

TEMA: INTRODUCCIÓN A JAVA. MATRICES.

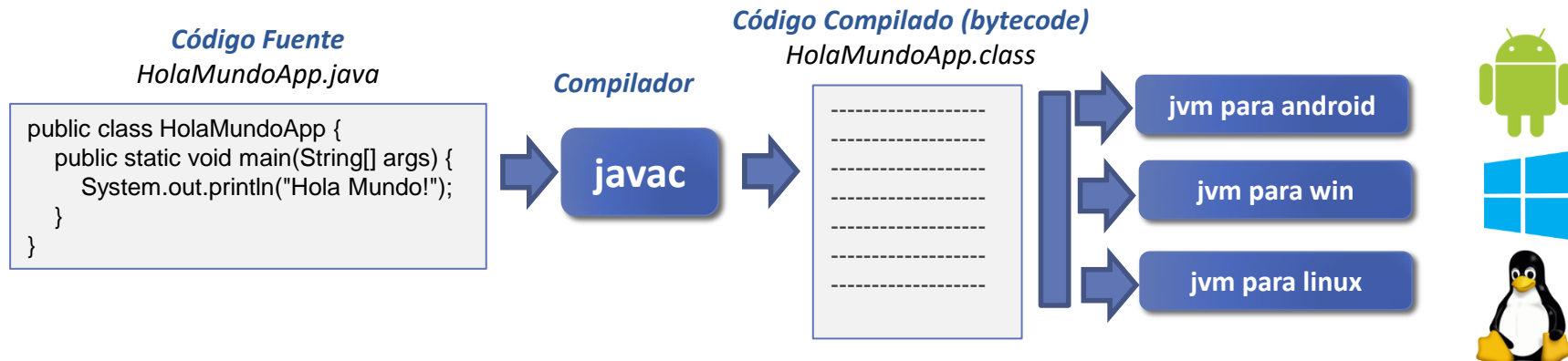
Taller de Programación

Módulo: Programación Orientada a Objetos



- Lenguaje Java: Propósito general. Paradigmas: Imperativo/OO.
- Plataforma Java:
 - Plataforma de desarrollo (JDK, [Java Development Kit](#)): incluye compilador, depurador, generador de documentación.
 - Plataforma de ejecución (JRE, [Java Runtime Environment](#)): incluye componentes requeridas para ejecutar aplicaciones Java, entre ellas la JVM ([Java Virtual Machine](#)).
- Codificación y ejecución de app. java:

App Multiplataforma



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 y 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: d1 es 1.3333  
        double d2 = 4/3;        // División entera: d2 es 1.0  
        double d3 = (double) 4/3; //División real d3 es 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);  
    }  
}
```

Para mostrar varios datos, unirlos con +

```
int año=2020;  
System.out.println ("Hola Mundo " + año + "!");
```

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("Ingrese nombre");
        String nombre = Lector.leerString();    //Lee y devuelve el string ingresado antes del enter
        System.out.println("Ingrese si trabaja (true/false)");
        boolean trabaja = Lector.leerBoolean(); //Lee y devuelve el boolean ingresado antes del enter
        System.out.println("Ingrese edad");
        int edad = Lector.leerInt();            //Lee y devuelve el int ingresado antes del enter
        System.out.println("Ingrese 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 Demo06Generador
{
    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 condición entre **()**
- Encerrar acción/es 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
```

Iteración post-condicional

```
do{
    acción/es
} while (condición)
```

do-while

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

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

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

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 (actualiza el índice).

