

# Programa Integral: Desarrollador de Aplicaciones Web con Java

Departamento de Informática

Copyright © 2012 por TECSUP

# Módulos del Programa Integral

1. **Java Básico** (36 horas)
2. Java Web (30 horas)
3. Java Web Avanzado (36 horas)

# Módulo 1:

## Java Básico

(36 horas)

# Unidad 1:

# Fundamentos del lenguaje

# Introducción

- En la presente unidad, se detalla los fundamentos de la tecnología Java, reconociendo las 3 plataformas que la conforman.
- Además, se revisa la sintaxis y fundamentos del lenguaje de Java.

# Objetivos

- Reconocer los componentes de la tecnología Java.
- Identificar el alcance de las plataformas de Java.
- Escribir programas usando la sintaxis del lenguaje Java.

# Índice

- La tecnología Java
  - La plataforma
  - El lenguaje Java
- El lenguaje de programación Java
  - Sintaxis básica
  - Estructuras de control
  - Estructuras repetitivas

# Tema 1: La tecnología Java



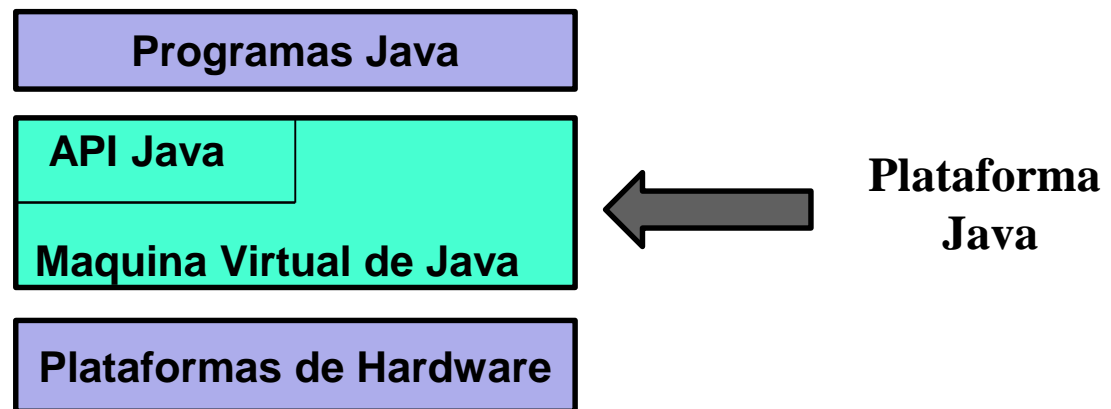
# Estadísticas (Mayo 2012)

Position May 2012	Position May 2011	Delta in Position	Programming Language	Ratings May 2012	Delta May 2011	Status
1	2	↑	C	17.346%	+1.18%	A
2	1	↓	Java	16.599%	-1.56%	A
3	3	=	C++	9.825%	+0.68%	A
4	6	↑↑	Objective-C	8.309%	+3.30%	A
5	4	↓	C#	6.823%	-0.72%	A
6	5	↓	PHP	5.711%	-0.80%	A
7	8	↑	(Visual) Basic	5.457%	+0.96%	A
8	7	↓	Python	3.819%	-0.76%	A
9	9	=	Perl	2.805%	+0.57%	A
10	11	↑	JavaScript	2.135%	+0.74%	A
11	10	↓	Ruby	1.451%	+0.03%	A
12	26	↑↑↑↑↑↑↑↑	Visual Basic .NET	1.274%	+0.79%	A
13	21	↑↑↑↑↑↑↑↑	PL/SQL	1.119%	+0.62%	A
14	13	↓	Delphi/Object Pascal	1.004%	-0.07%	A
15	15	=	Lisp	0.941%	-0.01%	A
16	24	↑↑↑↑↑↑↑↑	Logo	0.839%	+0.35%	A-
17	17	=	Pascal	0.808%	+0.10%	A
18	18	=	Transact-SQL	0.654%	-0.04%	A-
19	16	↓↓↓	Ada	0.649%	-0.10%	B
20	12	↓↓↓↓↓	Lua	0.566%	-0.54%	B

Fuente: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

# Plataforma Java

- La tecnología Java esta compuesta por dos partes principales:
1. La plataforma
    - La máquina virtual de Java (JVM)
    - El Java API (Aplication Programming Interface)
  2. El lenguaje de programación.



# El API

➤ On-line: <http://download.oracle.com/javase/7/docs/api/>

The screenshot shows the Java Platform, Standard Edition 7 API Specification website. The browser window is titled "Overview (Java Platform SE 7)". The address bar shows the URL [download.oracle.com/javase/7/docs/api/](http://download.oracle.com/javase/7/docs/api/). The page has a navigation bar with tabs: Overview, Package, Class, Use, Tree, Deprecated, Index, and Help. The main content area is titled "Java™ Platform, Standard Edition 7 API Specification" and includes a description: "This document is the API specification for the Java™ Platform, Standard Edition. See: Description".

Annotations with red arrows point to specific parts of the page:

- paquetes** (packages) points to the "Packages" section in the left sidebar, which lists packages like `java.applet`, `java.awt`, and `java.awt.color`.
- clases** (classes) points to the "All Classes" section in the left sidebar, which lists classes like `AbstractAction`, `AbstractAnnotationValueVisitor6`, and `AbstractAnnotationValueVisitor7`.
- atributos y métodos** (attributes and methods) points to the "Packages" table in the main content area, which lists packages and their descriptions.

