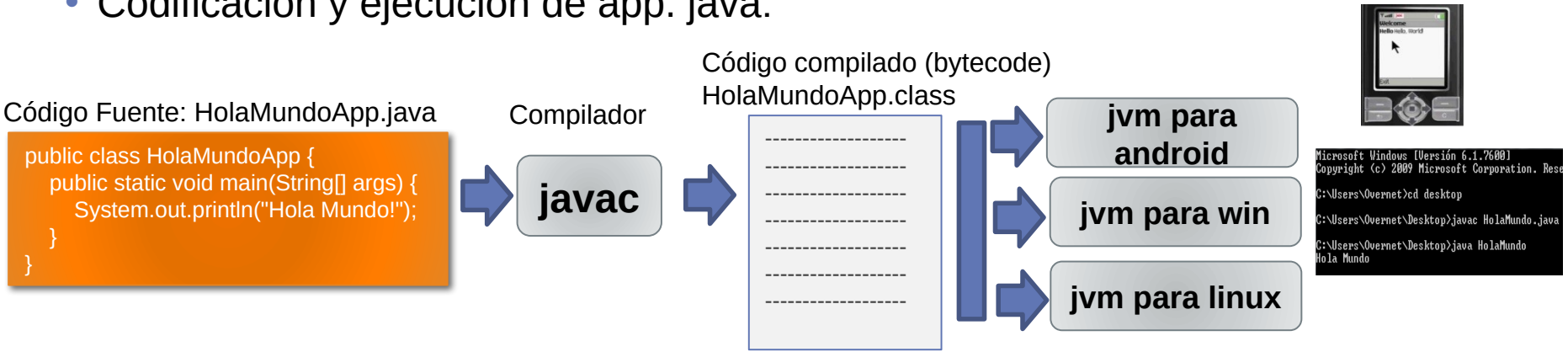
PROGRAMACIÓN II
MÓDULO 2 - PROGRAMACIÓN ORIENTADA A OBJETOS

Clase 1 – POO

- **Introducción a Java**
- **Matrices**

Java

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



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)

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 encontre=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 Demo04CalculoAritmeticoB{
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

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);
   }
}
```

Vamos a probar este ejemplo en BlueJ

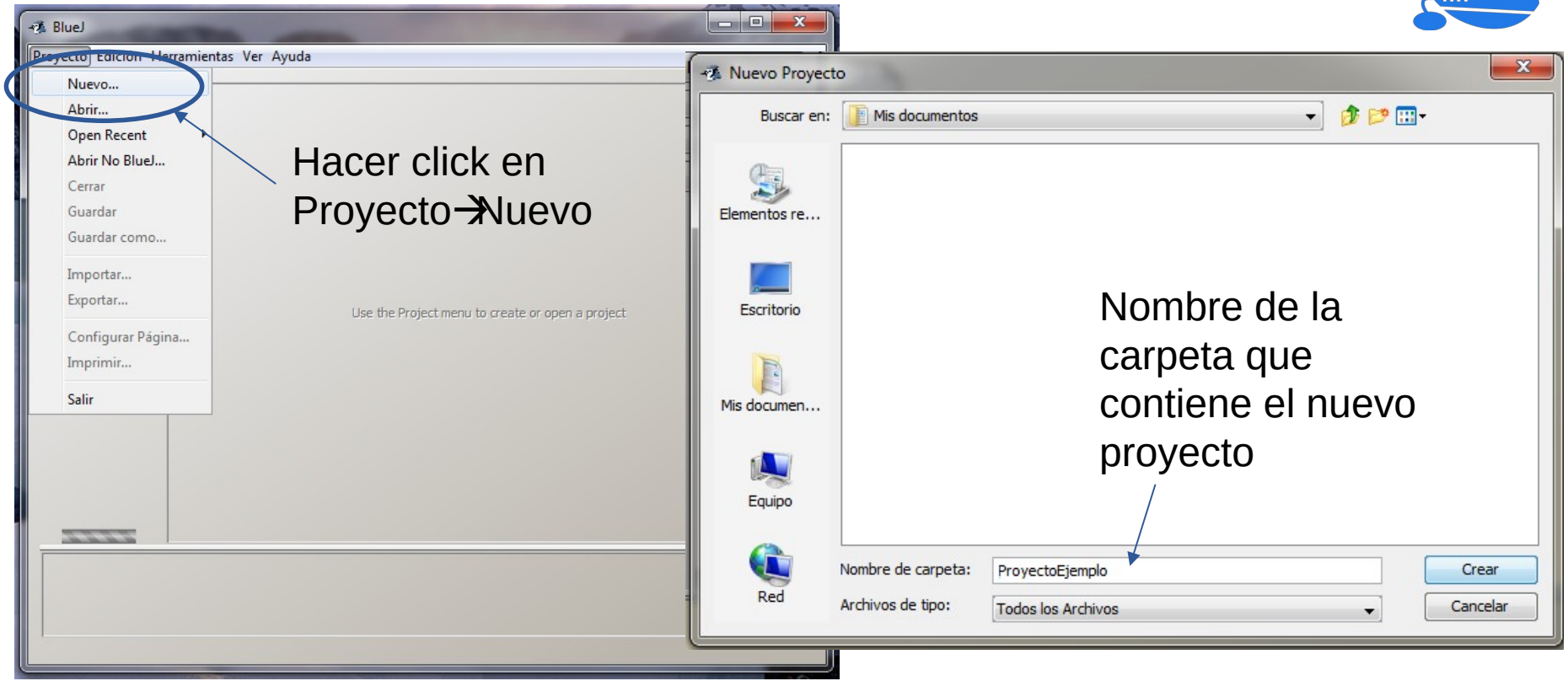
Para mostrar varios datos, unirlos con +

```
int año=2016;
System.out.println ("Hola Mundo " + año + "!");
```

BlueJ (www.bluej.org)

- Entrar al sitio y descargar el instalador.

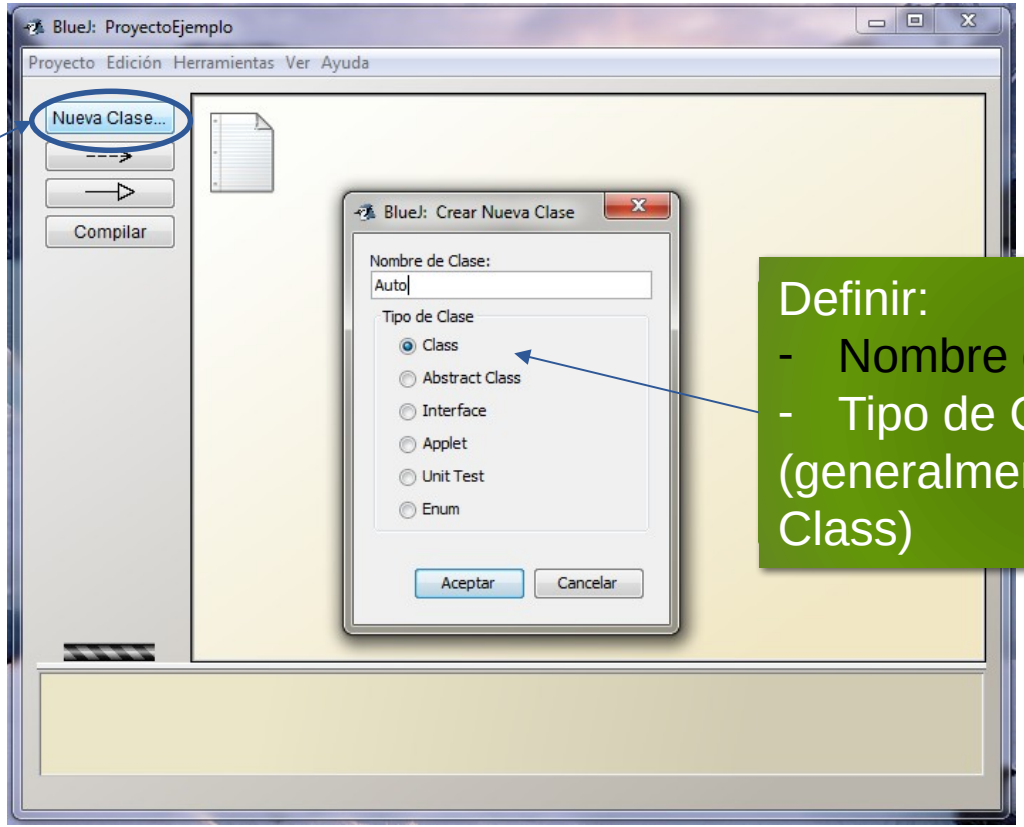
BlueJ. Uso.


BlueJ
Proyecto Edición Herramientas Ver Ayuda
Nuevo...
Abrir...
Open Recent
Abrir No BlueJ...
Cerrar
Guardar
Guardar como...
Importar...
Exportar...
Configurar Página...
Imprimir...
Salir
Use the Project menu to create or open a project
Hacer click en Proyecto→Nuevo
Nuevo Proyecto
Buscar en:
Mis documentos
Elementos re...
Escritorio
Mis documen...
Equipo
Red
Nombre de la carpeta que contiene el nuevo proyecto
Nombre de carpeta:
ProyectoEjemplo
Archivos de tipo:
Todos los Archivos
Crear
Cancelar

BlueJ. Uso.

Crear Clases

Hacer click en Nueva clase



Definir:
- Nombre de la clase
- Tipo de Clase (generalmente utilizaremos Class)

