Ejercicios Modularidad, Funciones

Nota: Las funciones y métodos propuestos tienen que usarse en el programa principal (main) para poder comprobar su correcto funcionamiento.

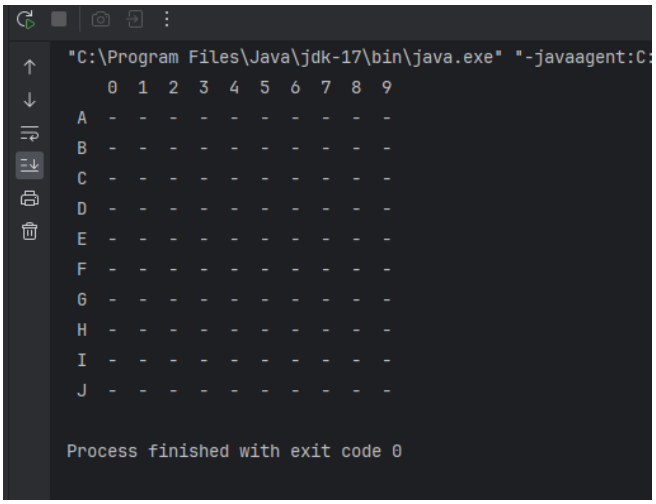
1. Realiza una función llamada suma que sume dos números enteros y te devuelva el resultado, posteriormente llámala desde el main y muestra su resultado.
2. Realiza un método llamado holaMundo que al ser llamada muestre por consola el mensaje “Hola Mundo!”.
3. Realiza un método llamado eco con un parámetro de entrada n, que al ser llamada muestre por consola el mensaje “ECO” un número n de veces.
4. Escribe un método llamado mostrarIntermedios que tenga dos parámetros de entrada a y b (números enteros) y muestre todos los números pares entre ambos números (a y b inclusive).
5. Realiza una función areaCirculo que reciba un parámetro r y te devuelva el área de un círculo de radio r.

   Nota: En un circulo: area = pi * r * 2
   Nota: en Java podemos usar el número pi con la constante de la clase Math PI (se accede con Math.PI)

6. Realiza una función que reciba dos números enteros y devuelva el mayor de los dos, si son iguales devuelve cualquiera de los dos.
7. Realiza un método llamado mostrar que te muestre un array bidimensional por consola.
8. Realiza una función llamada busqueda que dado un array de enteros y un elemento a buscar te devuelva el valor de la posición donde se encuentra el elementos a buscar en el array (una cualquiera de sus apariciones).
9. Realiza un método llamado sumaPorElementoM que reciba un array de enteros y un entero y le sume a cada posición del array recibido el entero recibido. (Ojo, es un método)
10. Realiza una función llamada sumaPorElementoF que reciba un array de enteros y un entero y devuelva un array con el tamaño del array recibido y que contiene en cada posición la suma del elemento del array recibido con el entero recibido

(comportamiento parecido al ejercicio anterior). (Ojo, es una función)

11. Trata de explicar las diferencias entre el comportamiento del método del ejercicio 9 y de la función del ejercicio 10. Fíjate cómo se utilizan en el main.
12. Realiza una función llamada negacionBooleana que reciba un array de booleanos y devuelva un array de las mismas dimensiones pero donde antes había un true pasará a haber un false y donde antes había un false pasará a haber un true.
13. Haz una función llamada factorial que reciba un entero y te devuelva el resultado del factorial de ese número. Usa esa función factorial para llamarla desde un método llamado factorialArray que reciba un array de enteros (se presupone que mayores de 0) y cambie el valor de cada posición por la del factorial del número que contiene. (Ojo, es un método).
14. Realiza una función que cree un array bidimensional de char 10x10 y lo inicialice poniendo en todas sus posiciones el valor '-'.
15. Haz un método llamado mostrarTableroRaya que genere un tablero de 10x10 donde en todas las casillas está el caracter '-' y muestre el tablero de la siguiente forma: (NOTA: utiliza la función anterior)



16. Realiza una función copyArrayMasUno que reciba un array y devuelva un array con los elementos copiados y una posición más con el valor 0.

17. Realiza una función llamada noRepetidos que reciba un array de enteros que pueden estar repetidos y devuelva un nuevo array sin ningún entero repetido. (NOTA: usa la función anterior)

Ejemplos:
entrada: [1, 3, 0, 1, 3, 6, 2, 1] → salida: [1, 3, 0, 6, 2]
entrada: [1, 1, 0, 1, 2, 1, 2, 1] → salida: [1, 0, 2]

18. Realiza una función llamada invertirArray que reciba un array de enteros y devuelva otro array con las posiciones invertidas:

Ejemplos:
entrada: [1, 3, 6, 2, 4, 1] → salida: [1, 4, 2, 6, 3, 1]

19. Realiza un método llamado ordenacionM que reciba un array de enteros y un char, si el char tiene el valor ‘c’ realiza la ordenación creciente, si el char tiene el valor ‘d’ realiza la ordenación decreciente.

Ejemplo:
entrada: [2, 4, 1, 3, 9, 3, 1, 1], ‘c’ → salida: [1, 1, 2, 3, 3, 4, 9]
entrada: [2, 4, 1, 3, 9, 3, 1, 1], ‘d’ → salida: [9, 4, 3, 3, 2, 1, 1]

20. Realiza una método que dado un array de números enteros positivos mayor o igual que 1 (naturales) lo ordene según el número de divisores que tiene cada número.

