ESTRUCTURAS DE DATOS. STRINGS Y ARRAYS.

1. CADENAS, STRING.

Hasta ahora hemos definido y usado cadenas de forma simple pero vamos a ver algunos métodos y operaciones muy útiles a la hora de trabajar con ellas.

Recordemos para crear y definir cadenas haríamos:

```
String cadena = "Hola Mundo";
```

También se pueden crear objetos String como se haría con cualquier otro objeto Java: utilizando new.

```
String cadena = new String("Hola Mundo.");
```

1.1 CONCATENACIÓN DE CADENAS.

Java permite concatenar cadenas fácilmente utilizando el operador +. El siguiente fragmento de código concatena tres cadenas:

```
"La entrada tiene " + contador + " caracteres.";
```

1.2 LONGITUD DE CADENAS.

Para saber la longitud de una cadena usamos el método .length(). Este devuelve un entero con la longitud total de la cadena. Por ejemplo:

```
String cadena = "HOLA";
int n = cadena.length();
```

n vale 4.

Se puede usar directamente con la cadena:

```
n= "abc".length();
```

n vale 3.

1.3 EXTRACCIÓN DE CARACTERES.

Para extraer un carácter de una cadena se usa el método .charAt(indice). Hay que tener en cuenta que el índice empieza en 0.

Por ejemplo:

```
char caracter = "hola".charAt(2);
```

caracter es igual a 'l'.

1.4 COMPARACIÓN DE CADENAS.

Para comparar dos cadenas se usa el método **cadena1.equals(cadena2)**. Este método devolverá True en el caso de que todos los caracteres sean iguales y false en caso contrario. Hay que tener en cuenta que las mayúsculas y minúsculas´son considerados caracteres diferentes así:

```
String x= "hola";
String y= "HOLA";

x.equals(y);    // devolverá FALSE.
```

Si queremos obviar si es mayúscula o minúscula se usa el método **cadena1.equalsIgnoreCase(cadena2).**

```
x.equalsIgnoreCase(y); //devolverá TRUE.
```

1.5 BÚSQUEDA EN LA CADENA.

Para buscar en la cadena tenemos los siguientes métodos:

- **int indexOf(char ch, int start)**: Devuelve el índice correspondiente a la primera aparición del carácter ch en la cadena, comenzando a buscar desde el carácter start (si no se especifica se busca desde el principio).
- **int indexOf(String str)**: Devuelve el índice correspondiente al carácter en que empieza la primera aparición de la subcadena str.
- **int lastIndexOf(char ch, int start):** Devuelve el índice correspondiente a la última aparición del carácter ch en la cadena, comenzando a buscar desde el carácter start (si no se especifica se busca desde el final).
- **int lastIndexOf(String str):** Devuelve el índice correspondiente al carácter en que empieza la última aparición de la subcadena str.

Si no se encuentra devuelve -1.

1.6 CONVERTIR DE MAYÚSCULAS A MINÚSCULAS y VICEVERSA.

Para convertir de mayúsculas a minúsculas se usa el método **.toLowerCase()** y para convertir de minúsculas a mayúsculas se usa el **.toUpperCase().**

Ejemplos:

```
String x = "hola";
String y;
y = x.toUpperCase();
```

y es “HOLA”.

```
x = "HoLa";
y = x.toLowerCase();
```

y es “hola”.

1.7 EXTRAER UNA CADENA DE OTRA.

Para extraer una parte de la cadena usaremos el método **.substring (int beginIndex, int endIndex)** que sacará la cadena empezando en el índice

beginIndex y terminando en el índice endIndex-1, es decir, el índice especificado como último no se extrae.

Si queremos extraer hasta el final no especificamos el índice final **.substring (int beginIndex)** . En ambos casos, este método devuelve un String.

```
String x = "hola mundo";
String y;
String z;
y = x.substring(0, 4);
y = z.substring(5);

y es "hola" (sin el espacio)
z es "mundo"
```

2. ARRAYS (UNIDIMENSIONALES).

Una *array* **es una colección de variables del mismo tipo**, a la que se hace referencia por un nombre común. En Java, los arrays pueden tener una o más dimensiones.