BlueJ. Uso.

Compilar

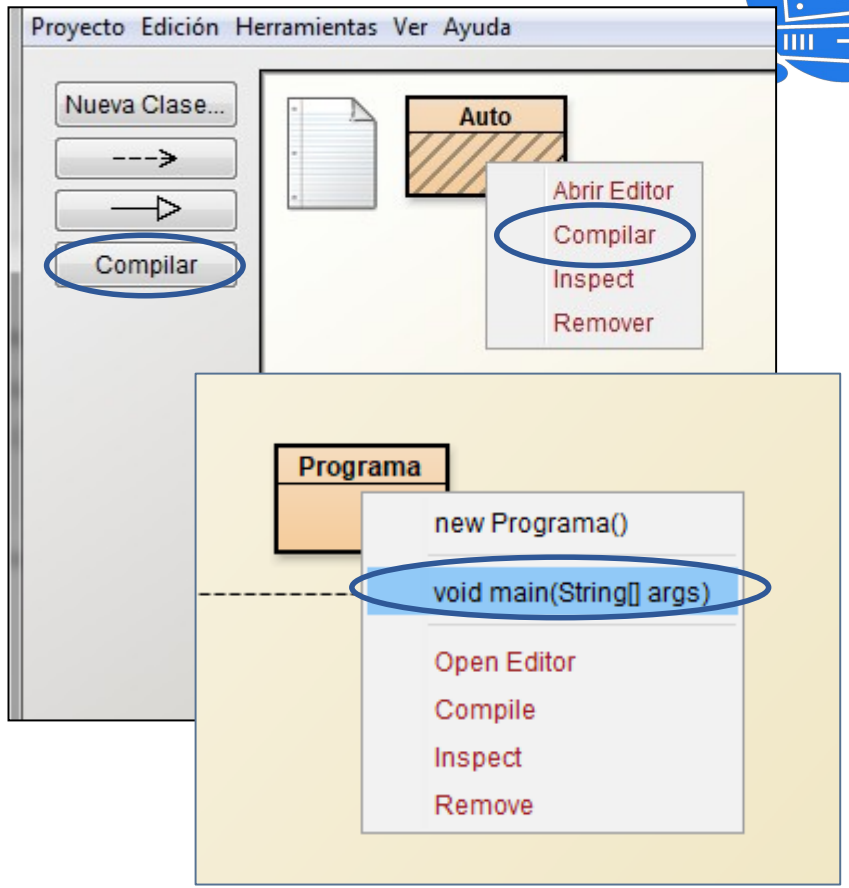
- Compilar desde el botón (para todas las clases)

ó

- Pararse sobre la clase y click derecho y compilar

Correr programa

- Pararse sobre la clase que contiene el *main*.
  - Ej: Demo04Salida.java
- Click derecho y seleccionar el método.



Ingreso de datos desde entrada estándar

- *Scanner* permite tomar datos desde una entrada (ej: System.in = teclado).

Lee y devuelve un **int**	in.**nextInt()**
Lee y devuelve un **double**	in.**nextDouble()**
Lee y devuelve un **boolean**	in.**nextBoolean()**
Lee y devuelve sec. **caracteres hasta ~~b~~/ t / ~~cr~~**	in.**next()**
Lee y devuelve sec. **caracteres** hasta **~~cr~~**	in.**nextLine()**

Ingreso de datos desde entrada estándar

- *Scanner* permite tomar datos desde una entrada (ej: System.in = teclado).