Ejemplo:el número 6 (tiene cuatro divisores: 1, 2, 3, 6), el número 12 (tiene seis divisores: 1, 2, 3, 4, 6, 12), el número 17 (tiene dos divisores: 1, 17), el número 5 (tiene dos divisores: 1 y 5). Por lo tanto su orden sería:

entrada: [1, 5, 6, 17, 12] → salida: [12, 6, 17, 5, 1] (También podría ir el 5 antes que el 17 puesto que tienen el mismo número de divisores)

NOTA1: Haz primero una función compararPorNumDivisores que reciba dos números enteros mayores o iguales que uno y los compare de tal forma que:

si el primero tiene más divisores que el segundo la salida sea 1
si el segundo tiene más divisores que el primero la salida sea -1
si tienen ambos el mismo número de divisores la salida sea 0

NOTA2: Realiza una función numDivisores que reciba un número entero mayor o igual que 1 y devuelva el número de divisores que tiene ese número.
Ejemplo:
entrada: 1 → salida: 1
entrada 12 → salida: 6
entrada 5 → salida: 2

NOTA3: Realiza una función isDivisor que reciba dos números enteros mayores o iguales que uno y devuelva si el segundo es divisor del primero (true) o si no es divisor (false).
Ejemplo:
entrada: 12, 5 → salida: false
entrada 12, 6 → salida: true
entrada 5, 5 → salida: true

21. Haz una función llamada desplazamiento que reciba un array de char y un número entero mayor o igual que cero, y devuelva un nuevo array que sea el resultado de desplazar los elementos el número de posiciones introducido. Se consideran las posiciones ciclicas, por ejemplo, si el array tiene tamaño 4 y el elemento de la posición 3 debe desplazarse 2 posiciones, la nueva posición será 1.

    ejemplo:
    entrada: [‘a’, ‘v’, ‘t’, ‘s’, ‘k’], 3 → salida: [‘t’, ‘s’, ‘k’, ‘a’, ‘v’]
    entrada: [‘a’, ‘v’, ‘t’, ‘s’, ‘k’], 0 → salida: [‘a’, ‘v’, ‘t’, ‘s’, ‘k’]
    entrada: [‘a’, ‘v’, ‘t’, ‘s’, ‘k’], 1 → salida: [‘k’, ‘a’, ‘v’, ‘t’, ‘s’]

22. Realiza una función llamada sumatorioDivisoresArray que realice el sumatorio de los divisores de todos los números de un array.
    Ejemplo:
    entrada: [6, 5, 4] → salida: 25
    seis tiene de divisores a 1, 2, 3 y 6
    cinco tiene de divisores a 1 y 5
    cuatro tiene de divisores a 1, 2 y 4
    la suma total de todos los divisores de los números del array es:
    (1+2+3+6)+(1+5)+(1+2+4) = 12+6+7 = 25

23. Realiza una función llamada subsecuenciaCrecienteMasLarga que dado un array de enteros te devuelva un entero que indique la posición de inicio de la subsecuencia creciente más larga. En caso de empate devuelve cualquiera de las posiciones de inicio de ellas.

    Ejemplo:
    entrada: [2, 4, **1, 3, 5, 8, 12**, 2, 3, 2, 1, 3, 5, 7, 8]
    salida: 2
    (La subsecuencia de enteros más larga en este caso es la que comienza en la posición 2, y está formada por 1, 3, 5, 8, 12)

24. Realiza un juego llamado alfil vs caballo, se juega en un tablero 8x8 de casillas blancas □ (`\u25A0`) y casillas negras ■ (`\u25A1`) semejante al ajedrez, se situarán de forma aleatoria las piezas caballo ♘ (`\u265E`) y alfil ♗ (`\u2657`), teniendo en cuenta que no estén en la misma casilla. Tras esto, el programa mostrará el tablero con las piezas colocadas y dejará mover (le pedirá las coordenadas por consola) al usuario1 y al usuario2 mostrando tras cada movimiento el tablero con la nueva posición de las piezas. Si alguno de los usuarios mete alguna coordenada incorrecta el programa deberá avisarle y volverá a pedírsela. El juego acaba cuando alguno de los dos usuarios se coma al otro. Es conveniente que el tablero se muestre a los usuarios con las columnas y filas numeradas con letras y números como en el ejercicio 15.

25. Realiza el juego anterior de nuevo, pero en esta ocasión, en vez de dos usuarios enfrentándose, habrá solo uno, que se enfrentará contra la máquina. La máquina seguirá el siguiente algoritmo: si la pieza del usuario está en una posición donde pueda comerla, se la comerá; si la pieza del usuario no está en una posición comible se moverá a alguna otra posición aleatoriamente que no pueda ser comida por el usuario en el siguiente turno.

26. Realiza un método llamado generarTresXAleatorias que reciba un array 8x8 de char “*espacio”* y un número “*cant”* e introduzca aleatoriamente *cant* veces bloques de tres X seguidas tanto horizontal como verticalmente (de forma aleatoria). Las X no podrán estar en la misma posición que otros bloques y tendrá que dejarse un espacio tanto vertical como horizontalmente (no diagonalmente) respecto a otros bloques de X. En las otras casillas habrá rayas (-). Finalmente el programa deberá mostrar por pantalla el tablero que ha generado el método.

Ejemplo: para una entrada espacio = new char[8][8], cant = 4, una posible salida sería: