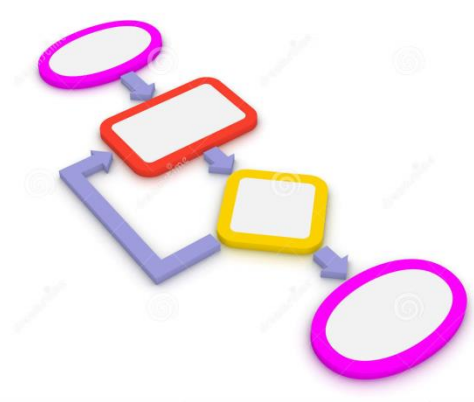
Sesión 3

Java
ORACLE

Concepto de algoritmo

- Los ordenadores son capaces de resolver problemas muy complejos pero…
- *¡No es capaz de resolver el problema por sí solo!*
- ***Es necesario que una persona analice el problema y diseñe un programa.***
- *El programa se ejecutará en el ordenador y solucionará el problema.*



Concepto de algoritmo

- ***Algoritmo*** = secuencia ordenada de pasos que conducen a la solución de un problema concreto
- Son independientes del lenguaje de programación en que se expresan y del ordenador que los ejecuta.

Ejemplo: preparar un sopa instántanea en el microondas

1. Inicio
2. Destapar la sopa
3. Agregar una taza pequeña con agua a la sopa
4. Introducir al horno de microondas
5. Programar el horno de microondas por 3 minutos
6. Sacar del horno
7. Fin

Concepto de algoritmo

- ***Etapas de un algoritmo***

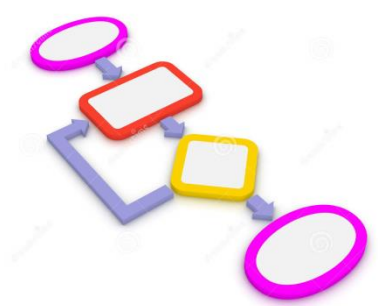
- ✓ ***Entrada:*** información dada al algoritmo.
- ✓ ***Proceso***: cálculos necesarios para encontrar la solución del problema.
- ✓ ***Salida***: resultados finales de los cálculos.

Concepto de algoritmo

- ***Ejemplo: algoritmo que sume 2 números***

1. Pedir el primer valor.
2. Pedir el segundo valor.
3. Realizar la suma.
4. Mostrar el resultado.

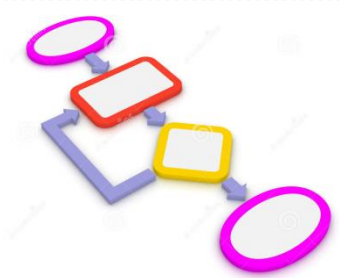
Concepto de algoritmo



Diseño de algoritmos:

- Un ordenador no tiene la capacidad de pensar y resolver el problema solo.
- Una vez que el problema queda claro tenemos que buscar una secuencia de pasos que lo resuelvan e indiquen al ordenador qué instrucciones tiene que ejecutar, es decir, **debemos encontrar un algoritmo**.

