



Argentina  
programa  
4.0

# Arrays y Strings

Sintaxis básica

---

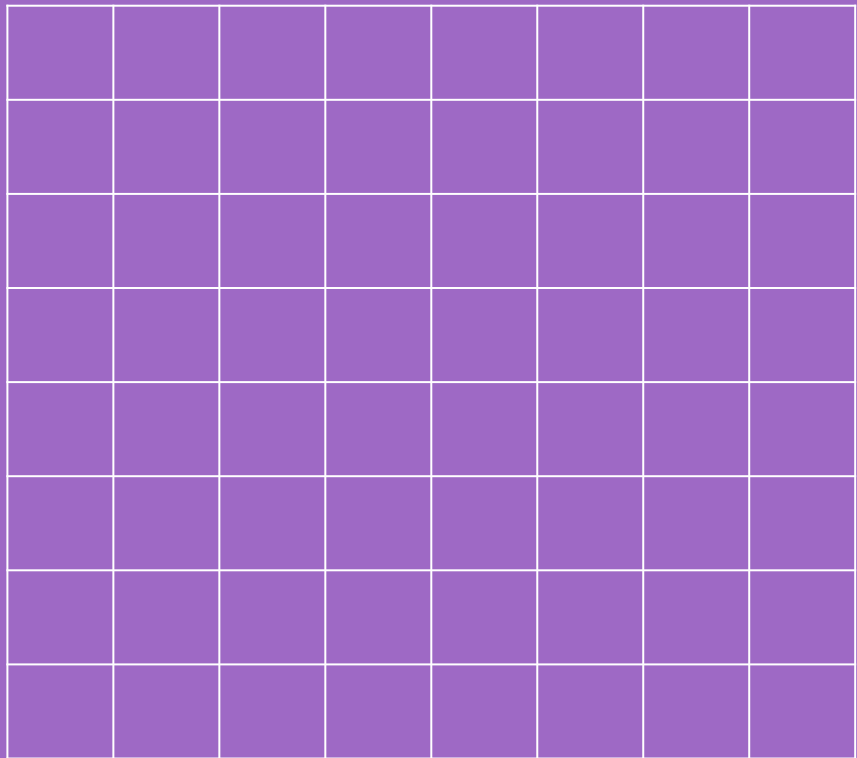
“Desarrollador Java Inicial”

# Agenda

- Arrays
  - Definición
  - Declaración
  - Iteración
- Strings
  - Declaración
  - Uso
  - Métodos principales
- Ejercicios



# Arrays



# Definición

- Es muy importante en todo algoritmo representar un grupo de elementos.
- Se necesitan cuando hay que almacenar y/o procesar un volumen elevado de datos y/o resultados.
- La forma más simple de hacerlo en todo lenguaje son los *Arrays* o “Arreglos” en memoria.
- En particular en Java, los arrays son estructuras de datos que agrupan un conjunto de elementos del mismo tipo:

1
37
16
...

true
true
false
...

'a'
'c'
'b'
...

← Elemento número 0 “cero”

← Elemento número 1

← Elemento número 2

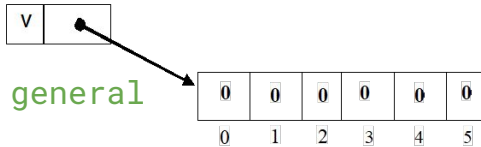
Etc

# Definición

- Tienen un nombre que referencia a la secuencia.
- Cada elemento se accede mediante el nombre y un índice, que indica la posición del elemento.
- Declaración:

```
int v [];
```

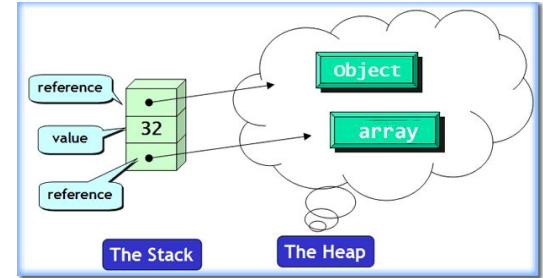
```
tipo variable []; // en general
```



- Creación:

```
v = new int [6];
```

```
variable = new tipo [tamaño]; // en general
```



