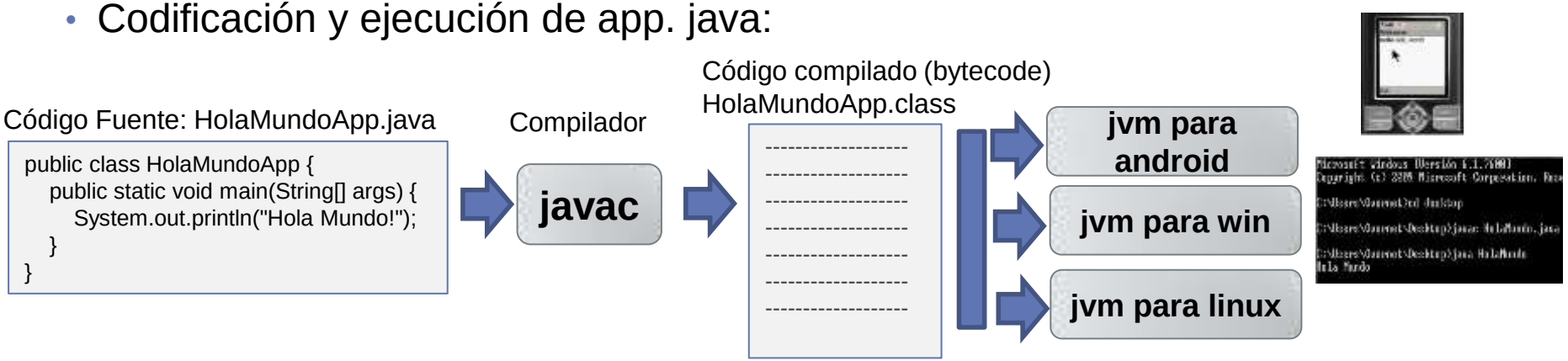
TEMA: INTRODUCCIÓN A JAVA

Taller de Programación.

Módulo: Programación Orientada a Objetos

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*

  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 Demo03CalculoAritmetico {
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
}
}
```

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

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
    int edad = in.nextInt();
    double peso = in.nextDouble();
    boolean tieneDueño = in.nextBoolean();
    String nombre = in.next();
    String descripcion = in.nextLine();
    in.close();   // Cerrar el scanner
  }
}
```

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()

Estructuras de control

Selección

```
if (condición)
   acción(es) a realizar cuando
   condición es true
else
   acción(es) a realizar cuando
   condición es false
```

Iteración pre-condicional

```
while (condición)
    acción(es) a realizar cuando
    condición es true
```

Iteración post-condicional

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

Encerrar entre {} en caso de incluir varias sentencias.

Cuando sólo incluye una sentencia, finalizarla con ;