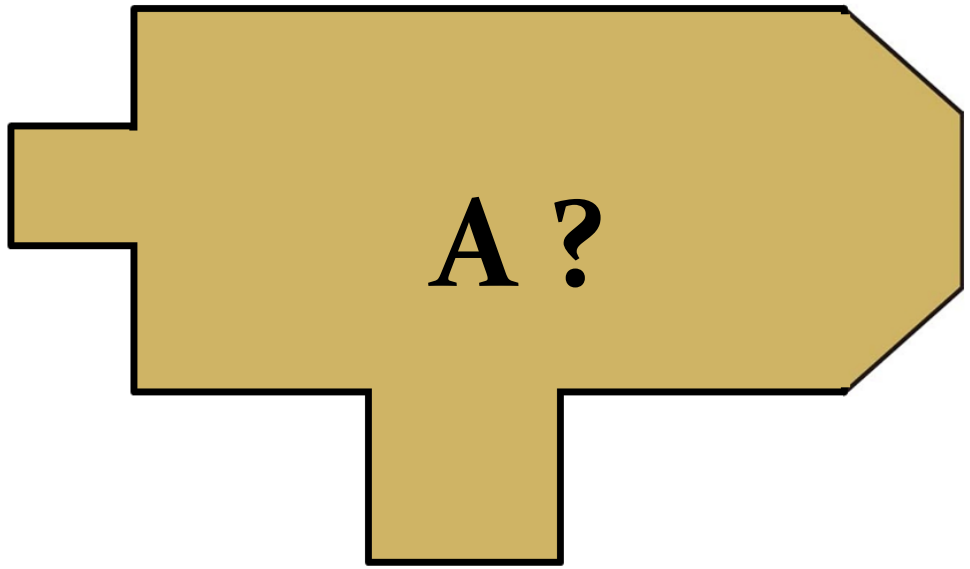
Concepto de algoritmo



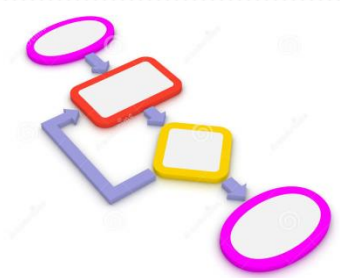
Algunas estrategias para diseñar un algoritmo:

1. ***"Divide y vencerás" o partición:***
   divide el problema en problemas más pequeños que sí puedes respolver.

Ejemplo:
Calcula el área de esta figura



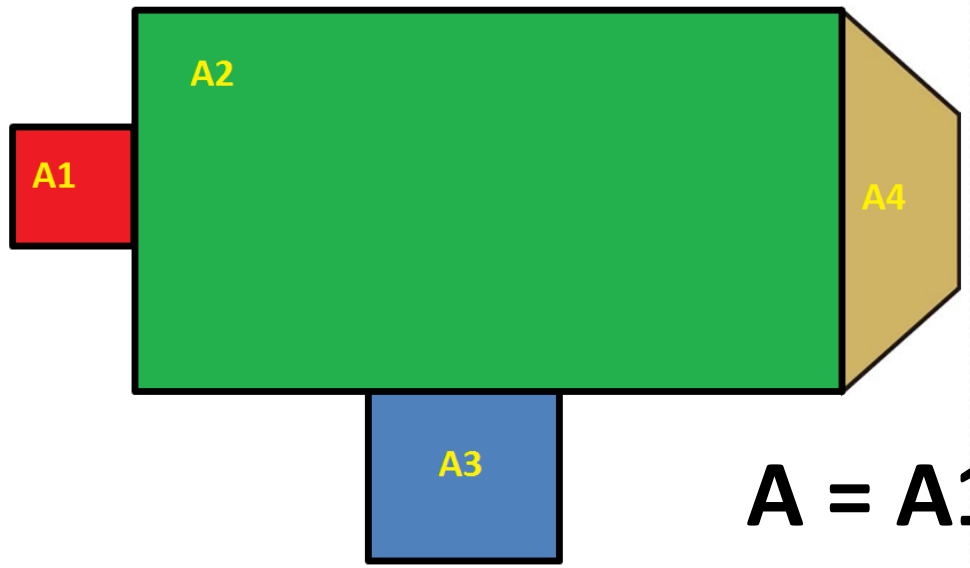
Concepto de algoritmo



Algunas estrategias para diseñar un algoritmo:

1. "Divide y vencerás" o partición:

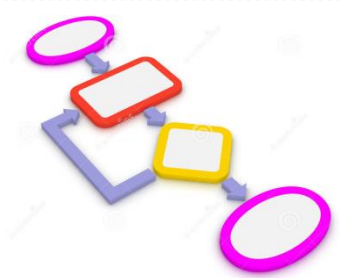
divide el problema en problemas más pequeños que sí puedes respolver.



Solución

A = A1 + A2 + A3 + A4

Concepto de algoritmo

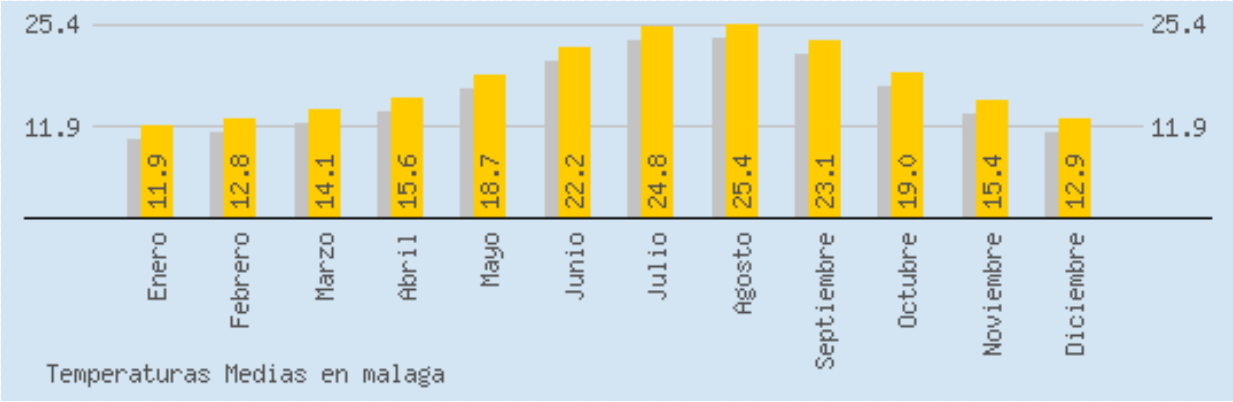


Algunas estrategias para diseñar un algoritmo:

2. Resolver por analogía o similitud:

usar un problema similar ya resuelto para solucionar el nuevo problema.

Ejemplo: Sabemos calcular la notas media de un alumno/a en una asignatura. ¿Cúal es la temperatura media anual en Málaga si conozco la Tª. de cada mes?



Secuencias de escape

- Se usan para representar acciones como salto de línea, tabulador… etc. o algunos caracteres (“, ‘, \).
- Está formado por \ + un carácter:

Secuencia de escape	*Significado*
\b	Retroceso
\n	Salto de línea
\t	Tabulación horizontal
\\	Barra invertida \
\’	Comilla simple
\”	Comilla doble

Secuencias de escape

- ***Ejemplo de uso 1***. Para mostrar por pantalla :

Ejemplo de "secuencias de escape"

en Java: veamos cómo se escribe

Escribiría**:**

System.out.println("Ejemplo de \"secuencias de escape\"\nen Java: veamos cómo se escribe");

Secuencias de escape

- ***Ejemplo de uso 2***. Para mostrar por pantalla :

HTML *“Mañana”*

Java *“Tarde”*

Escribiría**:**

System.out.println("HTML \t\"Mañana\"\nJava \t\" Tarde\"\n");

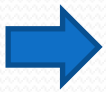
- ***Ejemplo de uso 3.***

```
String sPath = "";
String sDir  = "";
String sFile = "MyFile.txt";

// get file separator: \ in Windows , / in Linux, Unix, Mac...
String OS        = System.getProperty("os.name").toLowerCase();
String separator = System.getProperty("file.separator");
System.out.println("OS:" + OS + ", separator:" + separator);

// windows os
if (OS.indexOf("win") >= 0) {
    sDir = "c:\\home\\users\\javakiller";
// linux, mac os...
} else {
    sDir = "/home/users/javakiller/";
}

sPath = sDir + separator + sFile;
// sPath = sDir + File.separator + sFile; // import java.io.File!
System.out.println("sPath=" + sPath);
```

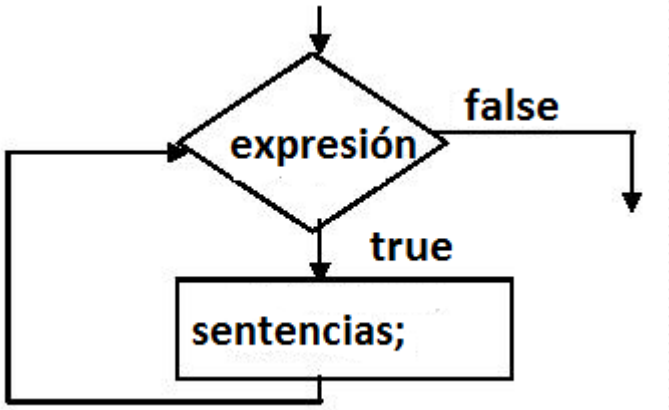
OUTPUT (consola) 

```
OS:windows 7, separator:\
sPath=c:\home\users\javakiller\MyFile.txt
```

Bucles … o cómo repetir tareas

- Usados para repetir una acción varias veces.
- En Java hay 3 bucles: *while, do-while, for y for mejorado*
- **Bucle while**:

```
1   while(expresion_booleana)
2   {
3       //Bloque de código
4   }
```



✓ Mientras expresion_booleana sea true se ejecutan las instrucciones dentro del bucle

Bucles … o cómo repetir tareas

- ***Ejemplos de bucles while (1):***

```
int n = 0;
while ( n > 0 ) System.out.println("Esto nunca lo verás");
```

```
boolean prueba = true;
while ( prueba ) {
      System.out.println("Esto lo verás una vez");
      prueba = false;
}
```

```
boolean prueba = true;
while ( prueba ) {
      System.out.println("Esto lo verás muchas veces");
}
```