# Ejemplo de array de números

```
int numeros[] = new int[3]; // Así se declara un arreglo de enteros
```

se declara  
la variable

se crea el  
array

```
numeros[0] = 1; // asignación
```

```
numeros[1] = 37;
```

```
numeros[2] = 16;
```

1
37
16

← Elemento número 0 “cero”

← Elemento número 1

← Elemento número 2

// “length” es una propiedad de los arreglos que nos dice el tamaño del mismo.

```
System.out.println(numeros.length) // imprime por pantalla: 3
```

// para recorrer todo el vector se usa un ciclo con la variable i para el índice

```
for(int i = 0; i < numeros.length; i++){
```

```
    System.out.println(numeros[i]); // imprime por pantalla cada elemento
```

```
}
```

# Ejemplo de array de números

1. Cargar un arreglo de **n números** enteros y luego mostrar los valores cargados.

```
1 package clase3;
2 import java.util.Scanner;
3
4 /** * Cargar un arreglo de n numeros enteros y luego mostrar los valores cargados*/
5 public class Ejercicio3_1 {
6     public static void main(String[] args) {
7         int n, num;
8         // declara el vector
9         int vec[];
10        Scanner sc = new Scanner(System.in);
11        System.out.print("Ingrese el valor de n: ");
12        n = sc.nextInt();
13        // crea el vector de n elementos
14        vec = new int[n];
15        // cargar el vector
16        for (int i = 0; i < vec.length; i++)
17        {
18            //resultados parciales
19            System.out.print("Ingrese el "+ i+ " valor: ");
20            num = sc.nextInt();
21            // guarda el num en el vector en la posición i
22            vec[i] = num;
23        }
24        // mostrar el vector
25        for (int i = 0; i < vec.length; i++)
26        {
27            //resultados parciales
28            System.out.println("Valor "+ i+ " : "+vec[i]);
29        }
30    }
31 }
```

## Output - Clase3 (run)



run:

```
Ingrese el valor de n: 4
Ingrese el 0 valor: 23
Ingrese el 1 valor: 56
Ingrese el 2 valor: 34
Ingrese el 3 valor: 12
```



```
Valor 0 : 23
Valor 1 : 56
Valor 2 : 34
Valor 3 : 12
```

# Java - Sintaxis Básica - Vectores/Arrays básicos

```
char letras[] = {'a', 'b', 'c'};

//solo sirve para recorrer y acceder sin cambiar los valores del vector
for (char letra : letras){
    System.out.println(letra);
}

// a
// b
// c

// solo sirve para recorrer
for (int numero : vec)
{
    System.out.println("Valor: "+numero);
}
```

