JAVA

1. Describe what Hibernate is and what it is used for.

1. Describa qué es Hibernate y para qué se utiliza.

Es una herramienta de mapeo objeto-relacional que busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su [modelo de datos](https://es.wikipedia.org/wiki/Modelo_de_datos), qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la [POO](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos). Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por [SQL](https://es.wikipedia.org/wiki/SQL). Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución (Hibernate. (2022, 2 de septiembre). *Wikipedia, La enciclopedia libre*. Fecha de consulta: 05:32, enero 20, 2023 desde <https://es.wikipedia.org/w/index.php?title=Hibernate&oldid=145729048>.).

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.

import java.util.\*;

public class solution {

    public static void main(String[] *args*) {

        Scanner scan = new Scanner(System.in);

        System.out.print("Input a number: ");

        int num = scan.nextInt();

        if ((num % 2 == 0) || (num <=0 ))

        {

            System.out.print("Input number must be odd and >0");

            System.exit(0);

        }

        int[][] magic\_square = new int[num][num];

        int row\_num = num-1;

        int col\_num = num/2;

        magic\_square[row\_num][col\_num] = 1;

        for (int i = 2; i <= num\*num; i++) {

            if (magic\_square[(row\_num + 1) % num][(col\_num + 1) % num] == 0) {

                row\_num = (row\_num + 1) % num;

                col\_num = (col\_num + 1) % num;

            }

            else {

                row\_num = (row\_num - 1 + num) % num;

            }

            magic\_square[row\_num][col\_num] = i;

        }

*// print the square*

        for (int i = 0; i < num; i++) {

            for (int j = 0; j < num; j++) {

                if (magic\_square[i][j] < 10)  System.out.print(" ");

                if (magic\_square[i][j] < 100) System.out.print(" ");

                System.out.print(magic\_square[i][j] + " ");

            }

            System.out.println();

        }

    }

}

*//Salida de muestra:*

Introduzca un número: 13

 79 94 109 124 139 154 169 2 17 32 47 62 77

 78 80 95 110 125 140 155 157 3 18 33 48 63

 64 66 81 96 111 126 141 156 158 4 19 34 49

 50 65 67 82 97 112 127 142 144 159 5 20 35

 36 51 53 68 83 98 113 128 143 145 160 6 21

 22 37 52 54 69 84 99 114 129 131 146 161 7

  8 23 38 40 55 70 85 100 115 130 132 147 162

163 9 24 39 41 56 71 86 101 116 118 133 148

149 164 10 25 27 42 57 72 87 102 117 119 134

135 150 165 11 26 28 43 58 73 88 103 105 120

121 136 151 166 12 14 29 44 59 74 89 104 106

107 122 137 152 167 13 15 30 45 60 75 90 92

 93 108 123 138 153 168 1 16 31 46 61 76 91

Consultado desde: <https://www.w3resource.com/java-exercises/math/java-math-exercise-19.php>

3) Given an array of integers.

Write an algorithm that brings all nonzero elements to the left of the array,

and returns the number of nonzero elements.

The algorithm should operate in place, i.e. shouldn't create a new array.

The order of the nonzero elements does not matter.

The numbers that remain in the right portion of the array can be anything.

Example:

Given the array [ 1, 0, 2, 0, 0, 3, 4 ], A possible answer is [ 4, 1, 3, 2, ?, ?, ? ],

4 non-zero elements, where "?" can be any number.

Code should have good complexity and minimize the number of writes to the array.

[ 1, 0, -2, 0, 0, 3, 4, 0, 0]

[ 4, 1, 3, -2, ?, ?, ? ] return 4

- What is the complexity of your algorithm?