Bucles … o cómo repetir tareas

- ***Ejemplos de bucles while (2):***

```
// Loop that prints from 5 to 1 ?
int n = 5;

// op1
while (n>0)
    System.out.println("n:" + n);
    n--;

// op2
while (n>0)
    System.out.println("n:" + n);
n--;
```

```
// op3
while (n>0) System.out.println("n:" + n);
n--;

// op4
while (n>0) {
    System.out.println("n:" + n);
    n--;
}
```

Bucles … o cómo repetir tareas

- ***Ejemplo de bucles while:*** programa que muestra por pantalla tres líneas de caracteres

```
1  public class Cuadrado {
2      public static void main(String[] args) {
3          int contador = 1;
4          while ( contador <= 3 ) {
5              System.out.println("***");
6              contador++;
7          }
8      }
9  }
```

Bucles … o cómo repetir tareas

- ***Ejemplo de bucles while:*** programa que muestra por pantalla tres líneas de caracteres

```
public class Cuadrado {
    public static void main(String[] args) {
        int contador = 3;
        while ( contador > 0 ) {
            System.out.println("***");
            contador--;
        }
    }
}
```

Bucles … o cómo repetir tareas

- **Bucle do-while**:

```
do {
    // instrucción o bloque;
} while ( expresion_booleana);
```

- ✓ Repite el bucle mientras expresion_booleana sea ***true***.
- ✓ Como mínimo el cuerpo del bucle se ejecuta 1 vez.

Bucles … o cómo repetir tareas

- **Bucle for**:

```
for (inicializacion; expresion_booleana; incremento) {
    // instrucciones dentro del cuerpo del bucle
}
```

- ✓ **inicializacion**: sólo se ejecuta la primera vez y sólo una vez. Las variables declaras en esta zona sólo viven dentro del bucle for.
- ✓ **expresion_booleana**: antes de ejecutarse el cuerpo del bucle se pregunta si es true o false: si es **true** se ejecuta el cuerpo del bucle.
- ✓ **incremento**: una vez ejecutado el cuerpo del bucle, se ejecuta esta parte donde se actualizan las variables de control del bucle.

Bucles … o cómo repetir tareas

- **Bucle for**:

```
for (inicializacion; expresion_booleana; incremento) {
    // instrucciones dentro del cuerpo del bucle
}
```

✓ ***Ejemplo 1***:

```
for (int i = 0; i < 10; i++) {
    System.out.println("Valor de i =" + i);
}
System.out.println(i); // ERROR - i fuera de su ámbito
```

Bucles … o cómo repetir tareas

- **Bucle for**:

```
for (inicializacion; expresion_booleana; incremento) {
    // instrucciones dentro del cuerpo del bucle
}
```

✓ **Ejemplo 2: ¿Qué salida produce este código?**

```
for (int x = 10; x <= 25; x++) {
  System.out.println("Valor de x*x =" + x*x);
}
```

break y continue

- ***break***: se usa para forzar la salida de un switch, bucle o un bloque de código.

Ejemplo:

```
for(int j = 5; j<10; j++){
    if (j==6) break;
    System.out.println("Valor de j=" + j);
 }
```

break y continue

- **continue**:
  salta a la siguiente la iteración de un bucle.

Ejemplo:

```
int sumaImpares = 0;
for(int i=1; i<=100; i++)  {
    if (i%2==0) continue;
    sumaImpares  += i;
 }
System.out.println("Suma impares=" + sumaImpares);
```

break y continue con etiquetas

- La **etiqueta** debe ir colocada antes del bucle seguida de “:”
- Para usarla:

  break [etiqueta];

  continue[etiqueta];

- Sirve para bucles anidados.

Ejemplo: break y continue con etiq.

- **¿Qué salida produce el código del fichero PruebaControl.java?**

```
boolean valor = true;
bucle_1: while (valor) {
    System.out.println("En bucle while");
    for (int i = 0; i < 5; i++) {
        System.out.println("En bucle for i:" +i);
        break bucle_1; // break;
    }
}
```

```
bucle_2: for (int i = 0; i < 2; i++) {
    System.out.println("i:" +i);
    for (int j = 0; j < 10; j++) {
        System.out.println("j:" +j);
        continue bucle_2; // continue
    }
}
```

Tema 3: Arrays

Tema 3: Arrays

- **Arrays**
  - Arrays en Java
  - Bucle for mejorado
  - Copia de arrays
  - Arrays multidimensionales
  - Arrays multidimensionales irregulares
- **El método main**
- **Métodos en Java**
  - Llamada a métodos
  - Paso de parámetros a métodos