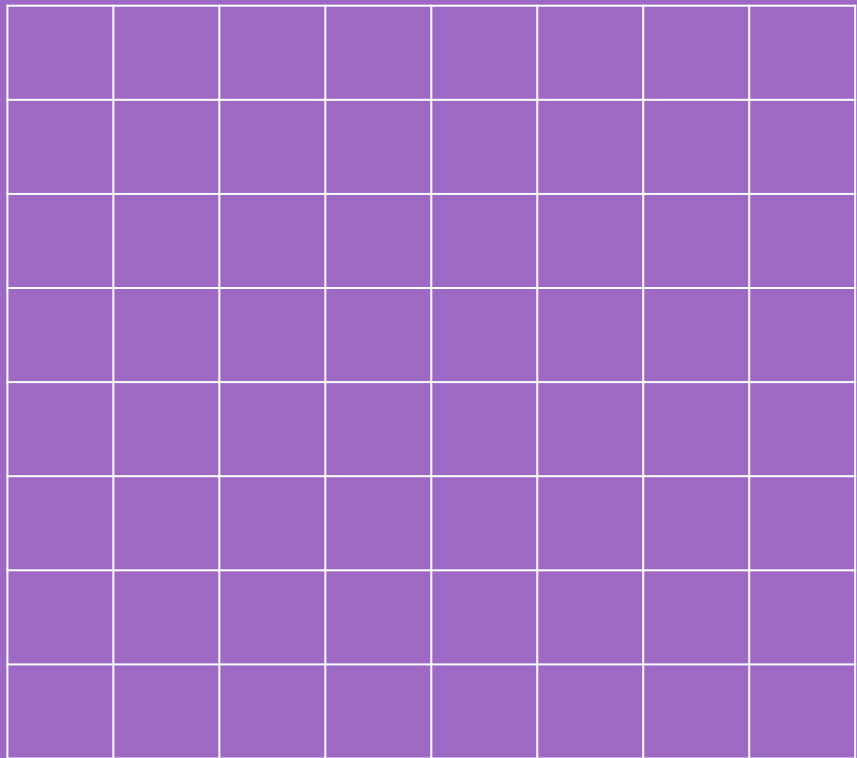


# Java - Vectores/Arrays básicos - Importante

- Siempre están numerados desde cero, pero la propiedad “length” es el tamaño total del vector. Por ejemplo, si hay un array de 5 elementos, la propiedad length es 5, pero el índice para acceder al último elemento es el “4” (ya que el primero es “0”).
- Los arrays son de tamaño fijo. Es decir, si quiero almacenar más elementos, debo crear un nuevo Array
- Para tener un conjunto de elementos variable, vamos a ver colecciones más adelante.



# String



# String

- String es el tipo de dato (Clase en realidad, ya veremos de qué se trata) de Java para representar cadenas de texto, de cualquier tamaño.
- Todos los lenguajes de programación tienen este dato.
- Los Strings se declaran con comillas dobles.

```
String txt1 = "Hola!";
```

```
String txt2 = "Chau!";
```

- Se puede pensar como un array del tipo de dato “char”, aunque como veremos, tiene muchas operaciones particulares. Por ejemplo para acceder a un carácter en particular se puede hacer:

```
txt1.charAt(0); // "H"
```

```
txt1.charAt(2); // "l"
```

# String - Operaciones

- Existen numerosas operaciones que se pueden hacer con los strings, vamos a tomar las variables de ejemplo:

```
String txt1 = "Hola!";  
String txt2 = "Chau!";
```

- Concatenar Strings con "+"

```
String txt3 = txt1 + txt2 ;           // "Hola!Chau!"  
String txt4 = txt1 + " " + txt2 ;    // "Hola! Chau!"
```

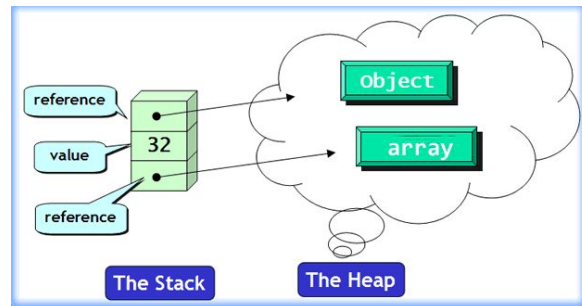
- Comparar

```
txt1.equals(txt2); // false  
txt1.equals(txt1); // true
```

-> **NO** debe usarse el operador == para comparar Strings

- Largo de un String

```
tamaño = txt1.length(); // 5
```



# String - Operaciones

```
String unTexto = "laLA";
```

- Pasar a mayúscula o minúscula

```
textoMinusculas = unTexto.toLowerCase(); // lala  
textoMayusculas = unTexto.toUpperCase(); // LALA
```

- Indicar si contiene otro string

```
unTexto.contains("la"); //true  
unTexto.contains("aL"); //true  
unTexto.contains("La"); //false
```


- Ver si inicia o termina con otra cadena

```
unTexto.startsWith("la"); // true  
unTexto.endsWith("LE"); // false
```

- Reemplazar

```
nuevoTexto = unTexto.replace("la", "le"); // leLA
```

Es importante notar que `toLowerCase` y `replace`, NO cambian el valor de la variable "unTexto", sino que retornan un nuevo String, que puedo guardar en otra variable o imprimir por pantalla. Por ejemplo se pueden encadenar:



```
"hola que tal?".replace("hola", "chau").toUpperCase();  
// CHAU QUE TAL?
```