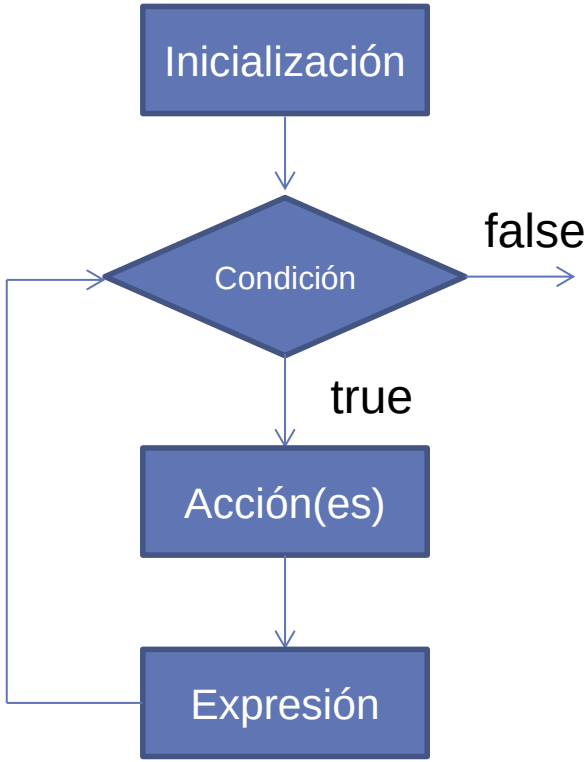
Diferencia do-while y while

- 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);
```

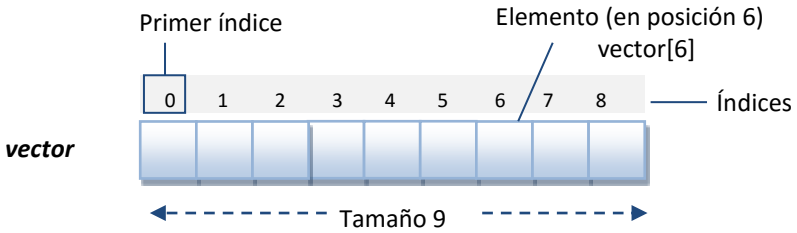
¿Qué imprime?
¿Modificar para imprimir pares?

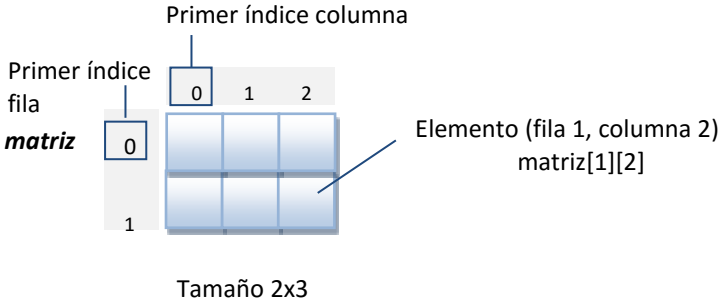
```
int i;
for (i=10; i > 0;  i=i-1)
    System.out.println(i);
```

¿Qué imprime?
¿Es lo mismo poner i-- ?

Arreglos

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





Arreglos

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]);
```

Matriz

- Declaración

TipoElemento [][] nombreVariable;

- Creación

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

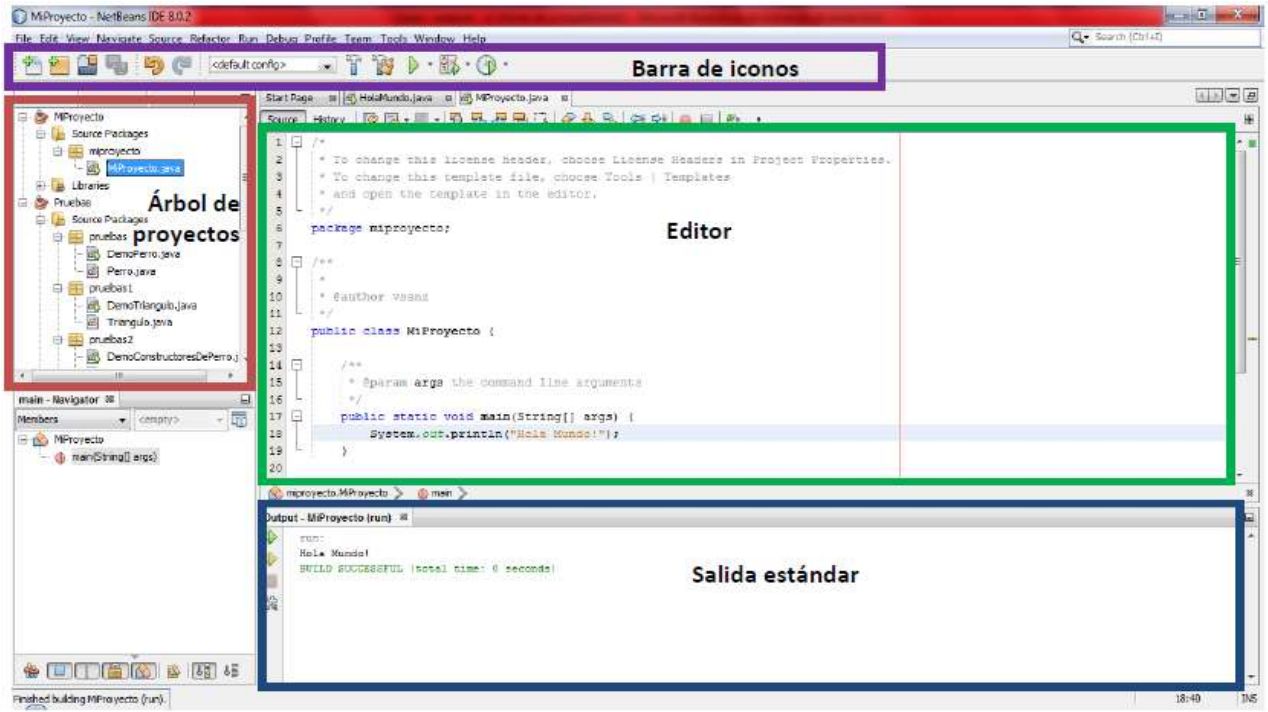
- Acceso a elemento

nombreVariable [posFil] [posCol]