Los arrays unidimensionales son una lista de datos de un mismo tipo. El Array tendrá un tamaño n y sus índices irán de 0 a n-1. ¡OJO! Los tamaños de los arrays son fijos y no se pueden variar. La declaración de un array en Java se puede realizar de las siguientes manera:

```
tipo_dato nombre_array[];
```

o

```
tipo_dato[] nombre_array;
```

Una vez declarado hay que crearlo como tal e inicializarlo indicado su tamaño:

```
nombre_array = new tipo_dato[tamaño];
```

Podemos hacer ambas cosas en una misma sentencia, como con los datos. Por ejemplo:

```
int edades[] = new int[10];
char[] cadenacaract = new char [20];
```

El array puede inicializarse desde el principio si se conocen todos los datos usando la siguiente estructura:

```
tipo_dato array[] = {elemento0,elemento1,...,elementoN-1};
```

Por ejemplo:

```
char cadenacaract[] = {'a','b','c','d','e'};
```

o

```
char cadenacaract[] =new char [] {'a','b','c','d','e'};
```

Si definimos los valores de la cadena no hace falta definir su tamaño, es más, si hacemos ambas cosas Netbeans dará error.

También se puede insertar valor a valor del array poniendo su índice entre los corchetes,por ejemplo:

```
cadenacaract[3] = 'z';
```

Se puede hacer arrays también de String, por ejemplo:

```
public class Array {
    public static void main(String[] args) {
        String arraystring[]={"hola","adios"};
        System.out.println(arraystring[0]);
        System.out.println(arraystring[1]);
    }
}
```

Este código sacará por pantalla:

hola

adios

2.1 MÉTODOS CON ARRAYS.

En Java, cualquier array es **una instancia de una clase implícita** generada por la JVM. Esta clase implícita extiende directamente de java.lang.Object. Al ser un objeto, tiene algunas características como acceder al atributo **length** para obtener su tamaño. Por ejemplo:

```
int[] numeros = new int[5];
System.out.println(numeros.length); // Muestra: 5
```

Aunque los arrays son objetos, no tienen métodos utilitarios propios, como el ordenamiento o la búsqueda. Sino que para ello podemos hacer uso de la clase

Arrays del paquete java.util. Esta clase proporciona métodos estáticos para manipular arrays, como ordenarlos, buscarlos, copiarlos o convertirlos en listas.

Algunos de los métodos disponibles son los vistos en la tabla 1.

MÉTODO	DESCRIPCIÓN	DATO DEVUELTO
.copyOf(T[] original, int newLength)	Copia un array y lo devuelve en un nuevo array.	array del mismo tipo que se introduce
.copyOfRange(T[] original, int from, int to)	Copia un array y lo devuelve en un nuevo array. Le indicamos la posición de origen (incl) y de fin(no incl).	array del mismo tipo que se introduce
.equals(T[] a, T[] b)	Indica si dos arrays son iguales.	true o false
.fill(T[] a, T val)	Rellena un array con un valor que le indiquemos como parámetro.	
.sort(T[] a)	Ordena el array.	
.toString(T[] a)	Pasa el contenido del array unidimensional pasado como parámetro a una cadena.	Devuelve una cadena con el contenido del array.

Tabla 1.