# String - Operaciones - Split

Una operación muy usada de String, es “split” el cual convierte un String en un String[] (es decir un array de strings), a partir de un separador. Por ejemplo:

```
String saludo = "hola que tal?";
```

```
String[] saludoPartido1 = saludo.split(" ");
```

```
// "hola", "que", "tal?"
```

```
String[] saludoPartido2 = saludo.split("a");
```

```
// "hol", " que t", "l?" --> notar la presencia de espacios
```

# String - Escape y caracteres especiales

Todos los lenguajes de programación cuentan con caracteres “especiales”, ya que hay cosas que o bien no se pueden escribir, coinciden con palabras reservadas o la sintaxis no las soporta. En el caso de Java, el caracter especial es la contrabarra “\” ( la barra que no es la de dividido ). Vamos con unos ejemplos:

- Enter / Cambio de línea

`"hola que tal?\n"` → `\n` es la forma de hacer un cambio de línea, ya que java no permite strings multi línea

- Tabulación (no es lo mismo que varios espacios)

`"Pedro\t18\tprogramador\n"` // `\t` escribe una tabulación

- Escribir una comilla doble: si con “ declaramos un string, como hacemos para que aparezca en el medio del texto que queremos escribir?

`"hola, \"que tal\" "` // con `\` decimos que “escapamos” el caracter

- Escribir una contrabarra “\”

`"Esto es una contra barra \\" "` // con `\` escapamos a la `\` misma, si lo imprimimos esto dice:

Esto es una contrabarra \

# String - Escape y caracteres especiales

Secuencia de Escape	Significado
\'	Comilla Simple ( ' )
\"	Comilla Doble ( " )
\n	Salto de Línea
\\	Barra invertida

## Representación de los caracteres

<b>ascii</b>	Codificación de caracteres, (American Standard Code for Information Interchange), utiliza 8 bits que pueden representar 256 caracteres.
<b>Unicode</b>	Codificación que permite representar más de un millón de caracteres, ya que utiliza 32 bits para su representación.
<b>UTF-8</b>	UTF-8 es un sistema de codificación de longitud variable para Unicode. Esto significa que los caracteres pueden utilizar diferente número de bytes.