Package	Description
<code>java.applet</code>	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
<code>java.awt</code>	Contains all of the classes for creating user interfaces and for painting graphics and images.
<code>java.awt.color</code>	Provides classes for color spaces.
<code>java.awt.datatransfer</code>	Provides interfaces and classes for transferring data between and within applications.
<code>java.awt.dnd</code>	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
<code>java.awt.event</code>	Provides interfaces and classes for dealing with different types of events fired by AWT components.
<code>java.awt.font</code>	Provides classes and interface relating to fonts.
<code>java.awt.geom</code>	Provides the Java 2D classes for defining and performing operations on objects

# Ediciones

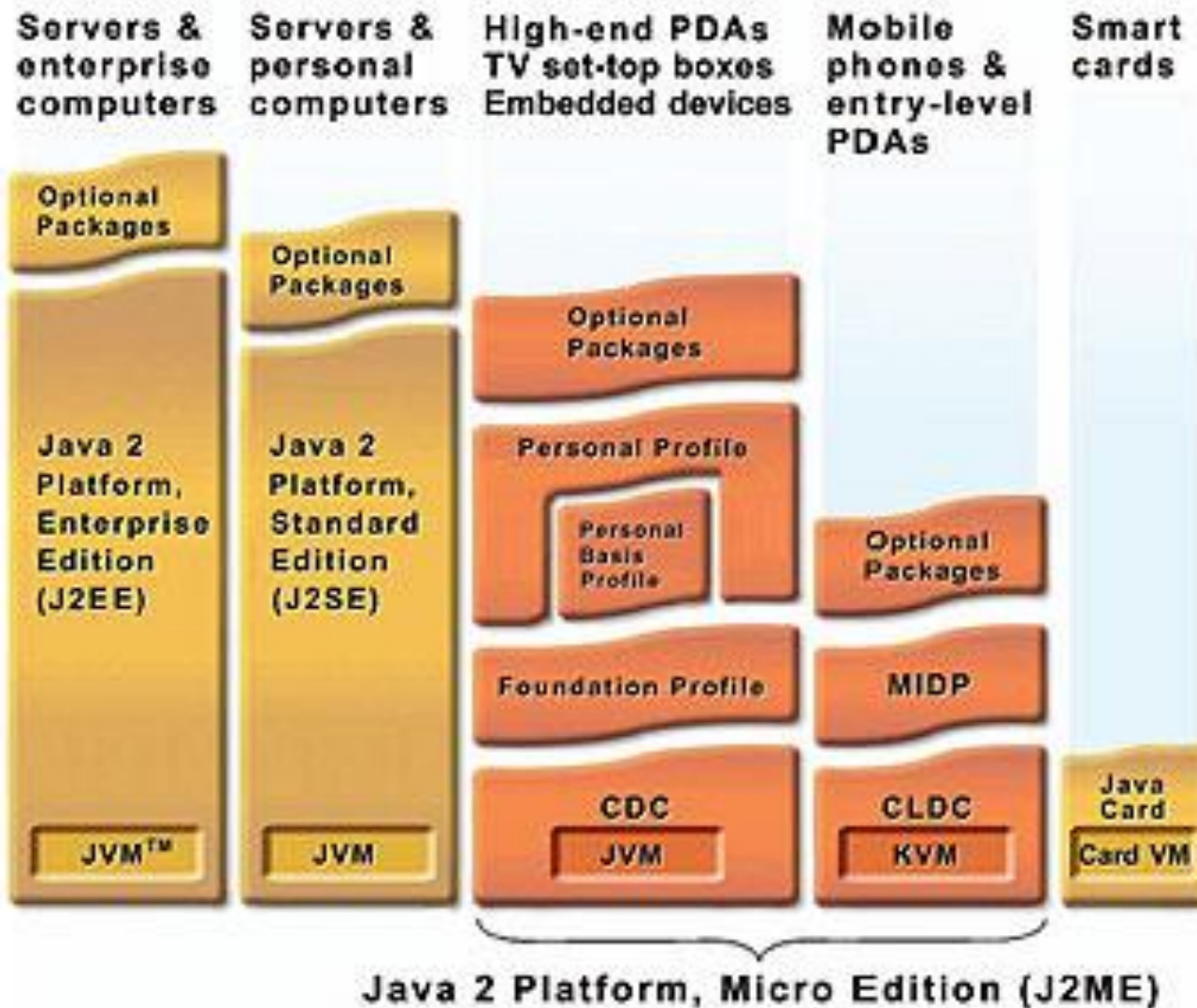
- La plataforma de Java está dividida en diferentes ediciones, entre ellas:
  - Java Standard Edition (Java SE)
  - Java Enterprise Edition (Java EE)
  - Java Micro Edition (Java ME)
  - Java Card

# El lenguaje de programación

- Creado por Sun Microsystems.
- Aparecio en 1990s.
- Difundido en 1995 con una nueva orientación hacia Internet.
- Paradigma: orientado a objetos.
- Sintaxis muy similar a la de C++.
- En el 2010, Oracle compró Sun Microsystems.

# 1. La plataforma

# Plataformas



# Java SE

## Java™ SE Platform at a Glance

JDK	Java Language		Java Language								Java SE API
	Tools & Tool APIs	java	javac	javadoc	apt	jar	javap	JPDA	JConsole		
		Security	Int'l	RMI	IDL	Deploy	Monitoring	Troubleshoot	Scripting	JVM TI	
	Deployment Technologies	Deployment		Java Web Start				Java Plug-in			
		AWT			Swing			Java 2D			
	User Interface Toolkits	Accessibility	Drag n Drop		Input Methods		Image I/O	Print Service	Sound		
		IDL	JDBC		JNDI		RMI	RMI-IIOP			
	Integration Libraries	Beans	Intl Support		Input/Output		JMX	JNI	Math		
		Networking	Override Mechanism		Security		Serialization	Extension Mechanism		XML JAXP	
	Other Base Libraries	lang and util	Collections	Concurrency Utilities		JAR	Logging	Management			
		Preferences API	Ref Objects	Reflection		Regular Expressions	Versioning	Zip	Instrumentation		
	Java Virtual Machine	Java Hotspot Client VM					Java Hotspot Server VM				
		Solaris		Linux		Windows			Other		

Java SE API

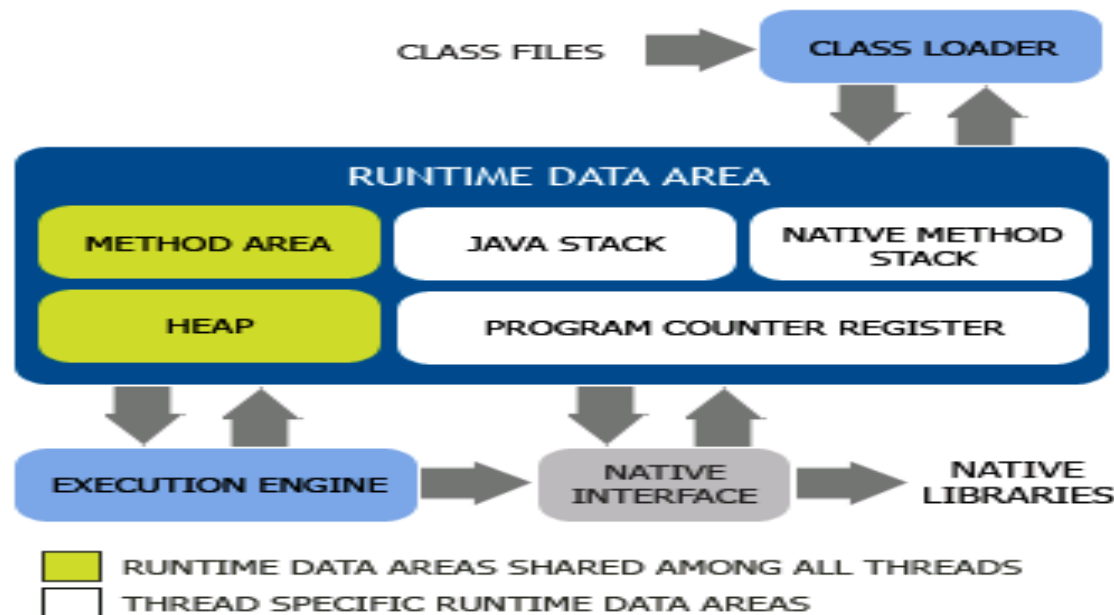


# ¿Qué es el JRE?

- JRE = Java Runtime Environment (Entorno de Ejecución de Java)
- Contiene las clases para ejecutar aplicaciones Java (varias en cada sistema operativo).
- Las clases apropiadas para cada aplicación son cargadas automáticamente.

# ¿Qué es la JVM?

- JVM = Java Virtual Machine (Maquina Virtual de Java)
- Está incluida en el JRE.
- Permite la ejecución de aplicaciones Java
  - Ejecuta el bytecode.



# ¿Qué es el JDK?

- El JDK incluye al JRE más las herramientas de desarrollo como los compiladores y debuggers que son necesarios para desarrollar aplicaciones.

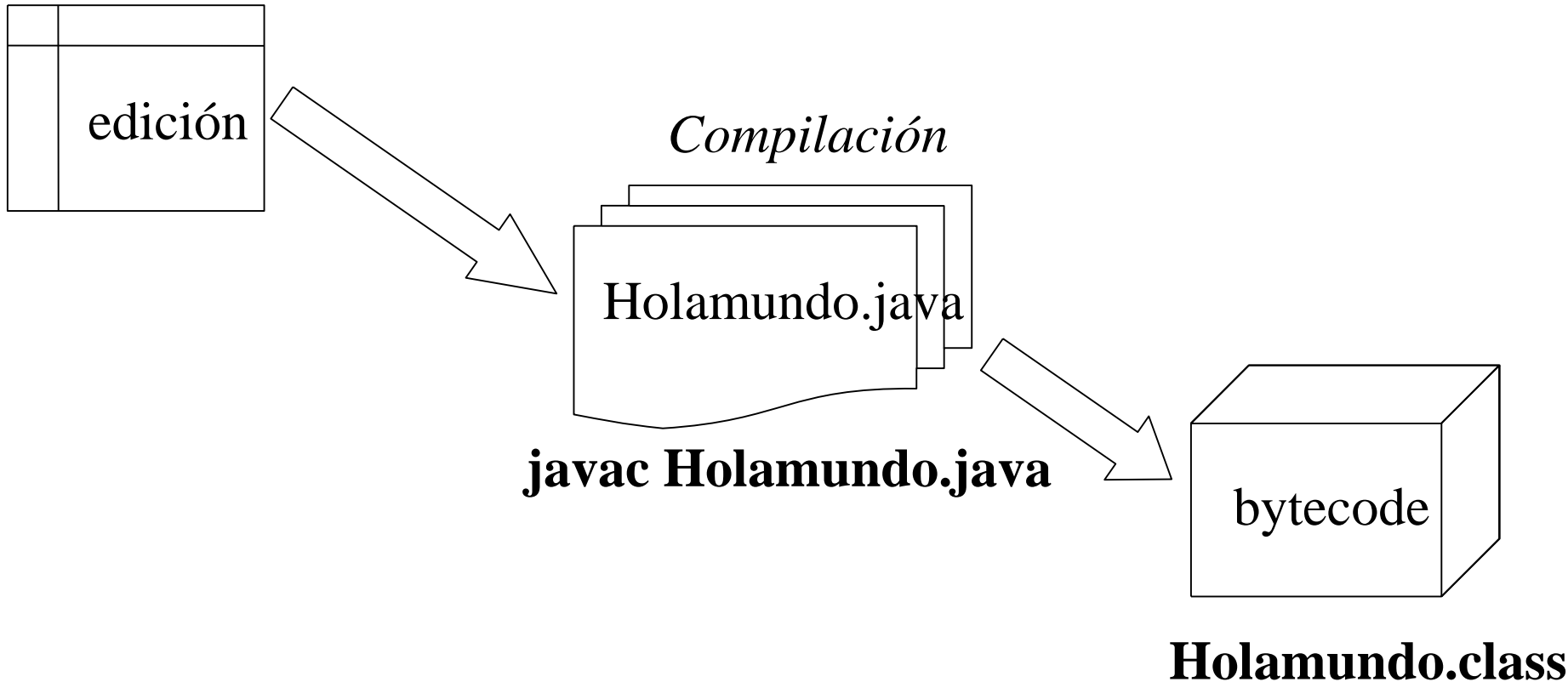
# ¿Qué es el Código Fuente?

- Instrucciones especiales almacenadas en un archivo.
- Escritas en un lenguaje específico (Java,C++,Visual Basic)
- Son convertidos en aplicaciones utilizando un compilador.
- Debe ser formateado correctamente, sin errores de sintaxis.

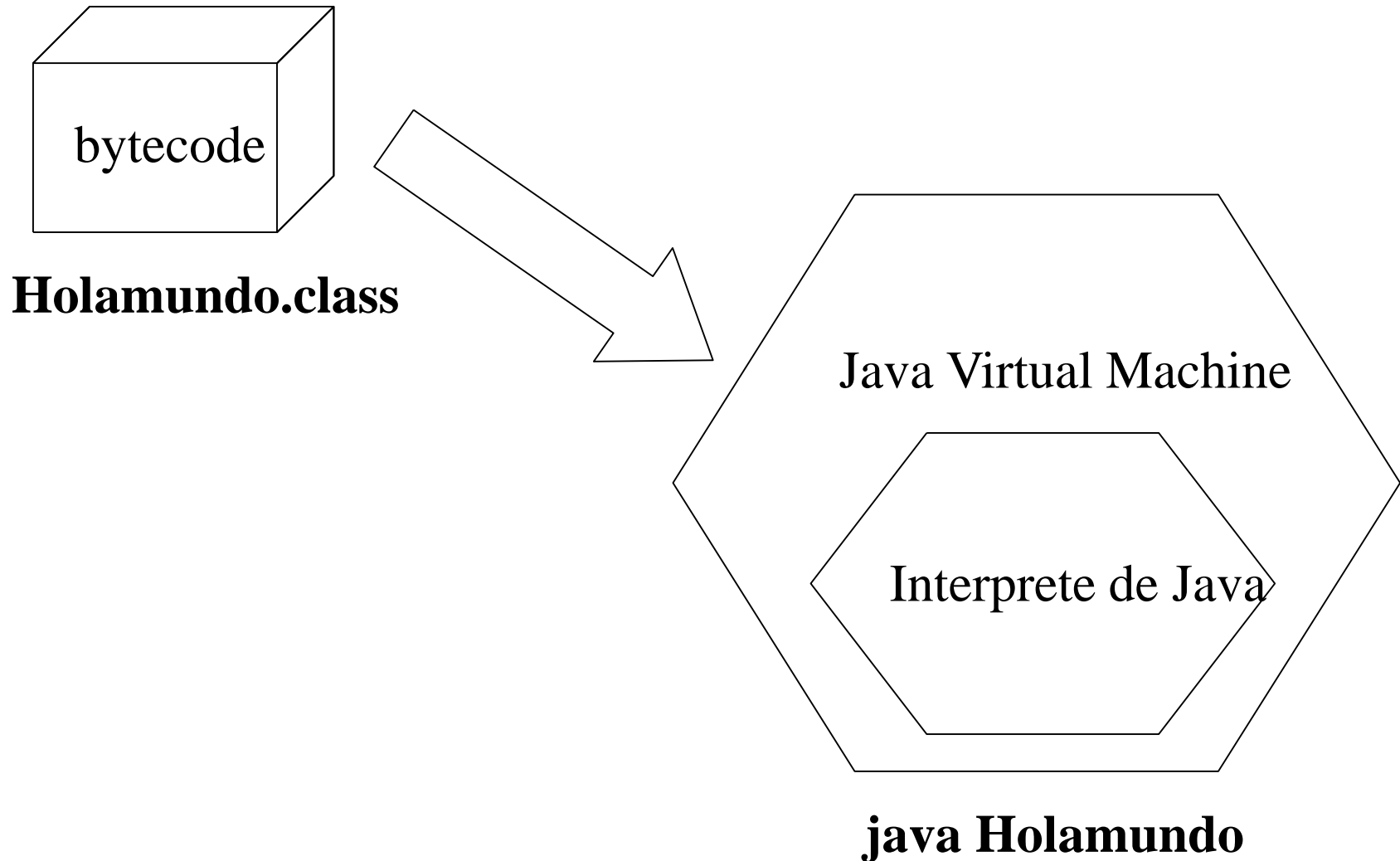
# ¿Qué es el Bytecode?

- Código Binario de las aplicaciones Java.
- Es interpretado por el JVM.
- Cada JVM (de diferentes sistemas) interpreta el mismo bytecode.
- Se ejecuta más despacio que si fuera compilado directamente un código nativo.

# Ambiente de Desarrollo



# Ambiente de Ejecución



# ¿Qué es una Aplicación Java?

- Programa Standalone
- Código fuente almacenado en un archivo de texto con extensión **.java**
- Compilado (javac) para obtener clases java (con extensión **.class**)
- Las Clases son cargadas utilizando el interprete de java (java)



# Java SE 7

- Liberado el 7 de Julio de 2011.
- Se han corregido 9,494 bugs e implementado 1,966 mejoras y 9,018 cambios.
- Permite utilizar otros lenguajes de programación (JPython, JRuby).
- Gestión automática de recursos y de memoria.
- Compatibilidad con el estándar UNICODE 6.0.
- Documentación en:

<http://download.oracle.com/javase/7/docs/>

## 2. El lenguaje de programación

# Características del lenguaje

- Sencillo.
- Orientado a Objetos.
- Distribuido.
- Interpretado.
- Robusto.
- Seguro.
- Arquitectura neutra.
- Portable.
- Multithread.

