Arrays. Ejercicios de recorrido

1. (Estaturas*)* Escribir un programa que lea de teclado la estatura de 10 personas y las almacene en un array. Al finalizar la introducción de datos, se mostrarán al usuario los datos introducidos con el siguiente formato:

```
Persona 1: 1.85 m.
Persona 2: 1.53 m.
...
Persona 10: 1.23 m.
```

2. (Lluvias) Se dispone de un array que contiene 31 datos correspondientes a las precipitaciones caídas en el mes de enero del pasado año. Se desea analizar los datos del array para extraer cierta información. Desarrollar los siguientes métodos y llamarlos desde el método main para probarlos:

   2.1. Devolver la lluvia total caída en el el mes:

   - *public static double lluviaTotal (double p[]),* que devuelve la suma de los elementos del array

   2.2. Devolver lo que ha llovido de media en el mes:

   - *public static double lluviaMedia (double p[]), que devuelve la media de los elementos del array. Se puede hacer uso del método del apartado anterior.*

   2.3. Devolver cuánto ha llovido el día que más ha llovido:

   - *public static double lluviaMaxima(double p[]), que devuelve el valor más grande almacenado en el array.*

   2.4. Devolver cuánto ha llovido el día que menos ha llovido:

   - *public static double lluviaMinima (double p[]), que devuelve el valor más pequeño almacenado en el array.*

   2.5. Devolver cual fue el día que más llovió (valor entre 0 y 30)

   - *public static int diaMasLluvia (double p[]), que devuelve en qué posición del array se encuentra el elemento más grande. Si éste se repitiera en el array se devolverá la posición en que aparece por primera vez.*

   2.6. Devolver cual fue el dia que más llovió (valor entre 0 y 30)

   - *public static int diaMenosLluvia (double p[]), que devuelve en qué posición del array se encuentra el elemento más pequeño. Si éste se repitiera en el array se devolverá la posición en que aparece por primera vez.*

   2.7. Devolver en cuántos días llovió cierta cantidad

   - *public static int contarDiasLluviaX (double p[], double x), que devuelve el número de veces que el valor x aparece en el array. Se puede usar, por ejemplo, para averiguar en cuántos días no hubo lluvias (lluvia igual a 0).*

   2.8. Devolver cuanto ha llovido en total en cierto rango de días:

   - *public static double lluviaTotalPeriodo (double p[], int ini, int fin), que devuelve la suma de los elementos del array que están entre las posiciones ini y fin. Se puede usar, por ejemplo, para averiguar cuánto ha llovido en la primera quincena del mes.*

   2.9. Devolver en cuántos días llovió menos que al día siguiente

   - *public static int contarDiasLluviaCreciente(double p[]),* que devuelve el número de elementos del array que son menores que el elemento que tienen a continuación.

3. (Dados) El lanzamiento de un dado es un experimento aleatorio en el que cada número tiene las mismas probabilidades de salir. Según esto, cuantas más veces lancemos el dado, más se igualarán las veces que aparece cada uno de los 6 números. Vamos a hacer un programa para comprobarlo.

   - Generaremos un número aleatorio entre 1 y 6 un número determinado de veces (por ejemplo 100.000). Definiremos una constante entera llamada LANZAMIENTOS = 100000

- Tras cada lanzamiento incrementaremos un contador correspondiente a la cifra que ha salido. Para ello crearemos un array veces de 7 componentes, en el que el veces[1] servirá para contar las veces que sale un 1, veces[2] para contar las veces que sale un 2, etc. veces[0] no se usará.
- Cada, por ejemplo, 1000 lanzamientos mostraremos por pantalla las estadísticas que indican que porcentaje de veces ha aparecido cada número en los lanzamientos hechos hasta ese momento (definir una constante MOSTRAR_CADA = 1000). Por ejemplo:

```
Número de lanzamientos: 1000
1: 18 %
2: 14 %
3: 21 %
4: 10 %
5: 18 %
6: 19 %
Número de lanzamientos: 2000
...
...
```

4. (Invertir) Diseñar un método *public static int[] invertirArray[int v[])*, que dado un array v devuelva otro con los elementos de v en orden inverso. Es decir, el último en primera posición, el penúltimo en segunda, etc. Desde el método main crearemos e inicializaremos un array, llamaremos a invertirArray y mostraremos el array invertido.
5. (SumasParciales) Se quiere diseñar un método *public static int[] sumaParcial(int v[]),*

   que dado un array de enteros v, devuelva otro array de enteros t de forma que t[i] = v[0] + v[1] + ...+ v[i]. Es decir:

   t[0] = v[0]

   t[1] = v[0] + v[1]

   t[2] = v[0] + v[1] + v[2]

   ...

   t[10] = v[0] + v[1] + v[2] + ... + v[10]