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

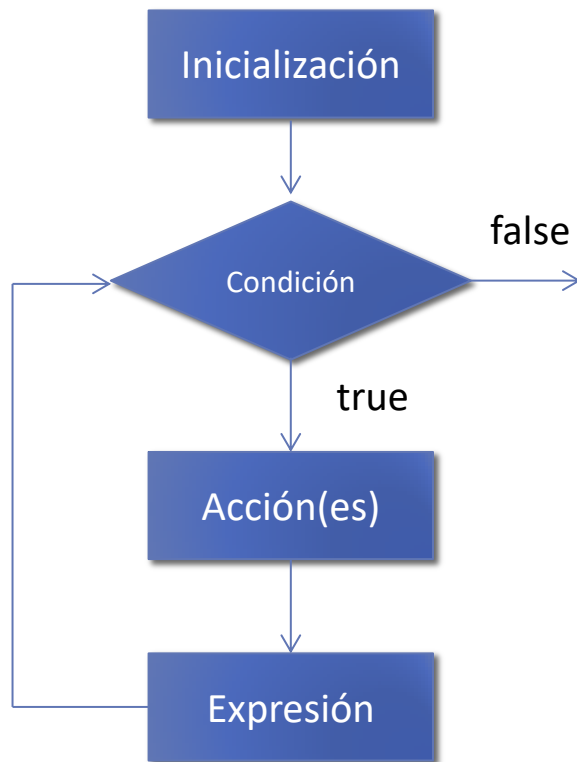
¿Qué imprime?

¿Modificar para imprimir pares?

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

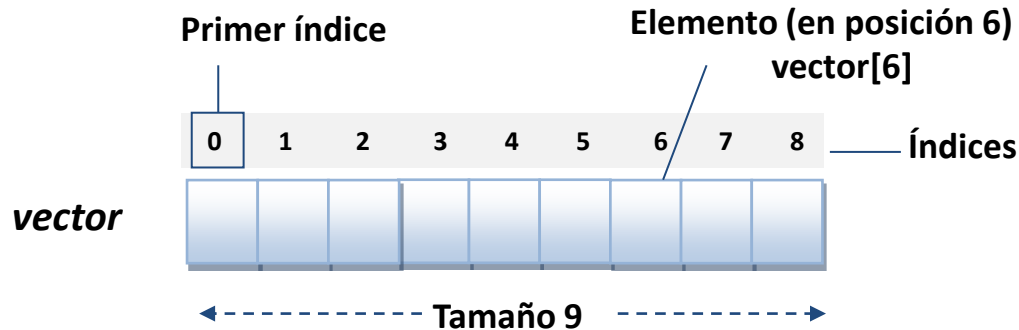
¿Qué imprime?

¿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:

“Cargar un vector que represente la tabla del 2”

```
public class Demo08Vector {  
    public static void main(String[] args) {  
        int DF=11;  
        int [] tabla2 = new int[DF]; //índices 0..10  
        for (int i=0; i<DF; i++)  
            tabla2[i]=2*i;  
        System.out.println("2x" + "5" + "=" + tabla2[5]);  
    }  
}
```

¿Y si quiero guardar en una estructura
todas las tablas del **0 al 10**?

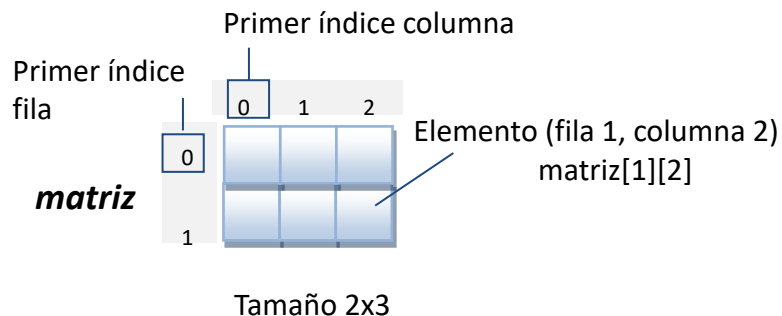
Arreglos bidimensionales - Matrices

Matriz:

- Arreglo de 2 dimensiones.
- Permite acceder a cada componente utilizando **dos índices (fila y columna)** para 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).