Arrays en Java

- Es un **objeto** que sirve para guardar variables del mismo tipo.
- **Declaración de un array**: hay 2 formas válidas

```
tipo[ ] nombreArray;   // recommended
tipo  nombreArray [ ]; // legal but less readable
```

- ***Ejemplos de declaración de arrays:***

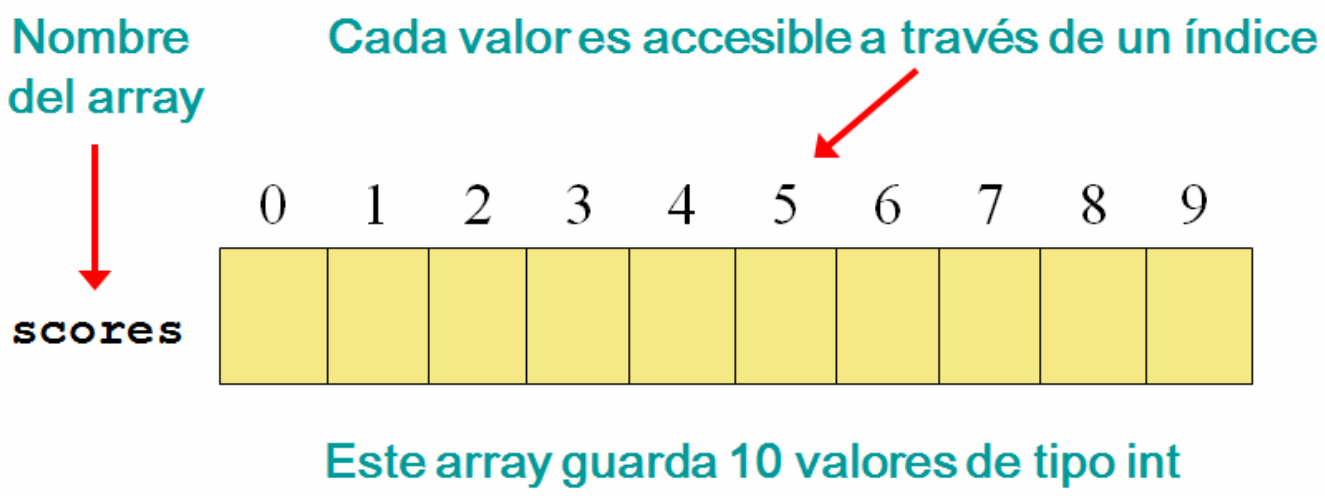
```
int[] key;   // Square brackets before name (recommended)
int key []; // Square brackets after name (legal but less
            // readable)
```

```
double[] salarios;          // declaro un array de double
char cAlfabeto[ ];          // declara un array de char
String[] direcciones;       // declara un array de String
```

Arrays en Java

- Tras declararlo, para crearlo debemos asignarle un tamaño mediante el operador **new**.

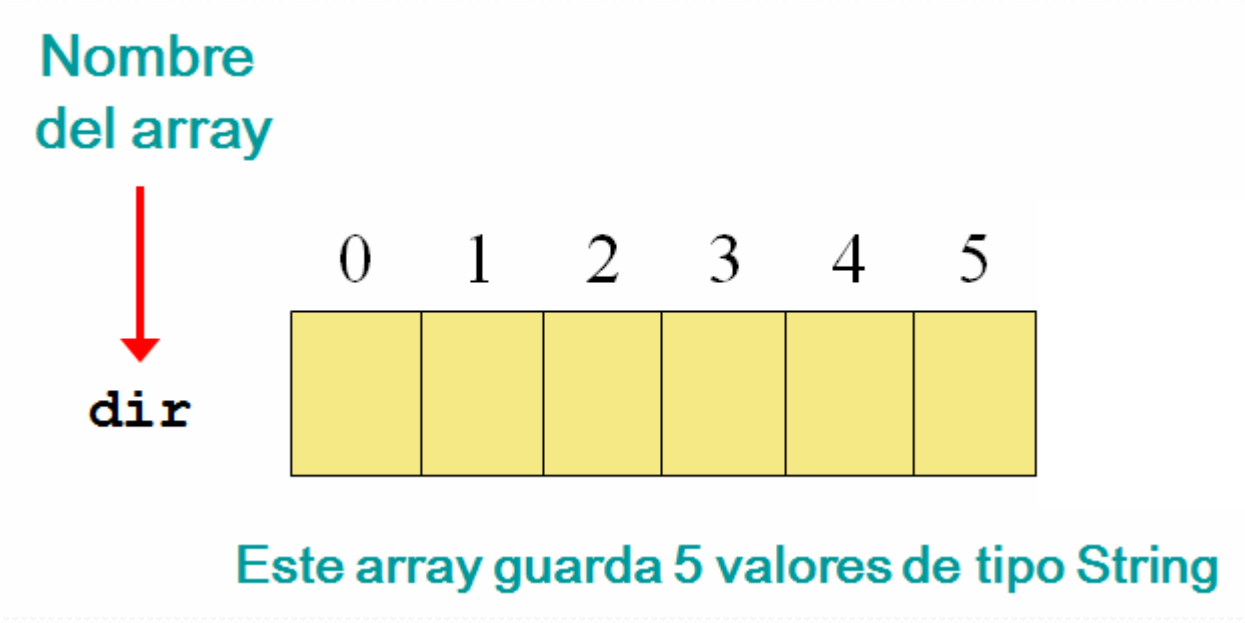
```
int[] scores;                 // 1. declaración un array de int
scores = new int[10];         // 2. crea array que guardará 10 int
```