   Desde el método main crearemos e inicializaremos un array, llamaremos a sumaParcial y mostraremos el array resultante.
6. (Rotaciones) Rotar una posición a la derecha los elementos de un array consiste en mover cada elemento del array una posición a la derecha. El último elemento pasa a la posición 0 del array. Por ejemplo si rotamos a la derecha el array {1,2,3,4} obtendríamos {4,1,2,3}.
   - Diseñar un método *public static void rotarDerecha(int v[]),* que dado un array de enteros rote sus elementos un posición a la derecha.
   - En el método main crearemos e inicializaremos un array y rotaremos sus elementos tantas veces como elementos tenga el array (mostrando cada vez su contenido), de forma que al final el array quedará en su estado original. Por ejemplo, si inicialmente el array contiene {7,3,4,2}, el programa mostrará

     ```
     Rotación 1: 2 7 3 4
     Rotación 2: 4 2 7 3
     Rotación 3: 3 4 2 7
     Rotación 4: 7 3 4 2
     ```

   - Diseña también un método para rotar a la izquierda y pruébalo de la misma forma.
7. (DosArrays) Desarrolla los siguientes métodos en los que intervienen dos arrays y pruébalos desde el método main

   7.1. public static double[] sumaArraysIguales (double a[], double b[]) que dados dos arrays de double a y b, del mismo tamaño devuelva un array con la suma de los elementos de a y b, es decir, devolverá el array {a[0]+b[0], a[1]+b[1], ...}

   7.2. public static double[] sumaArrays(double a[], double b[]). Repite el ejercicio anterior pero teniendo en cuenta que a y b podrían tener longitudes distintas. En tal caso el número de elementos del array resultante coincidirá con la longitud del array de mayor tamaño.

Arrays. Ejercicios de búsqueda

8. (Lluvias – continuación). Continuando con el ejercicio 2, queremos incorporar el programa la siguiente información:

   8.1. Cuál fue el primer día del mes en que llovió exactamente 19 litros (si no hubo ninguno mostrar un mensaje por pantalla indicándolo)

   - *public static int primerDiaLluviaX (double p[], double x),* que devuelve la posición de la primera aparición de x en el array. Si x no está en p el método devolverá -1. El método realizará una búsqueda ascendente para proporcionar el resultado

   8.2. Cuál fue el último día del mes en que llovió exáctamente 8 litros (si no hubo ninguno mostrar un mensaje por pantalla indicándolo)

   - *public static int ultimoDiaLluviaX (double p[], double x),* que devuelve la posición de la ultima aparición de x en el array. Si x no está en p el método devolverá -1. El método realizará una búsqueda descendente para proporcionar el resultado

9. (Tocayos) Disponemos de los nombres de dos grupos de personas (dos arrays de String). Dentro de cada grupo todas las personas tienen nombre distinto, pero queremos saber cuántas personas del primer grupo tienen algún tocayo en el segundo grupo, es decir, el mismo nombre que alguna persona del segundo grupo. Escribir un programa que resuelva el problema (inicializa los dos arrays con los valores que quieras y diseña los métodos que consideres necesarios).

   Por ejemplo, si los nombres son
   {"miguel","**josé**","**ana**","maría"}
   y
   {"**ana**", "**josé**", "luján", "juan", "**josé**", "pepa", "ángela", "sofía", "andrés", "bartolo"),
   el programa mostraría:

```
josé tiene tocayo en el segundo grupo.
ana tiene tocayo en el segundo grupo.
TOTAL: 2 personas del primer grupo tienen tocayo.
```

10. (PrimerImpar) Escribir un método que, dado un array de enteros, devuelva la suma de los elementos que aparecen tras el primer valor impar. Usar main para probar el método.

11. Para determinar si existe algún valor par en un array se proponen varias soluciones. Indica cual / cuales son válidas para resolver el problema.

```
public static boolean haypares1(int v[]) {
       int i=0;
       while (i<v.length && v[i]%2 != 0) {i++;}

       if(v[i]%2 == 0) return true;
       else return false;
}

public static boolean haypares2(int v[]) {
       int i=0;
       while (i<v.length && v[i]%2 != 0) {i++;}

       if(i< v.length) return true;
       else return false;
}

public static boolean haypares3(int v[]) {
       int i=0;
       while (v[i]%2 != 0 && i<v.length) {i++;}

       if(i< v.length) return true;
       else return false;
}
```

```
public static boolean haypares4(int v[]) {
	int i=0;
	encontrado = false;
	while (i<=v.length && !encontrado) {
		if (v[i]%2==0) encontrado = true;
		else i++;
	}
	return encontrado;
}
public static boolean haypares5(int v[]) {
	int i=0;
	encontrado = false;
	while (i<v.length && !encontrado) {
		if (v[i]%2==0) encontrado = true;
		i++;
	}
	return encontrado;
}

public static boolean haypares6(int v[]) {
	int i=0;
	while (i<v.length) {
		if (v[i]%2==0) return true;
		else return false;
	}
}
public static boolean haypares7(int v[]) {
	int i=0;
	while (i<v.length) {
		if (v[i]%2==0) return true;
		i++;
	}
	return false;
}
```

12. (Capicúa) Escribir un método para determinar si un array de palabras (String) es *capicúa*, esto es, si la primera y última palabra del array son la misma, la segunda y la penúltima palabras también lo son, y así sucesivamente. Escribir el método main para probar el método anterior.

