Ejercicios Java - Unidad 4

Estructuras de Datos Estáticas

Ejercicio 1

Programa que lea una cadena desde el teclado y muestre en la consola el número de veces que se repite cada vocal.

Ejercicio 2

Programa que lea una cadena en desde el teclado y la muestre invertida en la consola.

Ejercicio 3

Programa que lea dos cadenas desde el teclado y muestre el número de veces que la segunda está contenida en la primera.

Ejercicio 4

Programa que, utilizando una cantidad mínima de variables, simule el lanzamiento de un dado N veces, siendo N un número entero que se introducirá por teclado. Antes de finalizar mostrará el número de veces que salió cada una de las caras.

Ejercicio 5

Programa que lea desde el teclado una línea que contenga un NIF completo (número y letra) y a continuación verifique que es correcto. Para obtener la letra de validación del N.I.F. se realiza la división entera de la parte numérica entre 23 y el resto se utiliza como índice de la tabla siguiente:

Resto	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Letra	T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E

La validación consiste en comprobar que la letra obtenida en la tabla coincide con la letra introducida por teclado.

Ejercicio 6

Programa que realice las tareas siguientes:

- Crear un vector de números enteros de un tamaño aleatorio comprendido entre 10 y 50 elementos.
- Llenar el vector con números aleatorios comprendidos entre -100 y 100 sin repetir ninguno.
- Crear un segundo vector que contenga los elementos del anterior almacenados en orden inverso.
- Mostrar ambos vectores en la consola.

Ejercicio 7

Programa que realice las tareas siguientes:

- Crear un vector de números enteros de un tamaño especificado por el usuario entre 10 y 1.000.000 de elementos.
- Llenar el vector con números aleatorios comprendidos entre -999.999 y 1.000.000 sin repetir ninguno.
- Mostrar en la consola los datos siguientes:
    - Tamaño del vector.
    - Tiempo empleado en llenarlo
    - Diferencia entre el menor y el mayor de los números almacenados en el vector.
    - Tiempo empleado en calcular la diferencia entre el menor y el mayor de los números almacenados.

Ejercicio 8

Programa que realice las tareas siguientes:

- Crear un vector de números enteros de un tamaño especificado por el usuario que estará comprendido entre 10 y 200 elementos.
- Llenar el vector con números aleatorios comprendidos entre -100 y 100.
- Mostrar la suma de los números almacenados teniendo en cuenta que hay quien piensa que el número 13 es el número de la mala suerte. Por tanto, si en alguna posición se encuentra almacenado el número 13, no se sumará ni este número ni los que se encuentren almacenados en las 13 posiciones siguientes (o las que haya hasta el final del vector si estas son menos de 13) si la suma de todos ellos es distinta de 7.
- Mostrar el contenido del vector y la cantidad de números que no se han sumado.

Ejercicio 9

Programa que realice las tareas siguientes:

- Crear un vector de números enteros de un tamaño aleatorio entre 10 y 500 elementos.
- Llenar el vector con números aleatorios comprendidos entre -100 y 100.
- Mostrar el contenido del vector si su tamaño es menor o igual a 50.
- Mostrar, independientemente del tamaño del vector, el número de secuencias de números repetidos.

  Ejemplo:

9	2	2	3	2	5	7	7	7	4

$\rightarrow$ 2 secuencias

Ejercicio 10

Programa que realice las tareas siguientes:

- Crear un vector de números enteros de un tamaño especificado por el usuario que esté comprendido entre 10 y 20 elementos.
- Llenar el vector con números aleatorios.
- Mostrar el contenido del vector.
- Mostrar la mínima diferencia entre dos valores adyacentes. La diferencia entre dos valores adyacentes se calcula como el valor almacenado en cada posición [i] (excepto la primera) menos el valor almacenado en la posición [i-1].

Ejercicio 11

Definir un método que reciba un vector de cadenas y retorne la cadena de mayor longitud almacenada en dicho vector.

Poner a prueba el método invocándolo desde otro método que cree un vector con los nombres de 10 personas introducidos por teclado.

Ejercicio 12

Consideremos un vector de números enteros con índices entre 0 y n. Se define el centro del vector como el índice c que verifica la siguiente propiedad:

$$\sum_{i=0}^{c-1}(c-i)\cdot V[i] = \sum_{j=c+1}^{n}(j-c)\cdot V[j]$$

Esta propiedad no siempre se verifica; en ese caso, decimos que el vector no tiene centro.

