ÉRICA PALOMINO

TEMA 3

ESTRUCTURAS DE DATOS

erica.palomino@escuelaartegranada.com

ESCUELAARTEGRANADA

01 ARRAYS

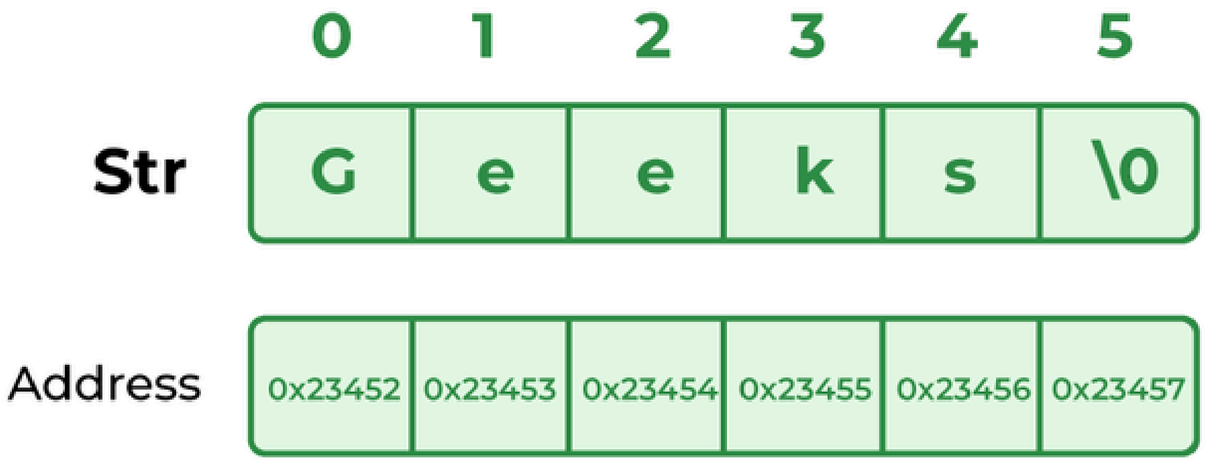
02 ARRAYS Y BUCLES

03 MATRICES

04 LISTAS

ARRAYS

- En Java existen unas estructuras concretas que nos permiten almacenar **múltiples valores** en **una sola variable**.
- Almacenan valores del mismo **tipo de dato**.



ARRAYS

- Los arrays se declaran indicando el **tipo de dato** que van a almacenar junto con **corchetes [ ]**.
- En cuanto al nombre, se sigue la **misma convención** que con las **variables**.

```
String[] cadenaStrings;
```

ARRAYS

- Para insertar valores en los arrays, se usa una **lista** con los elementos **separados por coma**, y acotada por **corchetes { }**

```
String[] cadenaStrings = {"Hello", "world", "!"};
```

```
int[] numeros = {1 , 45, 32, 5, -15};
```

ARRAYS

- Par acceder a valores concretos dentro del Array, se utiliza su posición dentro de la lista, es decir, su **índice**.
- En Java, se empieza a **contar siempre desde 0.**

```
String[] cadenaStrings = {"Hello", "world", "!"};
int[] numeros = {1 , 45, 32, 5, -15};
System.out.println(cadenaStrings[0] + " "+ numeros[2]);
//imprime Hello 32
```

ARRAYS

- De igual forma, podemos utilizar esta forma de acceder a las posiciones concretas para **modificar o añadir valores** a la lista.

```
int[] numeros = {1 , 45, 32, 5, -15};
numeros[2] = -52;
System.out.println(numeros[2]);
//imprime -52 en ligar de 32
```

ARRAYS

- En Java, los arrays son de **tamaño estático**, por lo que no se puede añadir más cantidad de valores que con el que se inicializó.
- Una forma de paliar esto es **sobredimensionándolos**.

```
int[] numeros = new int[20];
```

ARRAYS Y BUCLES

- Podemos **recorrer** los elementos de un array mediante los **bucles for**.
- Se especifica cuantas veces se hace el bucle mediante el atributo **length** de los array.

```
int[] numeros = {1 , 45, 32, 5, -15};
for (int i = 0; i < numeros.length; i++){
    System.out.println(numeros[i]); //imprime el elemento i del array
}
```

MATRICES

- También llamados **arrays multidimensionales,** son arrays de arrays.
- Este tipo de estructuras son útiles cuando se quieren guardar datos en forma de tabla. **Ej: Registro de temperaturas a lo largo de una semana.**

MATRICES

- Se inicializan añadiendo **las listas de datos con sus corchetes correspondientes**, dentro de los corchetes principales.

```
int[][] matrix = { {1, 2, 3}, {4, 5, 6} };
int[][]matrizVacia = new int[3][4];
```

MATRICES

- Para acceder a los datos de la matriz, se necesitan **2 índices**. Uno para seleccionar el sub-array y el otro para seleccionar el dato en sí.

```
int[][] matrix = { {1, 2, 3}, {4, 5, 6} };
System.out.print(matrix[1][0]); //Saca por pantalla el valor 4

matrix[0][2] = 7; //Sustituye el 3 por 7
```

MATRICES

- Para recorrer una matriz, necesitamos usar **2 bucles *for* anidados** ya que tenemos que llevar la cuenta de 2 índices a la vez.
- **Apunte: matriz.length** devuelve cuantos sub-arrays hay incluidos en la matriz, mientras que **matriz[i].length** devuelve los espacios del sub-array i

```
int[][] matrix = { {1, 2, 3}, {4, 5, 6} };
for(int i = 0; i < matrix.length; i++){
    for(int j = 0; j < matrix[i].length; j++){
        System.out.println(matrix[i][j]);
    }
}
```