# Características del lenguaje (Sencillo)

- Los creadores de java se basaron en C++, pero eliminaron la mayoría de sus complejidades.
- No soporta tipo de datos: struct, union, y puntero.
- No soporta typedef ni #define.
- No permite la sobrecarga de operadores.
- No soporta la herencia multiple.
- Posee una clase String, en vez del array de tipo char[] finalizado con nulo.
- Cuenta con un sistema automático para asignar y liberar memoria: el Garbage Collector.

# Características del lenguaje (Orientado a objetos)

- Java es un lenguaje para desarrollo de software Orientado a Objetos.
- Implementa los siguientes conceptos O.O.:
  - Herencia.
  - Encapsulación.
  - Abstracción.
  - Reutilización.
  - Polimorfismo

# Características del lenguaje (Distribuido)

- Está concebido para trabajar en un entorno conectado a red.
- Cuenta con una enorme biblioteca de clases para comunicarse mediante TCP/IP: HTTP, FTP.
- Permite manipular con gran facilidad recursos vía URL.

# Características del lenguaje (Interpretado)

- El compilador de Java traduce el código fuente a código intermedio (bytecode)
- Los bytecodes son interpretados (ejecutados) en cualquier entorno donde exista un intérprete de Java.
- El intérprete de Java se llama Máquina Virtual Java o Java Virtual Machine (JVM).
- De ahí el famoso: Write once, run everywhere.

# Características del lenguaje (Robusto)

- Un software robusto es aquel que no se 'interrumpe' fácilmente a consecuencia de fallos.
- Un lenguaje de estas características suele tener mas restricciones a la hora de programar.
- No permite sobrescribir memoria y corromper otros datos mediante punteros.
- Facilita el manejo de excepciones.



# Características del lenguaje (Seguro)

- Por su naturaleza distribuida, el tema de la seguridad es muy crítico.
- Contiene una API para encriptación de datos.
- Existen tecnologías de firma digital para confiar en un determinado código Java.
- A su vez, existen políticas de seguridad para controlar de una manera precisa que puede o no puede hacer.

# Características del lenguaje (Portable)

- Los bytecodes son interpretados en cualquier plataforma donde exista una JVM.
- El uso de estándares como UNICODE, IEEE 754 (Standard for Binary Floating-Point Arithmetic), etc, permite obtener los mismos resultados en todas las plataformas.

# Características del lenguaje (Multithread)

- Soporta la ejecución de varias tareas a la vez.
- Posee una serie de clases que facilitan su utilización.
- También conocido con el término en castellano: 'multihilo'.

# Características del lenguaje (Dinámico)

- El código C++, a menudo requiere una recompilación completa si cambia una clase.
- Java emplea un método de interfaces para evitar estas dependencias y recompilaciones.

# Java vs. C

Característica	Java	C
Independiente de Plataforma	SI	NO
Orientado a Objetos	SI	NO
Uso de Punteros	NO	SI
Acceso Directo a Memoria	NO	SI

# Enlaces de Java

- <http://www.java.sun.com>
- <http://www.javahispano.org>
- <http://www.theserverside.com>
- [http:// www.jguru.com](http://www.jguru.com) (FAQ)
- [http:// www.ibm.com/java](http://www.ibm.com/java) (librerías y proyectos)
- [http:// www.javapassion.org.com](http://www.javapassion.org.com) (artículos)
- [http:// www.javaworld.com](http://www.javaworld.com) (artículos)
- [http:// www.roseindia.net](http://www.roseindia.net) (artículos y utilitarios)
- [http:// www.javaranch.com](http://www.javaranch.com) (foros)

# Tema 2: Sintaxis del lenguaje Java

# Comentarios

- Existen tres formas distintas de escribir comentarios:

// Comentario en una linea

/\* Comentario de una o

mas lineas \*/

/\*\* Comentario de documentación,

utilizado por la herramienta javadoc.exe

\*/



# Puntos y coma, bloques y espacios en blanco

- Una sentencia es una línea simple de código terminada en punto y coma.

```
system.out.println("Hola curso");
```

- Un bloque es un conjunto de sentencias agrupadas entre llaves ({}):

```
while(true)
{
    x=y+1;
    x=x+1;
}
```

# Puntos y coma, bloques y espacios en blanco

- Los bloques pueden estar anidados.

```
while(true)
```

```
{
```

```
    x=y+1;
```

```
    if(x<0)
```

```
    {
```

```
        x=x+1;
```

```
    }
```

```
}
```

- Java permite los espacios en blanco entre elementos de código fuente.

# Identificadores

- Son los nombres unívocos que se le dan a las clases, métodos y variables.
- Hay que tener presente las siguientes reglas:
  - Deben empezar por una letra, subrayado(\_) o dólar(\$).
  - Déspues del primer carácter pueden usar números.
  - Distinguen las mayúsculas y minúsculas.
  - Nunca pueden coincidir con una 'keyword'

# Keyword

boolean	byte	char	double	float
int	long	short	public	private
protected	abstract	final	native	static
synchronized	transient	volatile	if	else
do	while	switch	case	default
for	break	continue	assert	class
extends	implements	import	instanceof	interface
new	package	super	this	catch
finally	try	throw	throws	return
void	null	enum	true	false

# Ejemplos de identificadores

➤ Estos identificadores serían validos:

Identificador

nombreUsuario

nombre\_usuario

\_sys\_var2

\$cambio

if2

# Variables

- Una variable es un contenedor de datos identificado mediante un nombre (identificador)
- Dicho identificador se utilizará para referenciar el dato que contiene.
- Toda variable debe llevar asociado un tipo que describe el tipo de dato que guarda.
- Por tanto, una variable tiene:  
Un tipo, un identificador y un dato(o valor).

# Declaración de variables

- Es la sentencia mediante la cual se define una variable, asignándole un tipo y un identificador:

*tipo identificador;*

**int** contador;

- Adicionalmente se le puede asignar un valor inicial mediante una asignación:

*tipo identificador = valor;*

**int** contador = 10;

- Si no se le asigna un valor, se inicializará con el valor por defecto para ese tipo.