Veamos un ejemplo donde creamos un array y usamos distintos métodos:

```
public class ArraysDemo {
    public static void main(String[] args) {
        // 1. Crear un array inicial
            int[] numeros = {5, 8, 2, 9, 3};
            // Imprimir el array original
            System.out.println("Array original: " +
Arrays.toString(numeros));


        // 2. Copiar todo el array
            int[] copiaNumeros = Arrays.copyOf(numeros, numeros.length);
            System.out.println("Copia completa del array: " +
      Arrays.toString(copiaNumeros));


        // 3. Copiar un rango (del índice 1 al 3, excluye el índice 3)
        int[] subArray = Arrays.copyOfRange(numeros, 1, 3);
        System.out.println("Subarray copiado (índices 1 a 3): " +
Arrays.toString(subArray));
```

```
        // 4. Rellenar el array con un valor específico
        int[] rellenado = new int[5];
        Arrays.fill(rellenado, 7);
        System.out.println("Array rellenado con 7: " +
Arrays.toString(rellenado));

        // 5. Ordenar el array original
        Arrays.sort(numeros);
        System.out.println("Array ordenado: " +
Arrays.toString(numeros));

        // 6. Comparar dos arrays
        int[] otroArray = {2, 3, 5, 8, 9}; // Array ya ordenado para la
comparación
        boolean sonIguales = Arrays.equals(numeros, otroArray);
        System.out.println("¿Son iguales los arrays?: " + sonIguales);
    }

}
```

3. MATRICES O ARRAYS DE DOS DIMENSIONES.

Las matrices son **arrays de arrays**, donde cada elemento del array contiene la referencia de otro array. Se crea una matriz multidimensional al agregar un conjunto de corchetes ([]) por dimensión.

```
Tipo_de_variable[ ][ ]… [ ]  nombre = new
Tipo_de_variable[dim1][dim2]… [dimN];
```

Ejemplos:

```
int[][] array = new int[4][3];      //un array 2D o matriz
int[][][] array2 = new int[4][2][3];   //un array 3D,
```

Lo más común es trabajar con matrices bidimensionales, es decir, con filas y columnas.

```
int[][]  matriz = new int[3][2]; //es una matriz de 3x2, 3
filas, 2 columnas.
```

Para introducir los valores podemos cargarlos uno a uno:

```
matriz[2][0] = 5; //2 indica el número de fila y 0 la columna.
```

También se pueden cargar directamente los elementos, durante la declaración de la matriz de la siguiente manera:

```
int[][]       numeros    =          {{1,2},{3,4},{5,6}};
```

Matrices Irregulares: Cabe destacar que las filas (arrays) de las matrices no tienen porqué tener el mismo tamaño, en este caso definimos el número de filas y luego a cada fila le asignamos un tamaño, por ejemplo:

```
int [][]numeros = new int [2][];
numeros[0]=new int [] {1,2,3};
numeros[1]=new int [] {0,-1,2,3, -5};
```

o

```
int [][]m = new int [3][];
m[0] = new int[3];
m[1] = new int[5];
m[2] = new int[2];
```

Al final tener un array de dos dimensiones es tener un array donde sus elementos son otros arrays. Por tanto, si queremos conocer el número de elementos del array, es decir, el número de filas de una matriz usamos nombre.length. y para conocer el número de columnas tendríamos que seleccionar el array que hay en una de las filas, nombre[i], y ver su tamaño con nombre[i].length

En el ejemplo anterior m.length es 3 y m[1].length es 5.

Veamos un ejemplo de esto.

```
package array;
class Array{
    public static void main(String args[])
    {
        // declarar e inicializar array 2D
        int arr[][] = { {2,7,9},{3,6},{7,4,2} };

        // imprimir array 2D
  // para cada una de las líneas.
        for (int i=0; i< arr.length ; i++)
        {
  //imprimimos cada una de las columnas
            for (int j=0; j < arr[i].length ; j++){
                System.out.print(arr[i][j] + " ");
            }
          /*al terminar de imprimir todas las columnas de
          una       fila, imprimimos en una nueva línea */
           System.out.println();


        }
    }
```

```
}
```

Este programa sacará por pantalla:

2 7 9
3 6
7 4 2

3.1 MÉTODOS CON ARRAYS BIDIMENSIONALES.

MÉTODO	DESCRIPCIÓN	DATO DEVUELTO
.deepEquals(matriz1, matriz2)	Indica si dos arrays son iguales.	true o false
.deepToString(matriz)	Pasa el contenido del array bidimensional pasado como parámetro a una cadena.	Devuelve una cadena con el contenido del array.

Tabla 2.