13. (Subsecuencia) Escribir un método que, dado un array, determine la posición de la primera subsecuencia del array que comprenda al menos tres números enteros consecutivos en posiciones consecutivas del array. De no existir dicha secuencia devolverá -1.

    Por ejemplo: en el array {23, 8, 12, 6, 7, **9, 10, 11**, 2} hay 3 números consecutivos en tres posiciones consecutivas, a partir de la posición 5: {9,10,11}

14. Se desea comprobar si dos arrays de double contienen los mismos valores, aunque sea en orden distinto. Para ello se ha escrito el siguiente método, que aparece incompleto:

```
public static bolean mismosValores(double v1[], double v2[]){
 encontrado = false;
 int i=0;
 while(i<v1.length && !encontrado) {
         boolean encontrado2 = false;
         int j=0;
         while (j<v2.length && !encontrado2){
                 if(v1[___] == v2[___]) encontrado2 = true;
                 else ____;
         }
         if (encontrado2== ____) encontrado = true;
         else _____;
 }
 return !encontrado;
}
```

    Completa el programa en los lugares donde aparece el símbolo ▢

Matrices

15. (Notas). Se dispone de una matriz que contiene las notas de una serie de alumnos en una serie de asignaturas. Cada fila corresponde a un alumno, mientras que cada columna corresponde a una asignatura. Desarrollar métodos para:

    15.1. Imprimir las notas alumno por alumno.

    15.2. Imprimir las notas asignatura por asignatura.

    15.3. Imprimir la media de cada alumno.

    15.4. Imprimir la media de cada asignatura.

    15.5. Indicar cual es la asignatura más fácil, es decir la de mayor nota media.

    15.6. ¿Hay algún alumno que suspenda todas las asignaturas?

    15.7. ¿Hay alguna asignatura en la que suspendan todos los alumnos?

16. (Ventas). Una empresa comercializa 10 productos para lo cual tiene 5 distribuidores.
    Los datos de ventas los tenemos almacenados en una matriz de 5 filas x 10 columnas, *ventas*, con el número de unidades de cada producto que ha vendido cada distribuidor. Cada fila corresponde a las ventas de un distribuidor (la primera fila, del primer distribuidor, etc.), mientras que cada columna corresponde a un producto :

100	25	33	89	23	90	87	6	5	233
28	765	65	77	987	55	4	66	4	8

    ....

    El array *precio*, de 10 elementos, contiene el precio en € de cada uno de los 10 productos.

    125.2 234.4 453.9

    ...

    Escribe el programa y los métodos necesarios para averiguar:

    16.1. Distribuidor que más artículos ha vendido.

    16.2. El artículo que más se vende.

    16.3. Sabiendo que los distribuidores que realizan ventas superiores a 30.000 € cobran una comisión del 5% de las ventas y los que superan los 70.000 € una comisión del 8%, emite un informe de los distribuidores que cobran comisión, indicando nº de distribuidor, importe de las ventas, porcentaje de comisión e importe de la comisión en €:

17. Dada una matríz con el mismo número de filas y de columnas, diseñar los siguientes métodos:

    17.1. *public static void mostrarDiagonal(int m[][])* que muestre por pantalla los elementos de la diagonal principal.

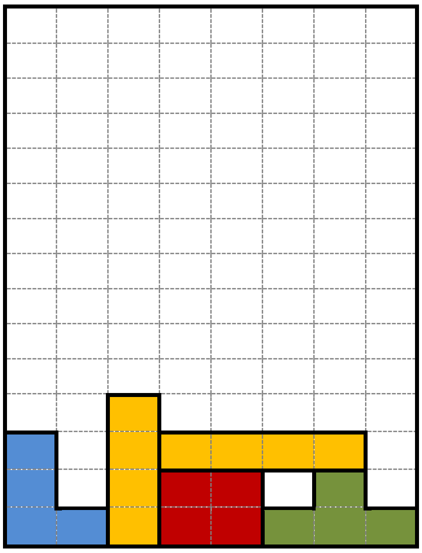
    17.2. *public static int filaDelMayor (int m[][])* , que devuelva la fila en que se encuentra el mayor elemento de la matriz

    17.3. *public static void intercambiarFilas(int f1, int f2),* que intercambie los elementos de las filas indicadas.

18. Escribir un método *public static boolean esSimetrica (int m[][])* que devuelva true si la matriz m es simétrica. Una matriz es simétrica si tiene el mismo número de filas que de columnas y además m[i][j] = m[j][i] para todo par de índices i,j.
    Por ejemplo, es simétrica:

    1 5 3
    5 4 7
    3 7 5

19. Supongamos que estamos desarrollando un *Tetris* en Java y para representar la partida utilizamos una matriz bidimensional de enteros 15 filas por 8 columnas. Se utiliza el valor 0 para indicar que la celda está vacía y un valor distinto de cero para las celdas que contienen parte de una pieza (distintos valores para distintos colores):



0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0
1	0	2	2	2	2	2	0
1	0	2	4	4	0	3	0
1	1	2	4	4	3	3	3

Escribir un método que reciba la matriz y elimine las filas completas, haciendo caer las piezas que hay por encima de las celdas eliminadas tal y como se hace en el juego