in the middle of the square, at some point when it is decreasing it could half of the square, at some point when it is decreasing it could reach row 0 (first row), and when it decreases again I will get row -1, in this case I will have to jump to the last row of the square which will be (n-1). square which will be (n-1). In the same way, as I start in the last column, at some point I will go beyond the last column (n). the last column (n) and jump to the first column, which will be 0.
•It can also happen that it falls in a position where a number has already been calculated, in this case the column is decreased by 2, and the row is increased by .
•If at some point it happens that the rows are equal to -1 and the column to n (at the same time), it will go to position (0, n-2).

```java
public static void cuadradoMagico (int n){
    int[][] cuadrado = new int[n][n];
    // Inicializo en la fila del medio, y la ultima columna
    int row = n/2;
    int column = n-1;
    // Coloco el primer elemento y paso al siguiente
    int elem = 1;
    cuadrado[row][column] = elem;
    elem++;
    row--;
    column++;
    // Los elementos del cuadrado iran desde 1 hasta n^2
    for (;elem <= n*n;){
        // Caso en el cual me pase de la primer fila, y de la ultima columna
        if (row == -1 && column == n){
            row = 0;
            column = n-2;
        }else{
            // Caso en el el que solo me pase de la primer fila
            if (row < 0){
                row = n-1;
            }
            // Caso en el que solo me pase de la ultima columna
            if (column >= n){
                column = 0;
            }
        }
        //Caso en el que ya este colocado un numero en esa posicion
        if (cuadrado[row][column] != 0){
            row++;
            column -= 2;
            continue; // Paso al siguiente ciclo
        }else{
            /*
            * Imprimo el elemento en su posicion correspondiente ya validada por
todos las condiciones anteriores
            * y paso al siguiente elemento
            */
            cuadrado[row][column] = elem;
            elem++;
        }
            // Proxima posicion
            row--;
            column++;
    }
    // Imprimo el cuadrado obtenido
    for (int i = 0; i < cuadrado.length ; i++){
        for (int j = 0; j < cuadrado.length; j++){
            // Imprimo cada elemento con un espacio entre ellos (columna)
            System.out.print(cuadrado[i][j] + " ");
        }
```

```
            //Hago un salto de linea para imprimir las filas
            System.out.println();
        }
        // Imprimo el resultado de la suma de los elementos de las filas, columnas o
diagonal
        int numSum = n*(n*n+1)/2;
        System.out.println("La suma de los elementos de las filas, columnas o diagonal
dan: " + numSum);
    }
```

Examples

```
n = 3
/* Returns
2 7 6
9 5 1
4 3 8
La suma de los elementos de las filas, columnas o diagonal dan: 15
*/

n = 5
/* Returns
9 3 22 16 15
2 21 20 14 8
25 19 13 7 1
18 12 6 5 24
11 10 4 23 17
La suma de los elementos de las filas, columnas o diagonal dan: 65
*/

n = 7
/* Returns
20 12 4 45 37 29 28
11 3 44 36 35 27 19
2 43 42 34 26 18 10
49 41 33 25 17 9 1
40 32 24 16 8 7 48
31 23 15 14 6 47 39
22 21 13 5 46 38 30
La suma de los elementos de las filas, columnas o diagonal dan: 175
*/
```