```
import java.util.Scanner; // Importar funcionalidad para entrada
public class Demo05Entrada
{
  public static void main(String[] args)
  {
    Scanner in = new Scanner(System.in);      // Declarar el scanner e indicar que se leerá desde teclado
    System.out.print("Ingrese edad: ");
    int edad = in.nextInt();
    System.out.print("Ingrese peso: ");
    double peso = in.nextDouble();
    System.out.print("Ingrese true o false: ");
    boolean tieneDueño = in.nextBoolean();
    System.out.print("Ingrese nombre: ");
    String nombre = in.next();
    in.nextLine(); //Para que lea el Enter del ingreso anterior y luego usar lectura por oración
    System.out.print("Ingrese descripción: ");
    String descripcion = in.nextLine();
    in.close();   // Cerrar el scanner
  }
}
```

Más en: https://bit.ly/3ch7fNF

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
```

Adicional:
Pueden 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 => finalizado

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);
```

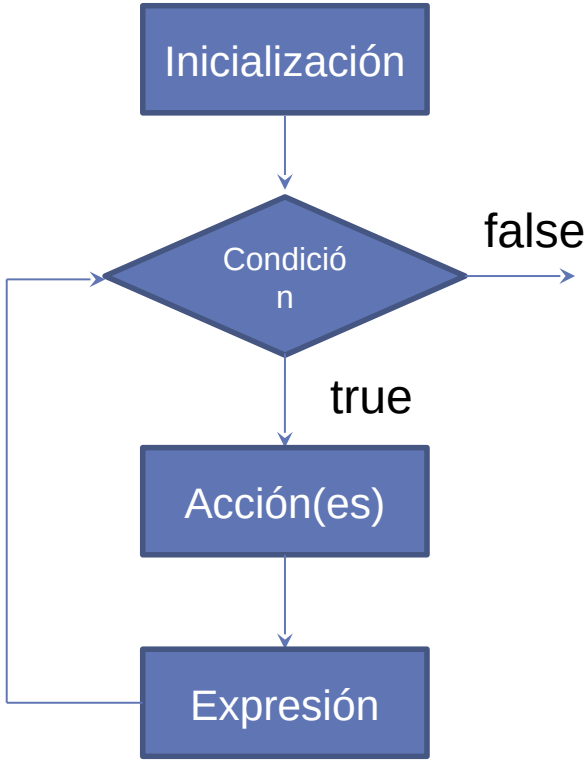
¿Qué imprime?

¿Modificar para imprimir pares?

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

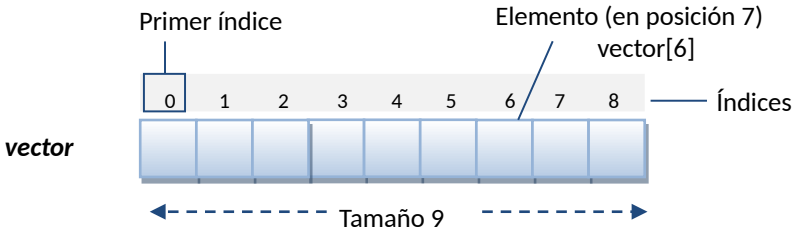
¿Qué imprime?

¿Es lo mismo poner i-- ?



Arreglos

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



Arreglos unidimensionales - Vector

- Declaración

TipoElemento [] nombreVariable*;*

- Creación

nombreVariable = new TipoElemento[DIMF];

- Acceso a elemento

nombreVariable [posición]

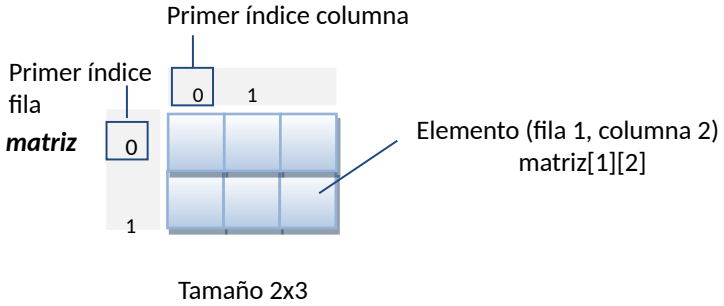
Ejemplo:

```
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

- 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

- 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

- Declaración

TipoElemento [][] nombreVariable;

- Creación

nombreVariable = new T*ipoElemento [DIMF][DIMC];*

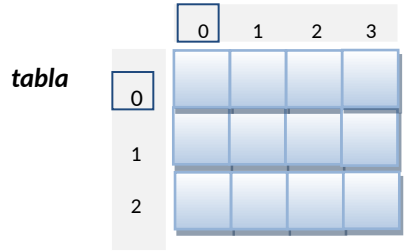
- Acceso a elemento

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

Gráficamente



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*