Arrays en Java

- Podemos declarar y crear el array en una misma línea:

String[] dir = new String[5];



Arrays en Java

- ***Resumen***: declaración y creación de un array

```
tipo[] nombreArray; // 1. declaración
nombreArray  = new tipo[tamanoArray];  // 2. creación
```

```
tipo[] nombreArray = new tipo[tamanoArray]; // 1. y 2.
```

donde ***tamanoArray*** puede ser una variable de tipo entero, por ej.

Arrays en Java

- **Ejemplo**: ¿es lo mismo?

  int[ ] a, b;

  int a[], b;

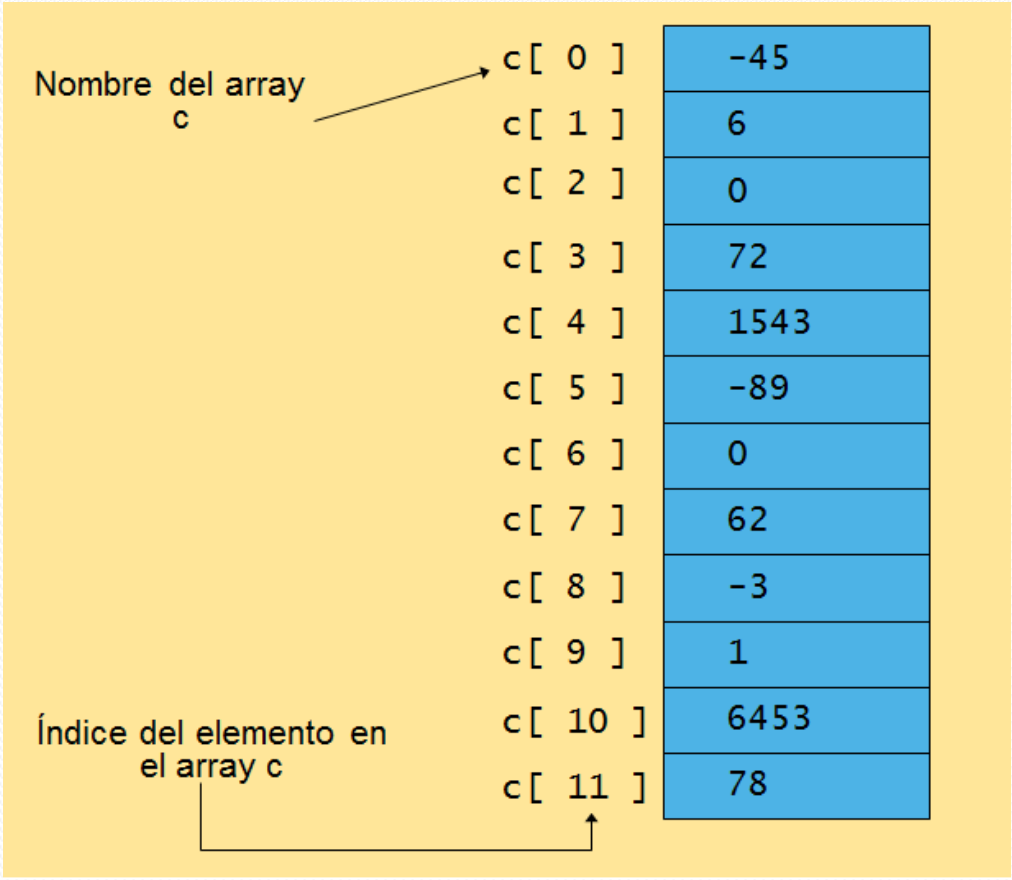
- Una vez definido el tamaño de un array éste será fijo.
- Cuando un array es creado, todos sus elementos son inicializados al valor por defecto del tipo (***importante***).
- También podemos declarar, crear e inicializar un array en una misma línea:

int[] nums = {10, 20, 30, 40, 50, 60};

String beatles[] = { "John", "Paul", "George", "Ringo" };

Arrays en Java

- ✓ Para acceder a un elemento de un array: **nombreArray[indice]**
- ✓ ***Ojo***: *indice* no puede ser de tipo **long** (sí **byte**, **short**, **int**, **char**).