# Tipo de dato

- En java existen dos tipos de datos genéricos:
  - Tipos primitivos.
  - Tipos complejos: clases.
- Existen ocho tipos de datos primitivos clasificados en cuatro grupos diferentes:
  - Lógico: *boolean*.
  - Carácter: *char*.
  - Números enteros: *byte, short, int y long*.
  - Números reales: *double y float*.



# Tipo de dato primitivo

Dato Primitivo	Tamaño	Minimo	Maximo	Valor Default
<i>boolean</i>	--	--	--	false
<i>char</i>	16 bits	Unicode 0	Unicode $2^{16}-1$	null
<i>byte</i>	8 bits	-128	+127	0
<i>short</i>	16 bits	$-2^{15}$	$+2^{15}-1$	0
<i>int</i>	32 bits	$-2^{31}$	$+2^{31}-1$	0
<i>long</i>	64 bits	$-2^{63}$	$+2^{63}-1$	0
<i>float</i>	32 bits	IEEE <sub>754</sub>	IEEE <sub>754</sub>	0.0
<i>double</i>	64 bits	IEEE <sub>754</sub>	IEEE <sub>754</sub>	0.0

# Tipo de dato primitivo

- Binary integral literals and underscores in numbers



```
int billion = 1_000_000_000;  
int binary = 0b1001_1001;  
long bytes = 0b11010010_01101001_10010100_10010010;  
long creditCardNumber = 1234_5678_9012_3456L;  
long socialSecurityNumber = 1977_05_18_3312L;  
float pi = 3.14_15F;  
long hexBytes = 0xFF_EC_DE_5E;  
long hexWords = 0xCAFE_BABE;  
long maxLong = 0x7fff_ffff_ffff_ffffL;
```

# Tipo de dato complejo

- La 'Keyword' es el nombre de la clase del objeto que va a contener la variable.

- Posibles valores:

Referencias a objetos (o instancias) en memoria.

- Su valor por defecto es `null`.

- Ejemplos:

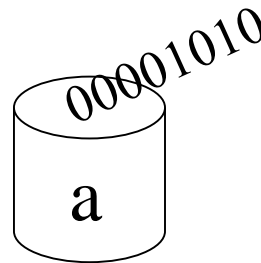
```
String unString = new String("Hola curso");
```

```
String otroString;
```

# Variable primitivas vs. complejas

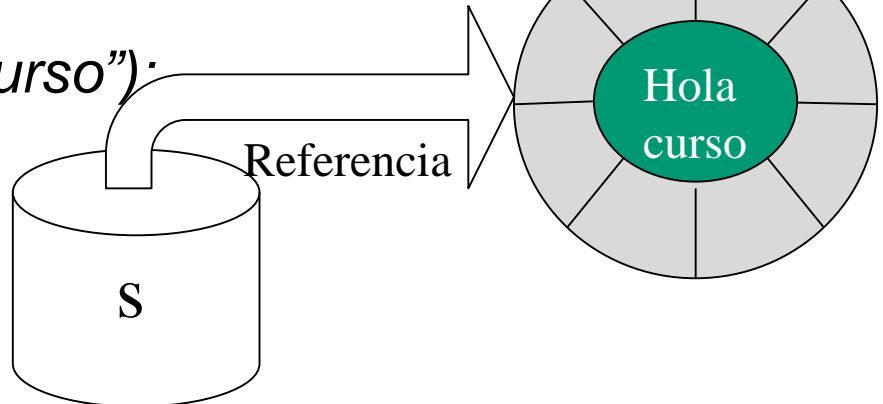
- Una variable de tipo primitivo contiene el dato directamente

**byte** a = 10;



- Una variable de tipo complejo contiene una referencia a la zona de memoria donde está

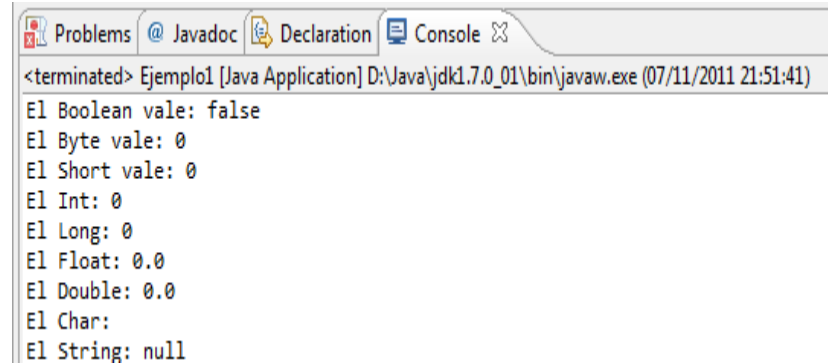
*String* s = **new** *String*("Hola curso");



# Ejemplo

```
public class VariablesTest1 {  
    static boolean unBoolean;  
    static byte unByte;  
    static short unShort;  
    static int unInt;  
    static long unLong;  
    static float unFloat;  
    static double unDouble;  
    static char unChar;  
    static String unString;
```

```
    public static void main(String[] args) {  
        System.out.println("El Boolean vale: " + unBoolean);  
        System.out.println("El Byte vale: " + unByte);  
        System.out.println("El Short vale: " + unShort);  
        System.out.println("El Int: " + unInt);  
        System.out.println("El Long: " + unLong);  
        System.out.println("El Float: " + unFloat);  
        System.out.println("El Double: " + unDouble);  
        System.out.println("El Char: " + unChar);  
        System.out.println("El String: " + unString);  
    }  
}
```



