

# TEMA: INTRODUCCIÓN A JAVA MATRICES

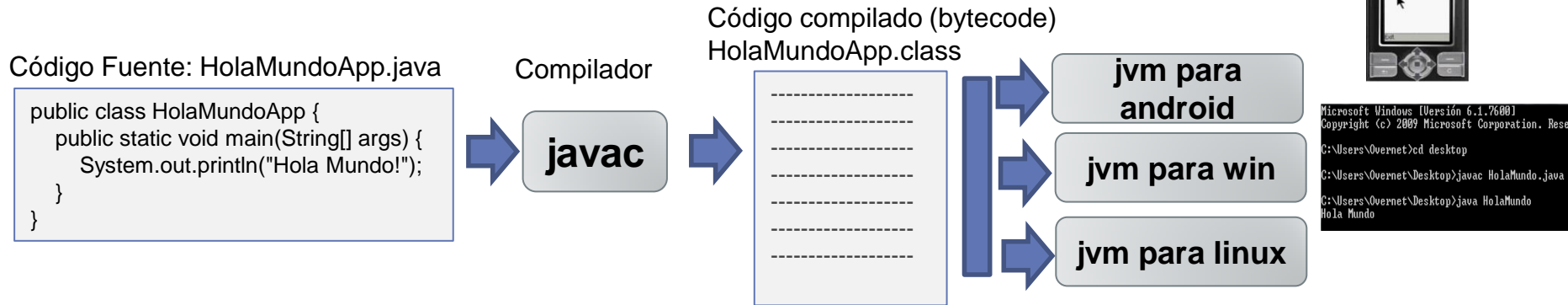
---

Taller de Programación.

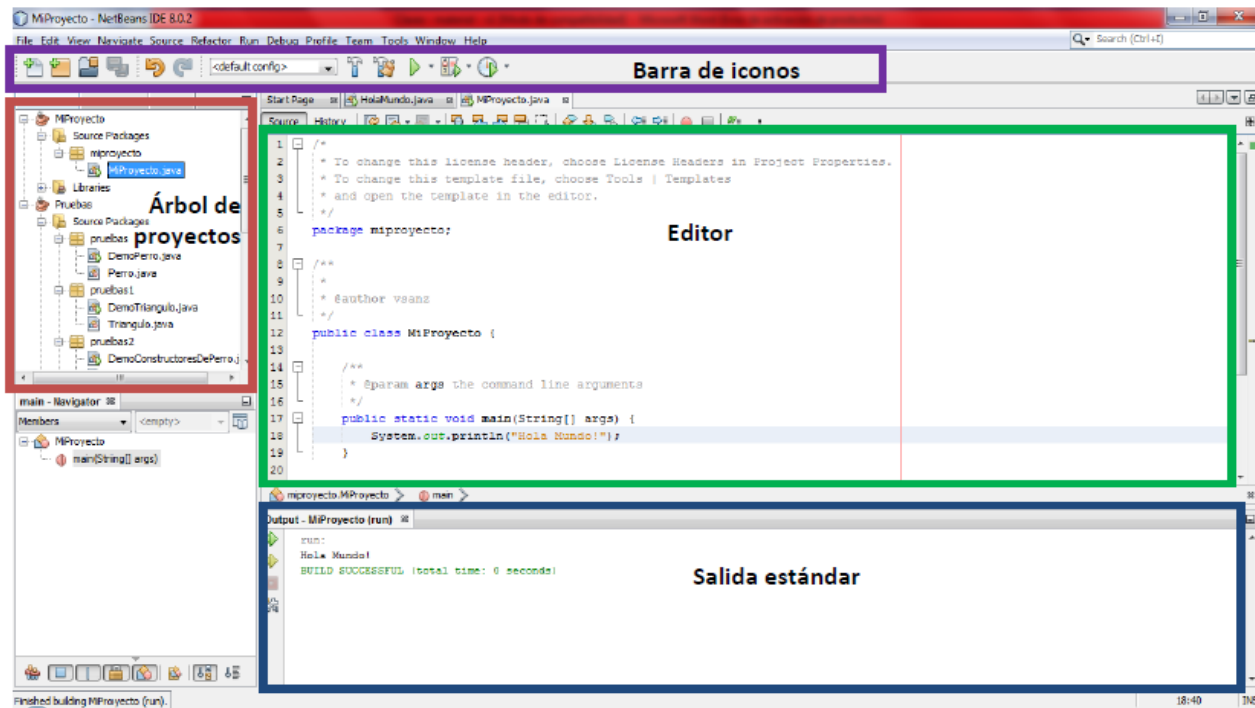
Módulo: Programación Orientada a Objetos