Arreglos bidimensionales - Matrices

• Ejemplo de situaciones de uso

- Representar todas las tablas de multiplicar.
- 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
- ...

x \ y	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	12	14	16	18	20
3	0	3	6	9	12	15	18	21	24	27	30
4	0	4	8	12	16	20	24	28	32	36	40
5	0	5	10	15	20	25	30	35	40	45	50
6	0	6	12	18	24	30	36	42	48	54	60
7	0	7	14	21	28	35	42	49	56	63	70
8	0	8	16	24	32	40	48	56	64	72	80
9	0	9	18	27	36	45	54	63	72	81	90
10	0	10	20	30	40	50	60	70	80	90	100



B I N G O					
7	25	44	57	62	
15	22	40	50	70	
11	30	FREE 8737 SPACE	46	74	
2	28	37	55	68	
10	27	39	59	75	

Arreglos bidimensionales - Matrices

- Declaración

```
TipoElemento [][] nombreVariable;
```

- Creación

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

- Acceso a elemento

```
nombreVariable[posFila][posColumna]
```

Ejemplo: “Cargar una matriz de 3x4 con nros. aleatorios”

```
public class Demo09Matrices {  
    public static void main(String[] args) {  
        int DIMF=3; int DIMC=4;  
        int[][] m = new int[DIMF][DIMC];  
        for (int i=0; i<DIMF; i++)  
            for(int j=0; j<DIMC; j++)  
                m[i][j] = GeneradorAleatorio.generarInt(10);  
    }  
}
```