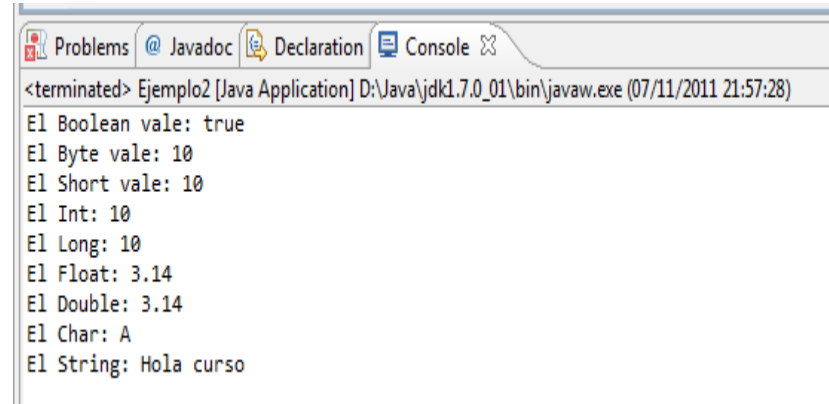
The screenshot shows a Java IDE window with the 'Console' tab selected. The title bar indicates the application is 'Ejemplo1 [Java Application]' running at 'D:\Java\jdk1.7.0\_01\bin\javaw.exe (07/11/2011 21:51:41)'. The console output displays the values of the static variables: 'El Boolean vale: false', 'El Byte vale: 0', 'El Short vale: 0', 'El Int: 0', 'El Long: 0', 'El Float: 0.0', 'El Double: 0.0', 'El Char:', and 'El String: null'.

```
<terminated> Ejemplo1 [Java Application] D:\Java\jdk1.7.0_01\bin\javaw.exe (07/11/2011 21:51:41)  
El Boolean vale: false  
El Byte vale: 0  
El Short vale: 0  
El Int: 0  
El Long: 0  
El Float: 0.0  
El Double: 0.0  
El Char:  
El String: null
```

# Ejemplo

```
public class VariablesTest2 {  
    boolean unBoolean = true;  
    byte unByte = 10;  
    short unShort = 10;  
    int unInt = 10;  
    long unLong = 10;  
    float unFloat = 3.14F;  
    double unDouble = 3.14;  
    char unChar = 'A';  
    String unString = new String("Hola curso");
```

```
    public static void main(String[] args) {  
        System.out.println("El Boolean vale: " + unBoolean);  
        System.out.println("El Byte vale: " + unByte);  
        System.out.println("El Short vale: " + unShort);  
        System.out.println("El Int: " + unInt);  
        System.out.println("El Long: " + unLong);  
        System.out.println("El Float: " + unFloat);  
        System.out.println("El Double: " + unDouble);  
        System.out.println("El Char: " + unChar);  
        System.out.println("El String: " + unString);  
    }  
}
```



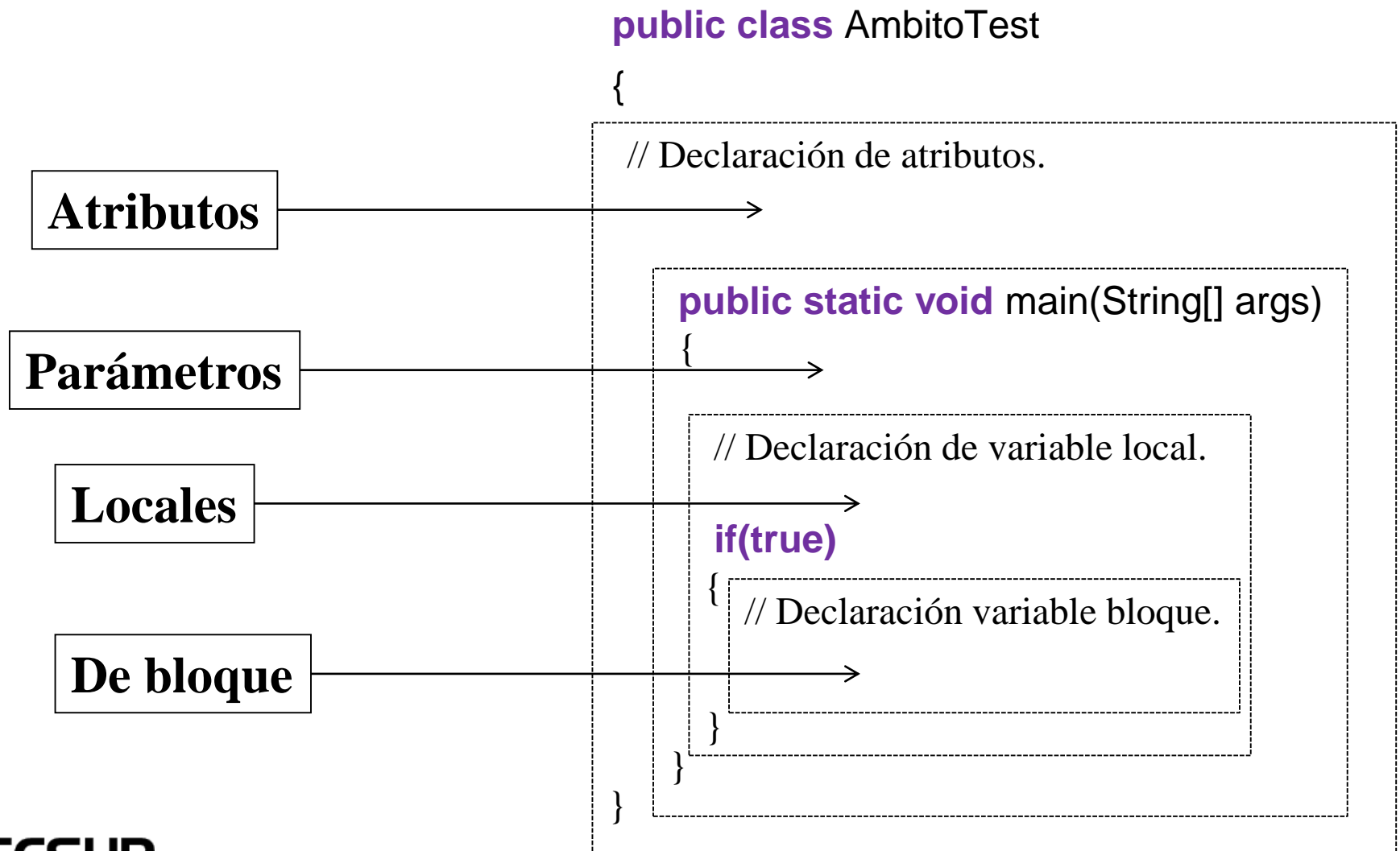
The screenshot shows a Java IDE window with the 'Console' tab selected. The title bar indicates the application is 'Ejemplo2 [Java Application]' running at 'D:\Java\jdk1.7.0\_01\bin\javaw.exe (07/11/2011 21:57:28)'. The console output displays the values of the variables defined in the code: 'El Boolean vale: true', 'El Byte vale: 10', 'El Short vale: 10', 'El Int: 10', 'El Long: 10', 'El Float: 3.14', 'El Double: 3.14', 'El Char: A', and 'El String: Hola curso'.

```
<terminated> Ejemplo2 [Java Application] D:\Java\jdk1.7.0_01\bin\javaw.exe (07/11/2011 21:57:28)  
El Boolean vale: true  
El Byte vale: 10  
El Short vale: 10  
El Int: 10  
El Long: 10  
El Float: 3.14  
El Double: 3.14  
El Char: A  
El String: Hola curso
```