# Java

- Lenguaje de propósito general. Paradigmas: Imperativo/OO
- Plataforma Java:
  - Plataforma de ejecución (JRE): incluye componentes requeridas para ejecutar aplicaciones Java, entre ellas la JVM.
  - Plataforma de desarrollo (JDK): incluye compilador, depurador, generador de documentación,
- Permite generar aplicaciones multiplataforma.
- Codificación y ejecución de app. java:



# IDE NetBeans



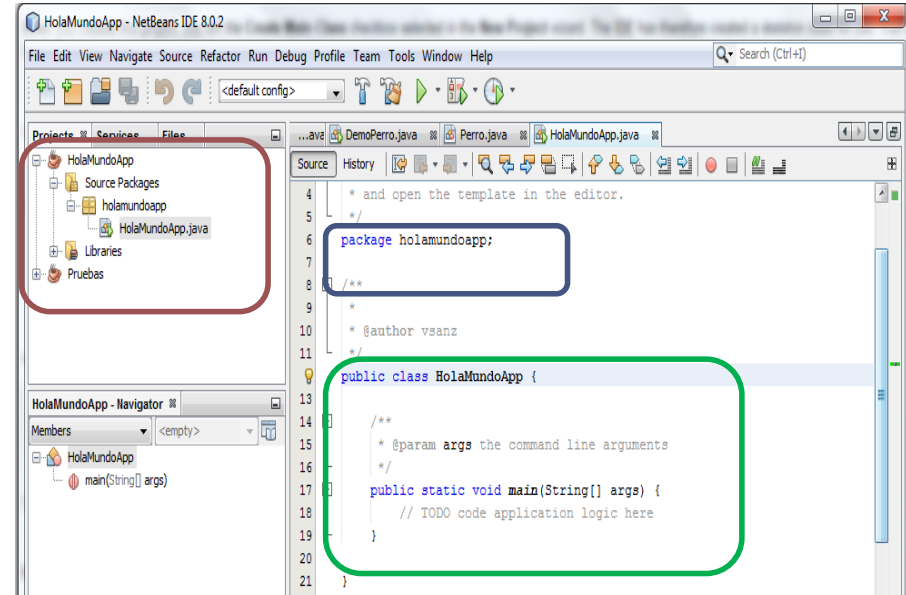
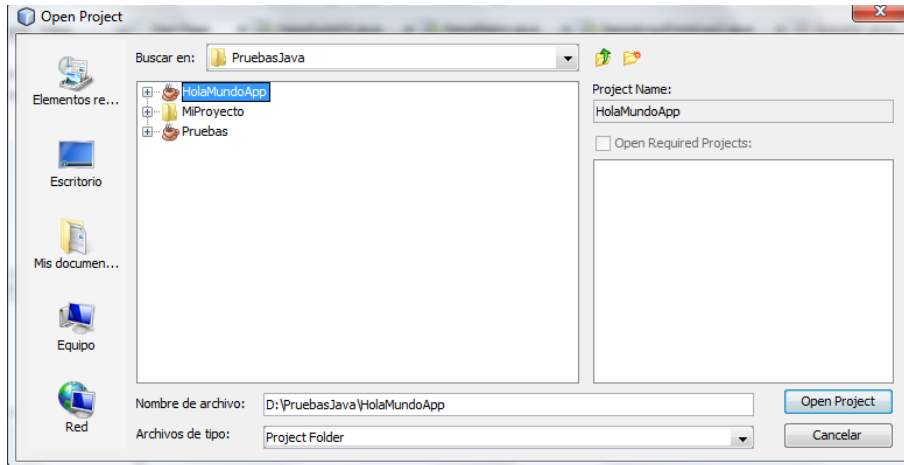
- Reúne herramientas para desarrollar SW.
  - Editor
  - Compilador
  - Depurador
  - ...
- Libre y gratuito
- Descargar desde
  - <https://netbeans.apache.org/download/index.html>