Gráficamente

m

	0	1	2	3
0				
1				
2				

Tamaño 3x4

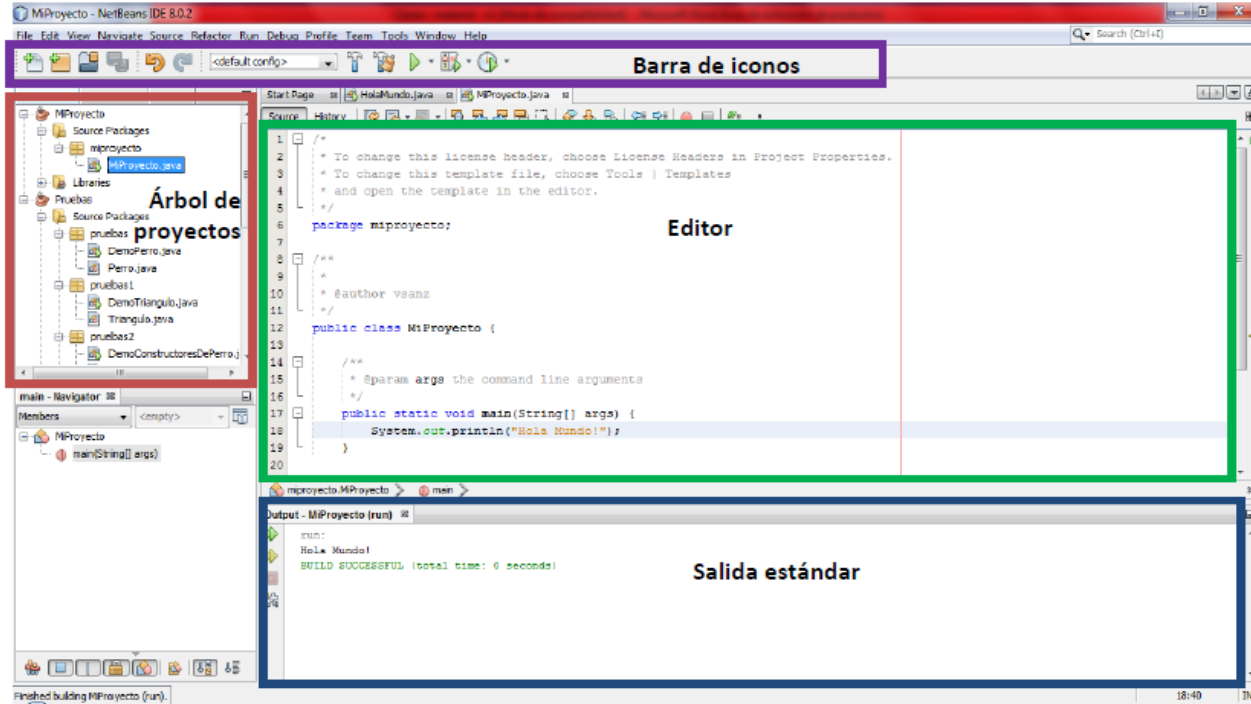
Pensar las operaciones

Imprimir el contenido de la matriz

Imprimir el contenido de una columna específica

Sumar los elementos de una fila específica

IDE NetBeans



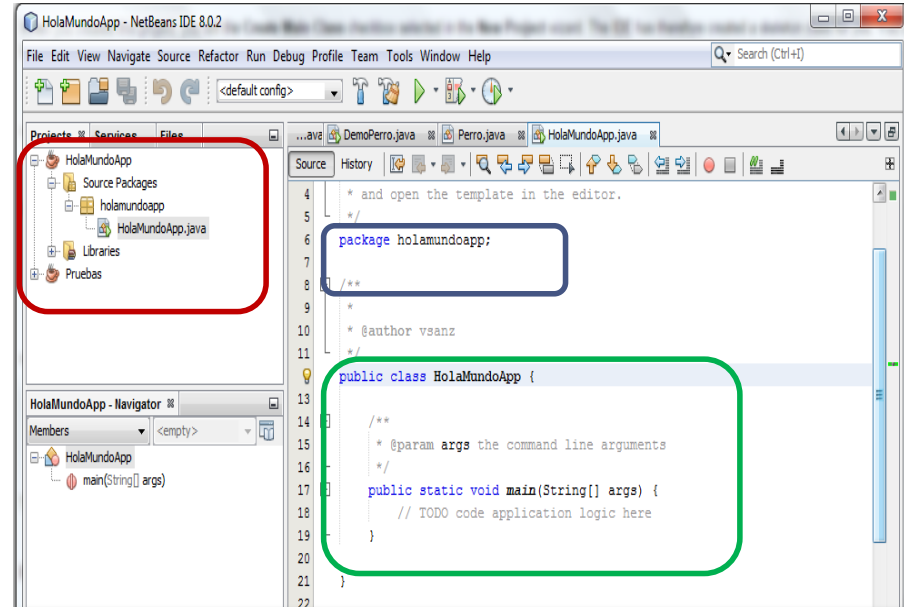
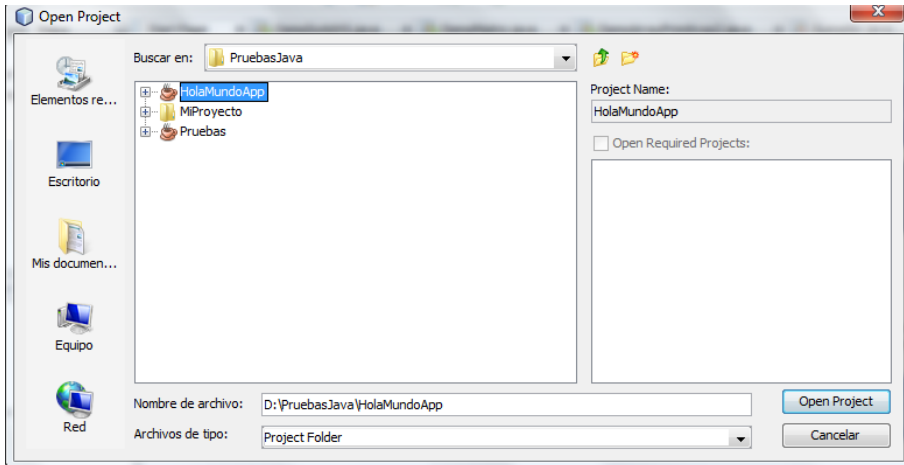
- Reúne herramientas para desarrollar SW.
 - Editor
 - Compilador
 - Depurador
 - ...
- Libre y gratuito
- Descargar desde la mediateca de Ideas