Ejemplo: ¿qué salida produce?

```
int valor;
valor = c[1];
System.out.println(valor);

valor = c[10];
System.out.println(valor);

valor = c[12];
System.out.println(valor);
```

Arrays en Java

- ✓ **Todos los arrays comienzan en el índice 0.**

  Ejemplo: ¿es correcto? c[-1] = 24;

- ✓ Para saber la **longitud de un array**, uso el atributo **length**, muy útil para recorrer arrays. *Ejemplo:*

```
int [] nums = new  int[10];
for (int i=0;i<nums.length;i++){
        nums[i]=i*2;
        System.out.println("num["+ i + "]:" + nums[i]);
}
```

Arrays en Java

- **Límites de un array**:
  - Los índices de todos los arrays comienzan en 0
  - Número de elementos almacenado en atributo ***length***
  - Si excedo los límites de un array → salta excepción del tipo *ArrayIndexOutOfBoundsException*
- **Redimensionamiento de arrays**:
  - El tamaño de un array es fijo.
  - Se puede usar la misma variable para un array nuevo:

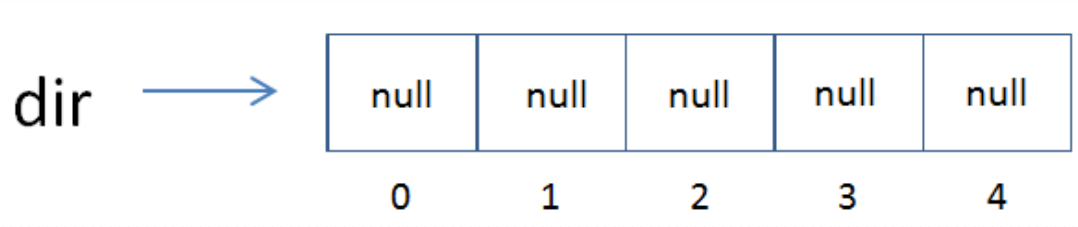
```
int[] miArray = new int[6];
miArray = new int[10];
// ¡No podemos acceder al primer array!
```

Arrays en Java

- ✓ **Ejemplos (1):**

```
String[] dir = new String[5];
for (int j=0;j<dir.length;j++){
    System.out.println("dir[" + j + "]=" + dir[j]);
}

System.out.println();
```




```
dir[0]=null
dir[1]=null
dir[2]=null
dir[3]=null
dir[4]=null
```

Arrays en Java

Sobre Unicode:

http://unicode-table.com/es/
https://es.wikipedia.org/wiki/Unicode

- ✓ **Ejemplos (2):**

```
// generates English alphabet
char[] alphabet = new char[26];
int i = 0;
for (char c='a';c<='z';c++){
    alphabet[i++] = c;
}

System.out.println("English alphabet:");
for (i=0;i<alphabet.length;i++){
    System.out.print(alphabet[i]);
    System.out.print( (i==alphabet.length-1) ? " ":", ");
}
```

```
English alphabet:
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
```

Arrays en Java

- **Actividad:**

*Escribe un programa que calcule la **suma** y la **media** de los números almacenados en un array de números decimales, mostrando además el **mayor** y el **menor** de los números.*

Bucle for mejorado

- ✓ Versión disponible a partir de Java 5.
- ✓ Facilita el recorrido de arrays y colecciones.
- ✓ Sintaxis:

```
for(tipo variable:var_array){
        //instrucciones
}
```

- ✓ *Ejemplo*:

"Versión tradicional" bucle for

```
int [] nums = {4, 6, 30, 15};

// Recorre los elementos del array
for (int i=0;i<nums.length;i++){
        System.out.println(nums[i]);
}
```

Bucle for mejorado

- *Ejemplo*:

```
int[] nums = {3, 6, 30, 15};

// Bucle for-each (for mejorado)
for (int valor:nums) {
      System.out.println(valor);
}
```