# IDE NetBeans. Uso.

## Abrir Proyecto

- File > Open Project.
- Buscar ubicación del proyecto.
- Click en “Open Project”.



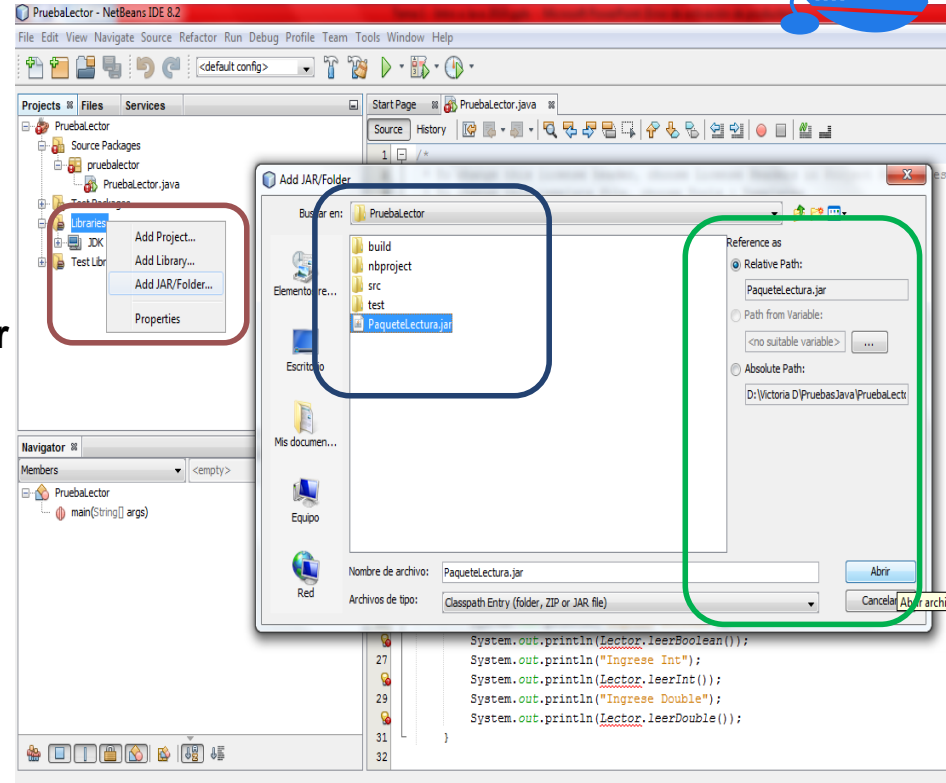
- Proyecto
- Paquetes (carpetas dónde organizamos los códigos)
- Códigos: extensión .java

# IDE NetBeans. Uso.

## Agregar PaqueteLectura.jar al Proyecto

- Copiar **PaqueteLectura.jar** en la carpeta del proyecto.
- Click derecho sobre **Libraries** > **Add JAR/Folder**
- **Seleccionar PaqueteLectura.jar desde la carpeta del proyecto**
- **Relative Path** debe quedar seleccionado marcando PaqueteLectura.jar
- **Abrir**

Realizar este paso cada vez que trabaje sobre un proyecto distinto





# IDE NetBeans. Uso.

## Crear nuevo “Prog Ppal”

- Click derecho sobre la carpeta contenedora.
  - Ej: “tema 1”
- New > Java Main Class
- Class Name: Poner un nombre
- Finish

Aparecerá un archivo .java con el esqueleto del programa principal

## Cerrar Proyectos Abiertos

- File > Close All Projects.