Ejemplo, consideremos el siguiente vector:

0	1	2	3	4
6	**2**	**3**	**0**	**1**

El centro de este vector es el índice 1. En efecto, si aplicamos la definición con c = 1 y con n = 4, obtenemos lo siguiente:

$$\sum_{i=0}^{c-1}(c-i)\cdot V[i] = 1\cdot 6 = \sum_{j=c+1}^{n}(j-c)\cdot V[j] = 1\cdot 3 + 2\cdot 0 + 3\cdot 1$$

Por el contrario, el siguiente vector no tiene centro:

0	1	2	3
1	**2**	**1**	**1**

Crear una clase que defina dos métodos:

- Un método llamado `centro` que reciba como parámetro un vector de números enteros y retorne el índice donde se encuentra su centro o `null` si no tiene centro.
- Un método `main` que cree un vector de números enteros leídos por teclado y ponga a prueba el método anterior.

Ejercicio 13

Programa que cree una matriz de números enteros aleatorios con un número de filas y de columnas aleatorios que estén comprendidos entre 2 y 20.

Escribir un método que reciba una matriz de números enteros y retorne un vector que contenga la suma de los valores de cada fila.

Escribir otro método que reciba una matriz de números enteros y retorne un vector que contenga la suma de los valores de cada columna.

Mostrar en la consola la matriz, la suma de las filas a su derecha y la suma de las columnas en la parte inferior.

Ejercicio 14

Programa que utilice un único array para leer y almacenar los datos de varias secuencias de números enteros. Los datos se introducirán por teclado de la forma siguiente:

- Se escribe una primera línea que contiene el número de secuencias.
- A continuación, se escriben las secuencias a razón de dos líneas por secuencia con el formato siguiente:
    - La primera contiene la cantidad de números de la secuencia.
    - La segunda contiene los números de la secuencia separados por espacios.

Ejercicio 15

Programa para la gestión de las calificaciones de los alumnos en las tres evaluaciones de una asignatura. El programa deberá pedir al profesor que introduzca por teclado los nombres de cada alumno junto a las calificaciones de cada evaluación. Después le permitirá realizar las acciones siguientes a través del menú correspondiente:

1. Mostrar la nota media de todos los alumnos.
2. Mostrar la nota media de un alumno determinado.
3. Visualizar las notas por evaluación y la nota final de cada evaluación.
4. Visualizar las notas por evaluación y la nota final de un alumno determinado.
5. Calcular la nota media del curso.
6. Calcular la nota más alta y la más baja e indicar a qué alumno y evaluación pertenece.
7. Salir.

Almacenar los datos en las estructuras de datos estáticas que se consideren necesarias, además de proporcionar una solución basada en la técnica de programación modular.

Ejercicio 16

Crear una clase llamada `MétodosOrdenación` que defina tres métodos estáticos para ordenar vectores de números enteros implementando la ordenación por inserción directa, por selección directa y por intercambio directo respectivamente.

Ejercicio 17

Programa que cree dos vectores de números enteros, cada uno de ellos de una longitud aleatoria entre 10 y 100, los rellene con valores aleatorios, los ordene, los mezcle en un tercer vector manteniendo la ordenación y finalmente muestre el contenido de los tres vectores.

Ejercicio 18

Crea una clase Matrices que defina los métodos de clase siguientes y ponlos a prueba en el método main de otra clase llamada PruebaMatrices:

`static int [][] cuadrada1(int dim)`

Crea una matriz cuadrada de la dimensión especificada en el parámetro dim, la rellena con los números enteros en el rango 1 .. dim*dim en el orden que se indica a continuación y la retorna:

(ejemplo para una matriz cuadrada de dimensión 4)

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

`static int [][] cuadrada2(int dim)`

Crea una matriz cuadrada de la dimensión especificada en el parámetro dim, la rellena con los números enteros en el rango 1 .. dim*dim en el orden que se indica a continuación y la retorna:

(ejemplo para una matriz cuadrada de dimensión 4)

1	8	9	16
2	7	10	15
3	6	11	14
4	5	12	13

`static String [][] palindromos(int c, int f)`

Crea una matriz de **f** filas y **c** columnas y la rellene con palíndromos de tres letras que se generan de la forma siguiente:

- La primera y tercera letra del palíndromo están determinadas el número de fila en la que se va a almacenar:

  fila 0 → letra 'a', fila 1 → letra 'b', fila 0 → letra 'c', …

- La segunda letra del palíndromo está determinada por el valor de la suma fila+columna:

  fil+col=0 → letra 'a', fil+col=1 → letra 'b', fil+col=2 → letra 'c', …

Una vez creada y rellenada, retornará la matriz.