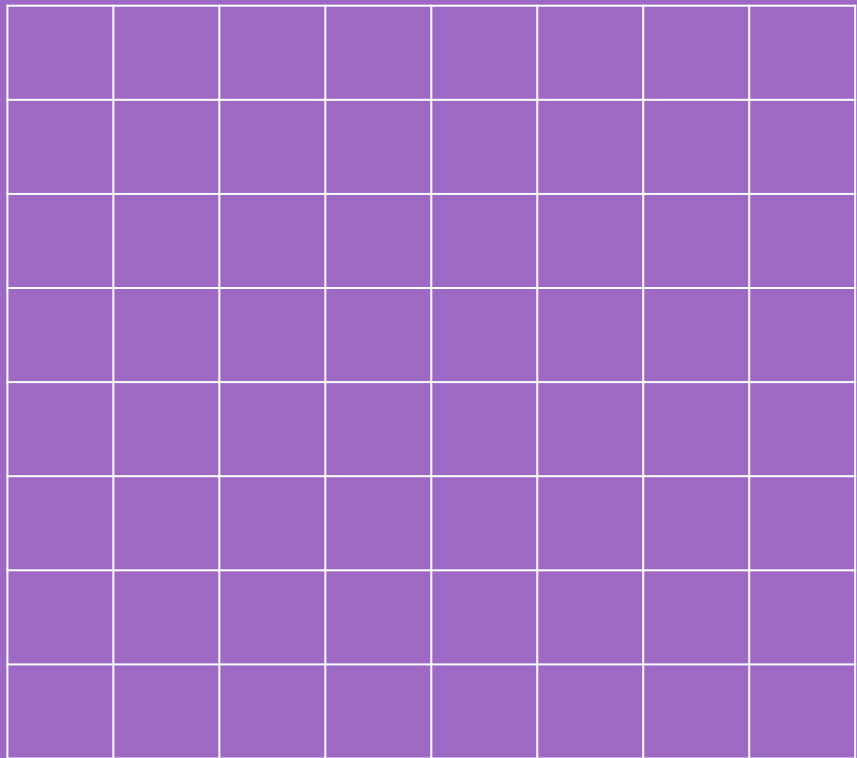


# String - Importante para más adelante

- Para tener una referencia completa de la cantidad de cosas que se pueden hacer con String, puede consultar el [JavaDoc](#) de la misma. Los Javadocs son la forma en la cual se documenta la referencia del código, sobre todo de las clases base de Java.
- No son “mutables” eso quiere decir que no se puede “cambiar” el valor de un string, sino que declarar uno nuevo. Esto en principio no parece muy importante, pero más adelante veremos que hay algunos casos en los que puede importar
- Si bien aún no vimos el concepto de herencia, es importante marcar que no se puede crear una clase heredada del tipo de dato String



# Ejercicios



2. Cargar por teclado una lista de números positivos, de tamaño  $n$  y mostrar:
  1. Los números de la lista.
  2. La sumatoria de los números.
  3. La cantidad de números mayores a 10.
  4. El porcentaje de números pares.
  5. El promedio de todos los números.

# Ejercicios

## 2. Cargar por teclado una lista de números positivos, de tamaño n y mostrar..

```
1 package clase3;
2 import java.util.Scanner;
3
4 /** 2. Cargar por teclado una lista de números positivos, de tamaño n y mostrar:
5 1. Los números de la lista.
6 2. La sumatoria de los números.
7 3. La cantidad de números mayores a 10.
8 4. El porcentaje de números pares.
9 5. El promedio de todos los números.
10 */
11 public class Ejercicio3_2 {
12     public static void main(String[] args) {
13         Scanner sc = new Scanner(System.in);
14         int n, num;
15         int vec[];
16         System.out.print("Ingrese el valor de n: ");
17         n = sc.nextInt();
18         // crea el vector de n elementos
19         vec = new int[n];
20         // cargar el vector
21         for (int i = 0; i < vec.length; i++)
22         {
23             //resultados parciales
24             System.out.print("Ingrese el "+ i+ " valor: ");
25             num = sc.nextInt();
26             // guarda el num en el vector en la posición i
27             vec[i] = num;
28         }
29     }
30 }
```

# Ejercicios

## 2. mostrar... 1. Los números de la lista.

```
30 // 1. mostrar el vector
31 String lista = "";
32 for (int numero : vec)
33 {
34     // guarda los valores en lista
35     lista += "\n"+numero;
36 }
37 System.out.println("Los valores del vector son: "+lista);
```

## 2. mostrar... 2. La sumatoria de los números.

```
39 // 2. La sumatoria de los números.
40 int suma = 0;
41 for (int numero : vec)
42 {
43     suma = suma + numero;
44 }
45 System.out.println("La suma es: "+suma);
```

# Ejercicios

2. mostrar... 3. La cantidad de números mayores a 10.

```
47 // 3. La cantidad de números mayores a 10.
48 int sumaMayor10 = 0;
49 for (int numero : vec)
50 {
51     if (numero > 10)
52         sumaMayor10++;
53 }
54 System.out.println("La cantidad valores mayores a 10 es: "+sumaMayor10);
```

# Ejercicios

## 2. mostrar... 4. El porcentaje de números pares.

```
56 // 4. El porcentaje de números pares.
57 int total = 0, parcial= 0; //porcentaje = parcial * 100 / total
58 // el total también se puede inicializar como total = vec.length
59 float porcentaje = 0;
60 for (int numero : vec)
61 {
62     total++; //cuento todos
63     if ((numero % 2) == 0) //es par?
64         parcial++; // solo cuento los pares
65 }
66 // calculo el porcentaje
67 if (total != 0)
68     porcentaje = parcial * 100 / total;
69 System.out.println("El porcentaje de valores pares es: "+porcentaje);
70
```

## 2. mostrar... 5. El promedio de todos los números.

```
71 // 5. El promedio de todos los números.
72 // promedio = sumatoria / cantidad
73 // usamos la variable suma del punto 2 y el total de elementos del porcentaje
74 float promedio = 0;
75 if (total != 0)
76     promedio = suma / total;
77 System.out.println("El promedio de valores es: "+promedio);
```



**Argentina  
programa  
4.0**

# Gracias!

---