## Crear nuevo proyecto (ej. parcial)

- File > New Project > Java Application
- Project Name: Poner un nombre
- Project Location: Seleccionar ubicación
- Finish

## El “programa principal”

```
public class NombreAplicacion {  
    public static void main(String[] args) {  
        /* Código */  
    }  
}
```

- Main = “Programa principal”. { } delimita el cuerpo.
- Sentencias de código separadas por punto y coma (;).
- Se recomienda indentar el código para facilitar su lectura.
- Comentarios:
  - De líneas múltiples /\* Esto es un comentario \*/.
  - De línea única // Este es un comentario
- Case-sensitive (sensible a las mayúsculas y minúsculas)

# Mostrar datos en la salida estándar

- Sentencias que permiten mostrar datos en consola:
  - `System.out.print(...)` NO realiza salto de línea
  - `System.out.println(...)` Realiza salto de línea
- Ejemplo

```
public class Demo04Salida{  
    public static void main(String[] args) {  
        System.out.print("Hola Mundo! ");  
        System.out.println("Hola Mundo! ");  
        System.out.println(1234);  
        System.out.println(true);  
    }  
}
```

**Para mostrar varios datos, unirlos con +**

```
int año=2018;  
System.out.println ("Hola Mundo " + año + "!");
```

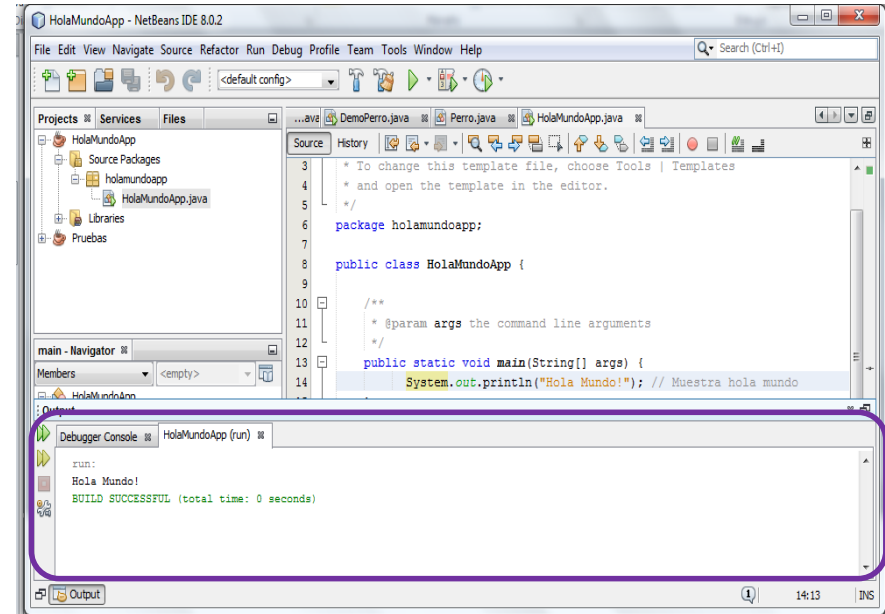




# IDE NetBeans. Uso.

# Correr programa

- Click derecho sobre el archivo que contiene el *main*.
  - Ej: Demo04Salida.java
- Run File.