El número de filas y de columnas tiene que estar comprendido entre 1 y 26. De no ser así, el método retornará el valor **`null`**.

(ejemplo para una matriz de 4x6)

aaa	aba	aca	ada	aa	aa
bbb	bcb	bdb	beb	bfb	bgb
ccc	cdc	cec	cfc	cgc	chc
ddd	ded	dfd	dgd	dhd	did

`static int max3x3sum(int [][] matriz)`

Recibe una matriz de NxM y retorna el valor máximo de todos los resultados que se obtienen sumando los elementos de cada matriz de 3x3 contenida en ella.

El valor de N y M deber de ser mayor o igual que 3.

(ejemplo para una matriz de dimensión 4x5)

1	5	5	2	4
2	1	4	14	3
3	7	11	2	8
4	8	12	16	4

= 75

Ejercicio 19

Definir un método **`rellenar`** que declare los parámetros formales siguientes:

- Matriz de caracteres en la que se almacenan letras minúsculas y dígitos numéricos (**`a-z`**, **`0-9`**).
- Un índice de fila al que llamaremos **`filaInicial`**.
- Un índice de columna al que llamaremos **`columnaInicial`**.
- Un carácter de relleno al que llamaremos **`caracterRelleno`**.

El método tendrá que realizar la tarea siguiente:

Comenzando en la posición **`[filaInicial][columnaInicial]`** de la matriz, donde se encuentra el carácter que llamaremos **`caracterInicial`**, se sustituirá por el **`caracterRelleno`** todo carácter de la matriz que cumpla las condiciones siguientes:

- Es igual a **`caracterInicial`**.
- Se puede llegar a él desde la posición **`[filaInicial][columnaInicial]`** con movimientos simples del tipo:
    - desplazarse 1 fila arriba
    - desplazarse 1 fila abajo
    - desplazarse 1 columna a la izquierda
    - desplazarse 1 columna a la derecha

atravesando únicamente celdas que contengan el **`caracterInicial`**.

En definitiva, se trata de realizar algo similar a lo que hacen algunos programas de dibujo cuando utilizamos la herramienta de relleno, pero en lugar de rellenar con un color una región de un bitmap, rellenamos con un carácter una región de una matriz.

Para poner a prueba el método se ha de crear un programa que lea de la entrada estándar los datos necesarios para crear la matriz de caracteres original y los parámetros de relleno según las especificaciones siguientes:

- En la primera línea el usuario escribirá dos números enteros, *F* y *C*, que representarán el número de filas y de columnas de la matriz.
- En cada una de las siguientes *F* líneas introducirá una cadena de longitud *C* que contendrá los caracteres que se guardaran en cada una de las *F* filas de la matriz respectivamente.
- En la línea siguiente introducirá el carácter de relleno.
- En las dos últimas filas introducirá el valor para los índices de fila y columna de la posición inicial dentro de la matriz a partir de la cual se ha de realizar el relleno.

Una vez leídos los datos de entrada y creada la matriz, la mostrará por pantalla, invocará al método de relleno con los parámetros correspondientes y finalmente volverá a mostrar la matriz para ver el resultado del relleno.

Ejercicio 20

Crea un programa Java que realice ciertas operaciones con los elementos de una matriz. El programa mostrará una línea de comando donde el usuario podrá introducir comandos siguientes. Cada vez que introduzca un comando, se ejecutará la acción correspondiente y se volverá a mostrar la línea de comando. Los comandos son los siguientes:

- **matriz nxm: valores…**

  crea una matriz de n filas y m columnas con los valores indicados. Por ejemplo, el comando:

  ```
  matriz 2x3: 1 2 3 4 5 6
  ```

  crea la siguiente matriz:

1	2	3
4	5	6

- **intercambia fila1 columna1 fila2 columna2**

  intercambia el valor matriz[fila1][columna1] con el valor matriz[fila1][columna1] y vuelve a mostrar el contenido de la matriz. Por ejemplo:

  ```
  intercambia 0 0 1 1
  ```

  deja la matriz así:

5	2	3
4	1	6

  Si el comando no es válido, no está bien escrita la palabra “intercambia” o los índices están fuera de rango o no hay exactamente cuatro índices o no está creada la matriz, se mostrará el mensaje “comando incorrecto” en lugar de la matriz.

- **fin**

  finaliza el programa