- *Ejemplo 2:*

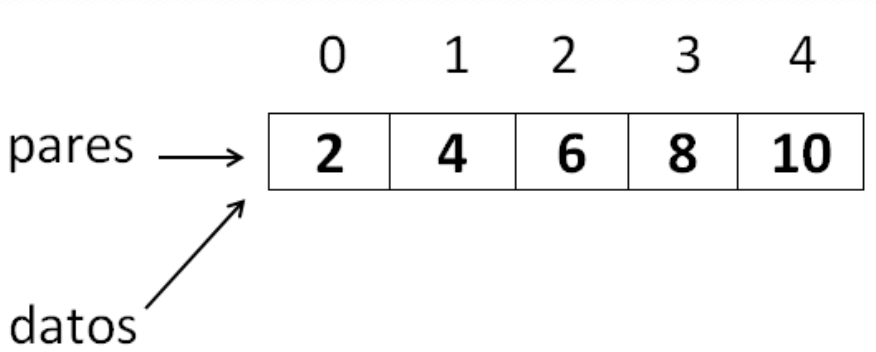
```
String[] ciudades = {"Málaga","Almería","Granada"};

for (String c:ciudades) {
     System.out.println(c);
}
```

Copia de arrays en Java

- ***Ejemplo***:

```
int[] pares = {2, 4, 6, 8, 10} ;
int[] datos = pares;
```

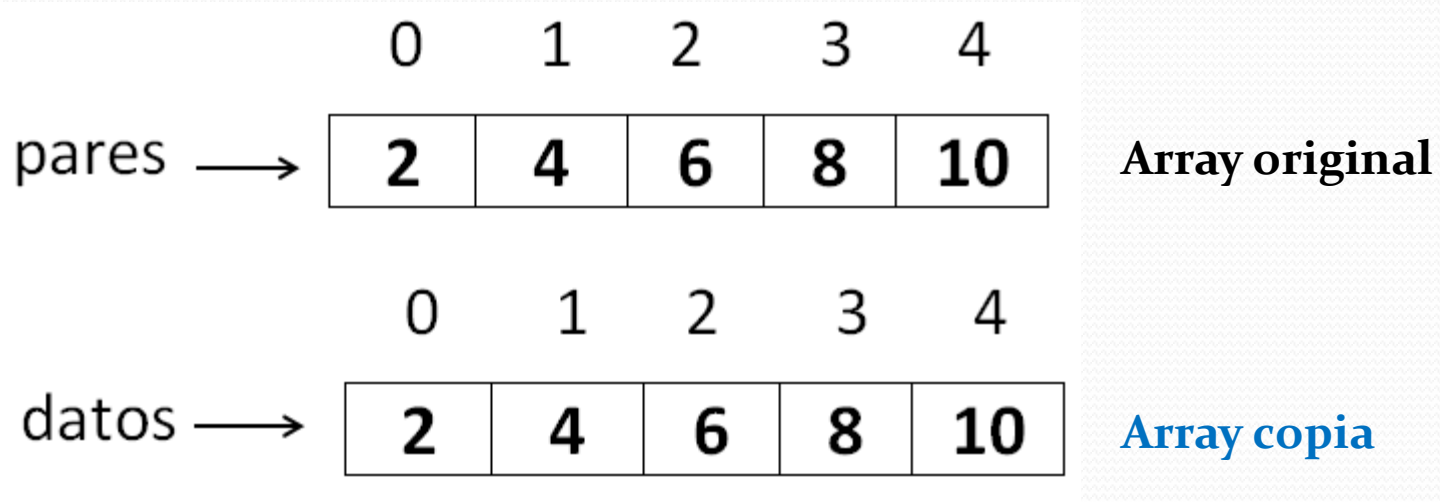


- **Ojo:** esto no copia el array, sólo copia la referencia al array

Copia de arrays en Java

- ***Ejemplo***:

  - Queremos conseguir algo como esto:



Copia de arrays en Java

- ✓ Para copiar un array, hay 2 opciones:

1) **Crear un nuevo array y copiar los elementos uno a uno**:

```
int[] datos = new int[pares.length];

for (int i=0;i<pares.length;i++) {
    datos[i] = pares[i]; // copia elemento a elemento
}
```

```
for (int aux:datos) {
    System.out.print(aux + " ");
}
```

Copia de arrays en Java

2) **Usar una función de la biblioteca estándar de clases de Java**:

```
System.arraycopy(from,fromIndex,to,toIndex,n);
```

```java
int[] datos = new int[pares.length];

// arraycopy(src, srcPos, dest, destPos, length)
System.arraycopy(pares, 0, datos, 0, pares.length);
```