SESIÓN 5: Introducción a los Arrays

Objetivos

- Saber declarar, crear e inicializar arrays (unidimensionales y bidimensionales).
- Saber acceder a componentes individuales de un array.
- Saber realizar operaciones comunes con arrays (mostrar un array, calcular la media, varianza, desviación típica, etc.).
- Saber realizar operaciones más complejas sobre arrays.
- Introducir al alumno en el uso de matrices binarias (sus elementos son o 0s o 1s).

Nota importante: Siga el esquema de nombrado de paquetes que se indicó en la sesión 01 es decir: **org.ip.sesion05**. En ese paquete se crearán todos los programas que se proponen en la sesión dándoles un nombre al programa y que se indica en cada ejercicio entre **paréntesis y en negrita**.
Al final de la sesión, el alumno deberá cargar el trabajo realizado a su repositorio personal indicando la clave correspondiente a la sesión.

Ejercicios propuestos

1. Implementa la clase **(EntradaSalidaArrays)** que contiene los métodos de clase (estáticos) relacionados con arrays (1D) y matrices (2D) que se detallan a continuación:

```
public static double[] leerRealesArray1D()
public static int [] leerEnterosArray1D()
public static int [] inicializarEnterosArray1D()
public static double [] inicializarRealesArray1D()
public static void mostrarArray1D(double [] array)
public static void mostrarArray1D(int [] array)
public static int [][] leerEnterosMatriz2D()
public static int [][] inicializarEnterosMatriz2D()
public static void mostrarMatriz2D(int [][] matriz)
public static void mostrarMatriz2D(double [][] matriz)
```

leerRealesArray1D(): double[] ⇒ Leerá de teclado valores reales que guardará en un array unidimensional de reales.
leerEnterosArray1D(): int[] ⇒ Leerá de teclado valores enteros que guardará en un array unidimensional de enteros.
inicializarEnterosArray1D(): int[] ⇒ Guardará en un array unidimensional valores enteros generados aleatoriamente.
inicializarRealesArray1D(): double[] ⇒ Guardará en un array unidimensional valores reales generados aleatoriamente.
mostrarArray1D(array: double[]):void ⇒ Muestra por pantalla los componentes de un array unidimensional de reales.

mostrarArray1D(array: int[]):void ⇒ Muestra por pantalla los componentes de un array unidimensional de enteros.
leerEnterosMatriz2D(): int[][] ⇒ Leerá de teclado valores enteros que guardará en un array bidimensional de enteros.
inicializarEnterosMatriz2D(): int[][] ⇒ Guardará en un array bidimensional valores enteros generados aleatoriamente.
mostrarMatriz2D(matriz: int[][]):void ⇒ Muestra por pantalla los componentes de un array bidimensional de enteros.
mostrarMatriz2D(matriz: double[][]):void ⇒ Muestra por pantalla los componentes de un array bidimensional de reales.

Por ejemplo:

```
public static int [] leerEnterosArray1D() {
	entrada = new Scanner(System.in);
	System.out.println("Introduce el número de componentes del array de
	enteros");
	int numElementos = entrada.nextInt();
	int [] array = new int [numElementos];
	System.out.println("Introduce valores enteros en el array ");
	for (int i = 0; i < array.length; i++) {
		System.out.print("Introduce valor " + (i + 1) + "=> ");
		array[i] = entrada.nextInt();
	}
	return array;
}
```

Una vez implementados dichos métodos, en la misma clase implementar el método `main` donde se muestre el uso de dichos métodos, mediante su ejecución.

2. Implementa una clase **(EstadisticasArrays)** en el paquete **org.ip.sesion05** de la carpeta de fuentes `src` que contenga los métodos de clase (estáticos) que se detallan a continuación:

```
public static double max(double[] array)
public static int max(int [] array)
public static double min(double[] array)
public static int min(int [] array)
public static double media(double[] array)
public static double media(int [] array)
public static double varianza(double[] array)
public static double varianza(int [] array)
public static double desviacionTipica(double [] array)
public static double desviacionTipica(int [] array)
public static void mostrarArray(double [] array)
public static void mostrarArray(int [] array)
```