## Declaración variables locales a método (main u otro)

- Se declaran en zona de *código* (*no toman valor por defecto*).  
    Tipo nombreVariable;      (Opcional: dar valor inicial)
- Convención de nombres: comenzar con minúscula, luego cada palabra en mayúscula.
- Asignación: nombreVariable = valor;

- Tipos primitivos: la variable almacena un valor

Tipo Primitivo	Ejemplo
boolean	true false
char	'a' '0' '*'
int	102
double	123.4

- *String* para manipular cadenas. Ejemplo “esto es un string”.

# Manipulación de variables

- Operadores para tipos primitivos y String

## Operadores aritméticos (tipos de datos numéricos)

+    operador suma  
-    operador resta  
\*    operador multiplicación  
/    operador división  
%    operador resto

## Operadores relacionales (tipos de datos primitivos)

==   Igual  
!=   Distinto  
>   Mayor  
>=   Mayor o igual  
<   Menor  
<=   Menor o igual

## Operadores unarios aritméticos (tipos de datos numéricos)

++   operador de incremento; incrementa un valor en 1  
--   operador de decremento; decrementa un valor en 1

## Operadores Condicionales

&&   AND  
||   OR  
!   NOT

## Operador de concatenación para String

+    Operador de concatenación de Strings


# Declaración de variables. Ejemplos.

```
public class Demo01DeclaracionVariables {  
    public static void main(String[] args) {  
        boolean encuentre=false;           //1  
        int miDNI =11222333, tuDNI = 10555444; //2  
        char sexo, inicial='C';             //3  
        sexo = 'F';                         //4  
        double miSueldo=1000.30;            //5  
        String miNombre="Pepe";            //6  
    }  
}
```

```
public class Demo02OperadoresUnarios {  
    public static void main(String[] args) {  
        int i = 3; // i vale 3  
        i++;       // i vale 4  
        i--;       // i vale 3  
    }  
}
```

```
public class Demo03CalculoAritmeticoA{  
    public static void main (String[] args) {  
        int result = 1 + 2;    // result es 3  
        result = result - 1;   // result es 2  
        result = result * 2;   // result es 4  
        result = result / 2;   // result es 2  
        result = result % 2;   // result es 0  
    }  
}
```

```
public class Demo03CalculoAritmeticoB{  
    public static void main (String[] args) {  
        int i = 4/3;           // División entera    i es 1  
        double d1 = 4.0/3.0;    // División real      d1es 1.3333  
        double d2 = 4/3;        // División entera    d2 es 1.0  
        double d3 = (double) 4/3; // División real    d3=1.333  
    }  
}
```



Conversión explícita del op1 a double

# Ingreso de datos desde teclado

- Uso de *Lector* (funcionalidad definida en *PaqueteLectura.Lector*)

```
import PaqueteLectura.Lector;                                // Importar funcionalidad para lectura
public class Demo05Entrada
{
    public static void main(String[] args) {
        System.out.println("Ingresa nombre");
        String nombre = Lector.leerString();                  //Lee y devuelve el string ingresado antes del enter
        System.out.println("Ingresa si trabaja (true/false)");
        boolean trabaja = Lector.leerBoolean();              //Lee y devuelve el boolean ingresado antes del enter
        System.out.println("Ingresa edad");
        int edad = Lector.leerInt();                          //Lee y devuelve el int ingresado antes del enter
        System.out.println("Ingresa sueldo");
        double sueldo = Lector.leerDouble();                 //Lee y devuelve el double ingresado antes del enter

        System.out.println("N:" + nombre + " T:" + trabaja + " E:" + edad + " S:" + sueldo );
    }
}
```

# Generación de datos aleatoria

- Uso de *GeneradorAleatorio* (funcionalidad definida en *PaqueteLectura.GeneradorAleatorio*)

```
import PaqueteLectura.GeneradorAleatorio;    // Importar funcionalidad Generador Aleatorio

public class Demo05Generador
{
    public static void main(String[] args) {
        GeneradorAleatorio.iniciar();           //inicia el generador aleatorio
        System.out.println(GeneradorAleatorio.generarInt(10));    //genera un int entre 0 y 9
        System.out.println(GeneradorAleatorio.generarDouble(10)); //genera un double entre 0 y 9
        System.out.println(GeneradorAleatorio.generarBoolean());  //genera un boolean
        System.out.println(GeneradorAleatorio.generarString(4));  //genera un string de long. 4
    }
}
```

# Estructuras de control

## Selección

if (condición)

acción(es) a realizar cuando

condición es true

else

acción(es) a realizar cuando

condición es false

Encerrar entre {} en caso de incluir varias sentencias.