IDE NetBeans



Abrir Proyecto

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



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

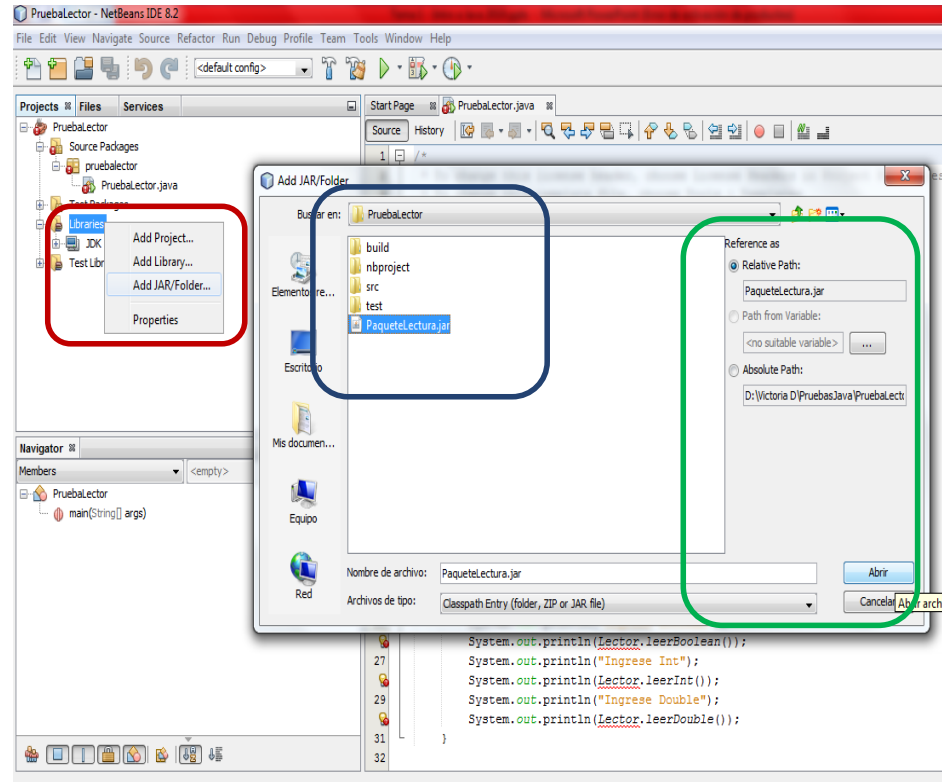
IDE NetBeans



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
- Click en “Abrir”

Realizar este paso cada vez que trabaje sobre un proyecto distinto



IDE NetBeans

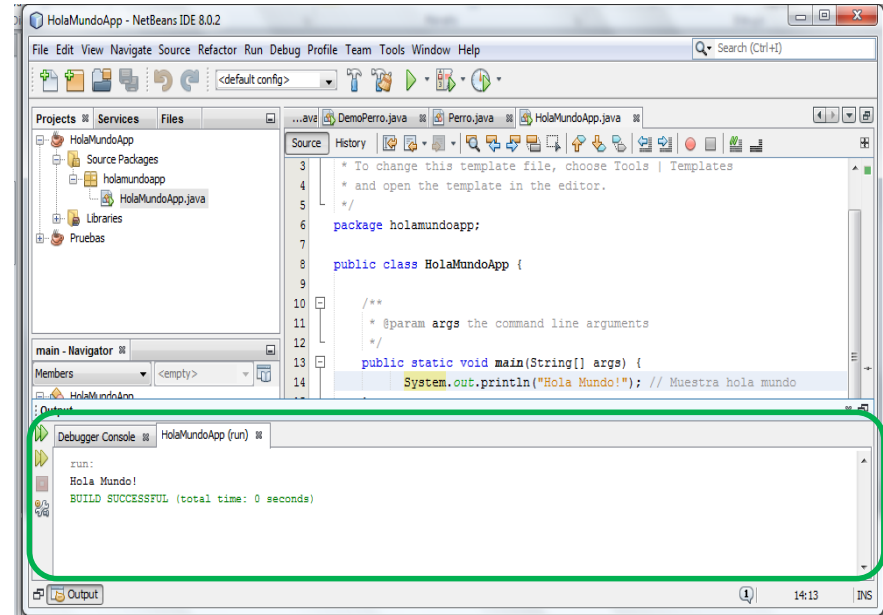


Ejecutar programa

- Click derecho sobre el archivo que contiene el programa (main)

Ej: Demo04Salida.java

- Run File.



IDE NetBeans



Crear Nuevo “Programa Principal”

- 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

- File → New Project →
 - *Categories*: Java *Projects*: Java Application
 - *Project Name*: Poner un nombre
 - *Project Location*: Seleccionar ubicación
- Finish