- Ejemplo:

```
int [][] tabla = new int[3][2];
…
```

IDE NetBeans

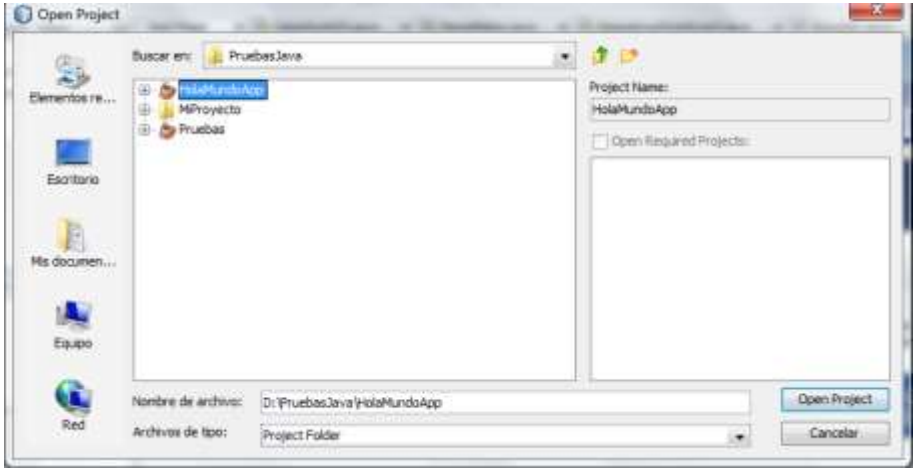


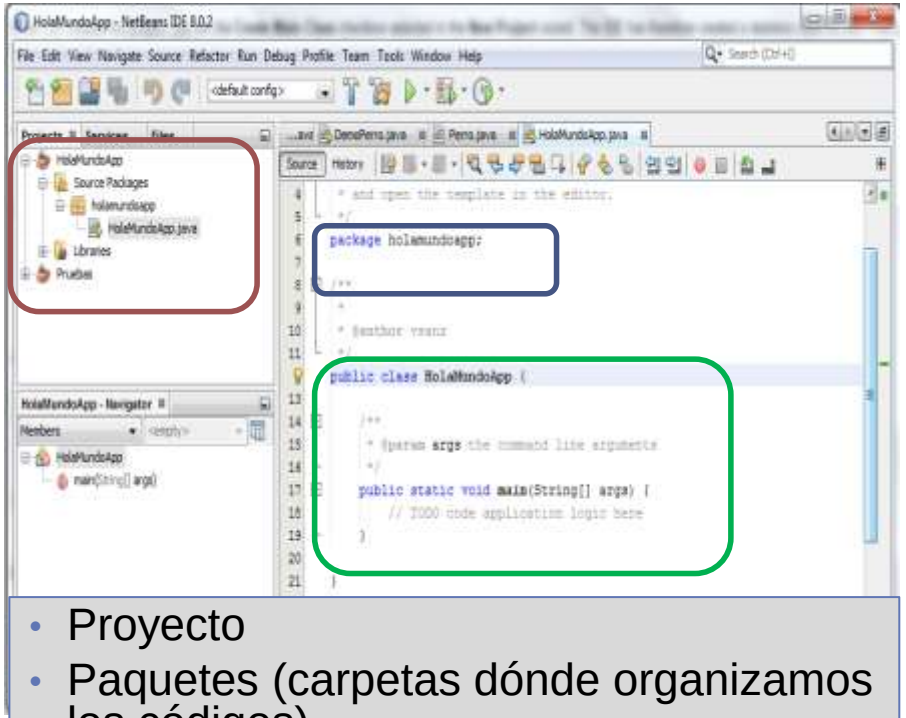
- Reúne herramientas para desarrollar SW.
  - Editor
  - Compilador
  - Depurador
  - …
- Libre y gratuito
- Descargar desde
  - http://www.oracle.com/technetwork/articles/javase/jdk-netbeans-jsp-142931.html

IDE NetBeans. Uso.

Abrir Proyecto

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



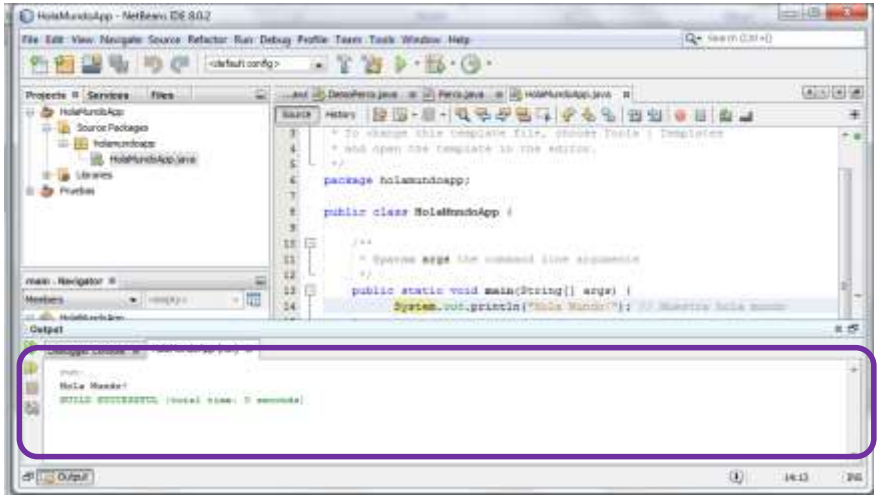


- Proyecto
- Paquetes (carpetas dónde organizamos los códigos)
- Códigos: extensión .java

IDE NetBeans. Uso.

Correr programa

- Pararse sobre el archivo que contiene el *main*.
  - Ej: Demo04Salida.java
- Menú contextual. Run File.



IDE NetBeans. Uso.

Crear nuevo “Prog Ppal”

- Pararse sobre la carpeta contenedora.
  - Ej: “tema 1”
- Menú contextual. New => Java Main Class => Poner un nombre
- Aparecerá un archivo .java con el esqueleto del programa ppal.

Cerrar Proyectos

- File. Close All Projects.