Cuando sólo incluye una sentencia, finalizarla con ;

## Iteración pre-condicional

while (condición)

acción(es) a realizar cuando

condición es true

Leer acerca del **case** (**switch** en java) en:

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html>

## Iteración post-condicional

do{

acción(es)

} while (condición)

### Diferencia do-while y while

- Ejecuta acción(es) y luego evalúa condición
- Cuando condición es true => ejecuta otra vez acción(es)
- Cuando condición es false => finaliza do

# Estructuras de control

**Repetición**    `for (inicialización; condición; expresión)`  
                  `acción(es)`

- *Inicialización*: expresión que se ejecuta una vez al comienzo y da valor inicial a la variable índice.
- *Condición*: expresión lógica, se evalúa antes de comenzar una nueva iteración del `for`; cuando da `false` termina el `for`.
- *Expresión*: expresión que se ejecuta al finalizar cada iteración del `for` (incr. o decr. del índice).

```
int i;  
for (i=1; i<= 10; i++)  
    System.out.println(i);
```

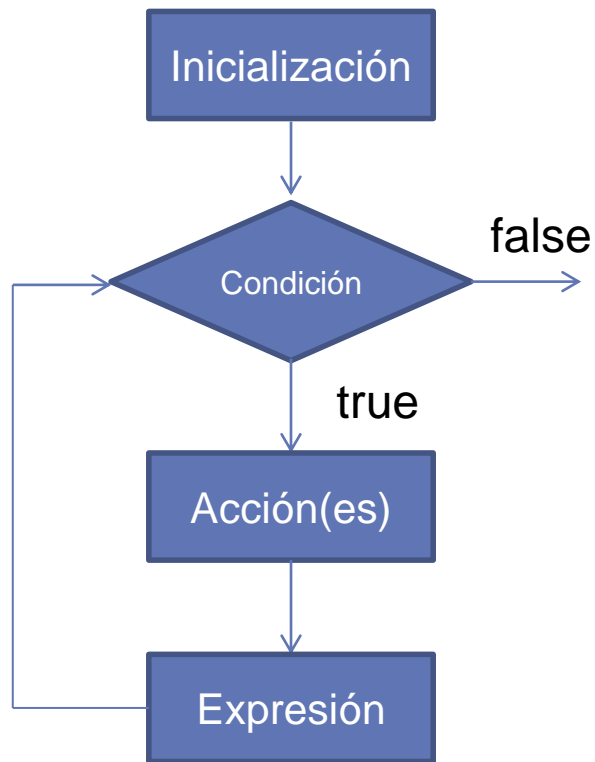
¿Qué imprime?

¿Modificar para imprimir pares?

```
int i;  
for (i=10; i > 0; i=i-1)  
    System.out.println(i);
```

¿Qué imprime?