Como aclaración de las operaciones estadísticas más importantes, se proporcionan las siguientes descripciones, algunas de ellas con sus respectivas fórmulas

max(array:double[]): double ⇒ valor máximo del array a.
min (array:double[]): double ⇒ valor mínimo del array
media(array:double[]): double ⇒ media de los valores del array:

$$media = \frac{\sum_{i=1}^{n} a_i}{n} = \frac{a_1 + a_2 + \cdots + a_n}{n}$$

var(array:double[]): double ⇒ varianza de los valores del array:

$$varianza = \frac{\sum_{i=1}^{n} (a_i - media)^2}{n - 1}$$

desviacionTipica(array:double[]): double ⇒ desviación estándar del array:

$$stdDev = \sqrt{varianza}$$

Una vez implementados dichos métodos estáticos, en la misma clase se debe implementar el método `main` donde se muestre el uso de dichos métodos y se obtenga la salida que se muestra a continuación.

```
Array de Enteros: [5, 10, 15, 20]
Numero Elementos del array de enteros: 4
Minimo del array de enteros: 5
Maximo del array de enteros: 20
Media del array de enteros: 12.5
Varianza del array de enteros: 41.666666666666664
Desviacion Tipica del array de enteros: 6.454972243679028

Array de Reales: [5.0, 25.5, 15.75, 10.25, 12.5]
Numero Elementos del array de reales: 5
Minimo del array de reales: 5.0
Maximo del array de reales: 25.5
Media del array de reales: 13.8
Varianza del array de reales: 58.106249999999996
Desviacion Tipica del array de reales: 7.622745568363147
```

Trabajo autónomo

3. Implementa una clase **(PracticarConArrays)** que contiene los métodos de clase (estáticos), para practicar con arrays, que se detallan a continuación:

```
public static int [] invertir(int [] array)
public static void desplazar(int [] array)
public static int indiceMaximoValor(int [] array)
public static int indiceMinimoValor(int [] array)
public static int [] eliminarDuplicados(int [] array)
public static void mostrarArray(int [] array)
```

Para el método `eliminarDuplicados` tendrá que crear y utilizar arrays auxiliares para que al final, el array resultante no tenga ningún duplicado, y el tamaño del array que se devuelve (`return`) sea exactamente el número de elementos sin repetición. Una vez implementados todos estos métodos, en la misma clase se debe implementar el método `main` donde se muestre su uso y se obtenga la salida que se muestra a continuación.

```
Array de Enteros: [5, 10, 15, 2, 4, 23, 7, 13, 8, 17, 27]
Indice del menor valor del array de enteros: 3
Indice del mayor valor del array de enteros: 10
Array de Enteros: [27, 17, 8, 13, 7, 23, 4, 2, 15, 10, 5]
Indice del menor valor del array invertido: 7
Indice del mayor valor del array invertido: 0
Desplazando el array una posicion ...
Array de Enteros: [10, 15, 2, 4, 23, 7, 13, 8, 17, 27, 5]
Desplazando el array 5 posiciones ...
Array de Enteros: [7, 13, 8, 17, 27, 5, 10, 15, 2, 4, 23]
Array de Enteros: [1, 1, 3, 2, 1, 6, 3, 4, 4, 4, 5, 2, 1, 2, 6, 4, 7, 5, 6, 7, 1, 8]
Array de Enteros: [1, 3, 2, 6, 4, 5, 7, 8]
```

4. Implementa una clase **(MatrizMagica)** que contiene un método de clase (estático), para comprobar si una matriz de 4 × 4 es *mágica* o no. Una matriz *mágica* es aquella matriz donde la suma de cualquier fila o de cualquier columna vale lo mismo. Los datos de la matriz en el `main` se deberán de introducir por teclado. Las salidas que se muestran a continuación son el resultado de tres ejecuciones independientes.

```
public static boolean esMatrizMagica(int [][] matriz)
```

```
M[0][0]: 16
M[0][1]: 3
M[0][2]: 2
M[0][3]: 13
M[1][0]: 5
M[1][1]: 10
M[1][2]: 11
M[1][3]: 8
M[2][0]: 9
M[2][1]: 6
M[2][2]: 7
M[2][3]: 12
M[3][0]: 4
M[3][1]: 15
M[3][2]: 14
M[3][3]: 1
La matriz es magica
```

```
M[0][0]: 1
M[0][1]: 2
M[0][2]: 3
M[0][3]: 4
M[1][0]: 4
M[1][1]: 3
M[1][2]: 2
M[1][3]: 1
M[2][0]: 1
M[2][1]: 2
M[2][2]: 3
M[2][3]: 4
M[3][0]: 4
M[3][1]: 3
M[3][2]: 2
M[3][3]: 1
La matriz es magica
```

```
M[0][0]: 1
M[0][1]: 2
M[0][2]: 3
M[0][3]: 4
M[1][0]: 1
M[1][1]: 2
M[1][2]: 3
M[1][3]: 4
M[2][0]: 1
M[2][1]: 2
M[2][2]: 3
M[2][3]: 4
M[3][0]: 1
M[3][1]: 2
M[3][2]: 3
M[3][3]: 4
La matriz no es magica
```