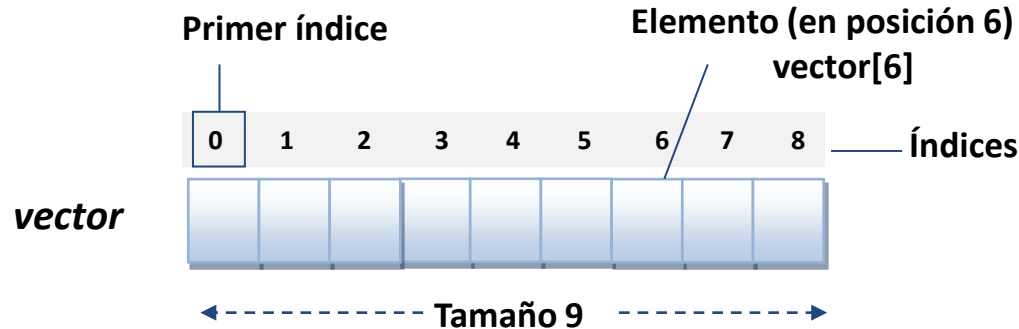
¿Es lo mismo poner `i--` ?





# Arreglos

- Almacenan un número fijo de valores primitivos // *objetos* (del mismo tipo)
- Acceso en forma *directa* a las posiciones.
- Dimensión física: se establece al crearlo.
- Índice: entero, comenzando desde 0.



# Arreglos unidimensionales - Vector

- Declaración

*TipoElemento* [] nombreVariable;

- Creación

nombreVariable = new TipoElemento[DIMF];

- Acceso a elemento

nombreVariable [posición]

## Ejemplo:

```
int [] contador = new int[10];  
for (i=0;i<10;i++) contador[i]=i;  
...  
System.out.println("La Pos. 1 tiene " +contador[1]);
```

# Arreglos bidimensionales - Matrices

- Ejemplo de situaciones de uso
  - Representar sala de un teatro (30 filas, 20 butacas por fila) para saber si cada butaca se encuentra vendida o no.
  - Representar una tabla que indique la cantidad de lluvia caída para cada provincia de Argentina y cada mes del año actual.
  - Representar un cartón del BINGO
  - ...

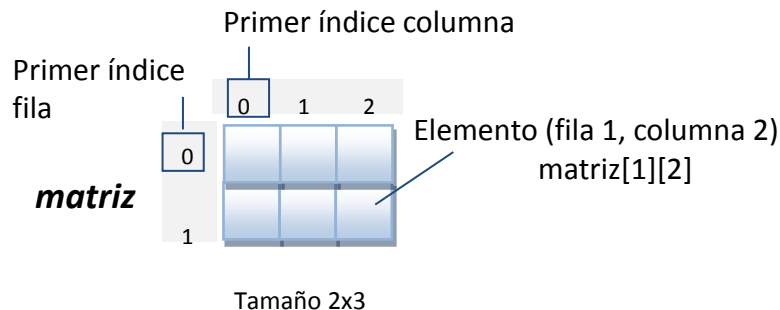


# Arreglos bidimensionales - Matrices

- Colección ordenada e indexada de elementos.
- Esta estructura de datos compuesta permite acceder a cada componente utilizando **dos índices (fila y columna)** que permiten ubicar un elemento dentro de la estructura

- Características :

- Homogénea
- Estática
- Indexada
- Lineal



En Java, cada **índice** es **entero** y comienzan desde 0.

Los **elementos** de la matriz pueden ser int, double, char, boolean u objetos (mismo tipo).

¿Otros lenguajes?

# Arreglos bidimensionales - Matrices

- Declaración

*TipoElemento* `[][]` nombreVariable;

- Creación

nombreVariable = new *TipoElemento* `[DIMF][DIMC]`;

- Acceso a elemento

nombreVariable `[posFil] [posCol]`

- Ejemplo:

```
int [][] tabla = new int[3][4];
```

```
int i, j;
```

```
for (i=0;i<3;i++)
```

```
    for (j=0;j<4;j++)
```

```
        tabla[i][j] = GeneradorAleatorio.generarInt(10);
```

```
System.out.println("La Pos. 1,2 tiene " +tabla[1][2]);
```

## Gráficamente

*tabla*

	0	1	2	3
0				
1				
2				

Tamaño 3x4

### Pensar las operaciones:

- Imprimir el contenido de la matriz
- Imprimir el contenido de una columna específica
- Sumar los elementos de una fila específica