5. Diseña una nueva clase **(MatricesBinarias)** que contenga los métodos de clase (estáticos), relacionados con matrices binarias, que se detallan a continuación. Recordamos que una *matriz binaria* es aquella que los elementos son únicamente 0's o 1's.

```
public static boolean esBinaria(int [][] matriz)
public static int [][] transformaBinaria(int [][] matriz)
public static int numeroDeUnos(int [][] matriz)
public static boolean tieneNumeroParCeros(int [][] matriz)
public static int [] sumaFilas(int [][] matriz)
public static int [] sumaColumnas(int [][] matriz)
public static int [] extraerFila(int [][] matriz, int numFila)
public static int [] extraerColumna(int [][] matriz, int numColumna)
public static int [][] obtenerLaterales(int [][] matriz)
public static void mostrarMatriz(int [][] matriz)
public static void mostrarArray(int [] array)
```

esBinaria(matriz: int[][]): boolean ⇒ Devuelve verdadero si la matriz que pasamos como parámetro es binaria y falso en caso contrario.
transformaBinaria(matriz: int[][]): int[][] ⇒ Devuelve una nueva matriz en la que cada elemento será 0 o 1 dependiendo del resultado de aplicar el operador módulo sobre dicho elemento de la matriz (es decir, `matriz[i][j] % 2`).
numeroDeUnos(matriz: int[][]): int ⇒ Devuelve un entero que corresponde con el número de 1's que hay en la matriz binaria. Si la matriz no es binaria que devuelva -1.
tieneNumeroParCeros(matriz: int[][]): boolean ⇒ Devuelve verdadero si la matriz que pasamos como parámetro tiene un número par de 0's y falso en caso contrario.
sumaFilas(matriz: int[][]): int[] ⇒ Devuelve un array con la suma de los valores de cada fila.
sumaColumnas(matriz: int[][]): int[] ⇒ Devuelve un array de enteros con la suma de los valores de la cada columna.
extraeFila(matriz: int[][], int): int[] ⇒ Devuelve un array de enteros que coincide con la fila de la matriz que se le pasa como parámetro.
extraeColumna(matriz: int[][], int): int[] ⇒ Devuelve un array de enteros que coincide con la columna de la matriz que se le pasa como parámetro.
obtenerLaterales(matriz: int[][]): int[][] ⇒ Devuelve una matriz (con cuatro filas) con los valores de los cuatro laterales de la matriz que se le pasa como parámetro. Debe tener en cuenta que en la matriz que se devuelve, la `fila 0` será para la *lateral superior* (fila superior), la `fila 1` para la *lateral derecha* (columna de la derecha), la `fila 2` para la *lateral inferior* (fila inferior) y la `fila 3` para la *lateral izquierda* (columna izquierda).
mostrarMatriz(matriz: int[][]): void ⇒ Muestra la matriz *matriz* que se pasa como parámetro.
mostrarArray(array: int[]): void ⇒ Muestra el array *array* que se pasa como parámetro.

Una vez implementados todos estos métodos, en la misma clase se debe implementar el método `main` donde se muestre su uso y se obtenga la salida que se muestra a continuación.

```
Matriz:
[1      3       5       7       9
12      8       37      50      11
2       4       6       8       10
77      12      21      31      28
47      2       84      87      63]
La matriz no es binaria
Fila 2 => Array: [2     4       6       8       10]
Columna 3 => Array: [7  50      8       31      87]
Suma de las filas => Array: [25 118     30      169     283]
Suma de las columnas => Array: [139     29      153     183     121]
Transformando la matriz de enteros en matriz binaria ...
Matriz:
[1      1       1       1       1
0       0       1       0       1
0       0       0       0       0
1       0       1       1       0
1       0       0       1       1]
La matriz es binaria
El numero de 1s es: 13
La matriz binaria tiene un numero par de ceros
Fila 3 => Array: [1     0       1       1       0]
Columna 2 => Array: [1  1       0       1       0]
Suma de las filas => Array: [5  2       0       3       3]
Suma de las columnas => Array: [3       1       3       3       3]
Laterales => Matriz:
[1      1       1       1       1
1       1       0       0       1
1       0       0       1       1
1       0       0       1       1]
```