# Ámbito de las variables

- El ámbito de una variable es la zona de código donde se puede referenciar dicha variable a través de su identificador.
- El lugar de definición de una variable establece su ambito.
- Ámbitos:
  - Atributos (o variables miembro).
  - Parámetros de métodos.
  - Variables locales: siempre hay que inicializarlas.
  - Variables de bloque: siempre hay que iniciarlas.

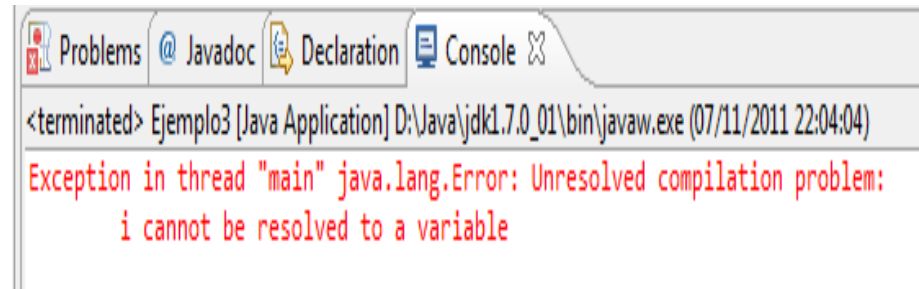
# Ámbito de las variables





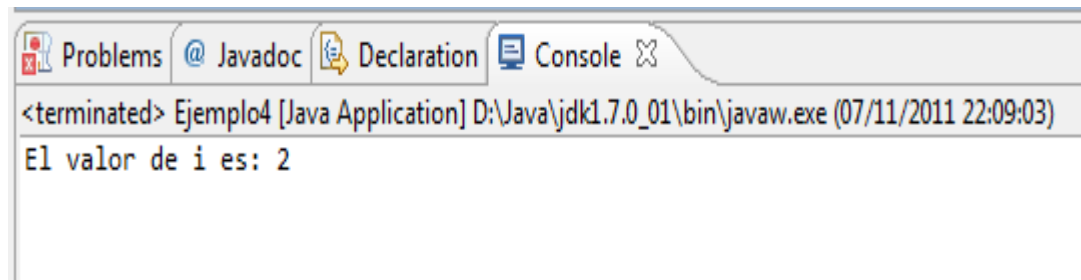
# Ejemplo

```
public class AmbitoTest2 {  
  
    public static void main(String[] args) {  
  
        if(true)  
        {  
            int i = 2;  
        }  
        System.out.println("El valor de i es: " + i);  
    }  
  
}
```



# Ejemplo

```
public class AmbitoTest3 {  
  
    public static void main(String[] args) {  
  
        if(true)  
        {  
            int i = 2;  
            System.out.println("El valor de i es: " + i);  
        }  
    }  
}
```



# Conversiones entre tipos

- En Java es posible transformar el tipo de una variable u objeto en otro diferente al original con el que fue declarado.
- Por norma las conversiones de tipo destino siempre deben ser igual o mayor que el tipo fuente.

Tipo Origen	Tipo Destino
<i>byte</i>	<i>double, float, long, int, char, short</i>
<i>short</i>	<i>double, float, long, int</i>
<i>char</i>	<i>double, float, long, int</i>
<i>int</i>	<i>double, float, long</i>
<i>long</i>	<i>double, float</i>
<i>float</i>	<i>double</i>

- El dato de tipo primitivo “boolean” es al único que no se le puede hacer cast.

# Ejercicio

➤ Identificar que sentencias son correctas y cuáles no:

1.int x = 34.5:

2.boolean boo = x:

3.int g = 17:

4.int y = g;

5.y = y + 10;

6.short s;

7.s = y;

8.byte b = 3;

9.byte v = b;

10.short n = 12;

11.v = n;

12.byte k = 128;

13.int p = 3\*g+y;

# Ejercicio (solución)

1. `int x = 34.5;` → `int x = (int)34.5;`
2. `boolean boo = x;` → no hay solución
3. `int g = 17;`
4. `int y = g;`
5. `y = y + 10;`
6. `short s;`
7. `s = y;` → `s = (short)y`
8. `byte b = 3;`
9. `byte v = b;`
10. `short n = 12;`
11. `v = n;` → `v = (byte)n`
12. `byte k = 128;` → `byte k = (byte)128`
13. `int p = 3*g+y;`

# Ejemplo

// Casting y redondeo de un dato float o double

```
public class Ejemplo5b {
```

```
    public static void main(String[] args) {
```

```
        double above = 0.7, below = 0.4;
```

```
        float fabove = 0.7f, fbelow = 0.4f;
```

```
        System.out.println("(int)above: " + (int)above);
```

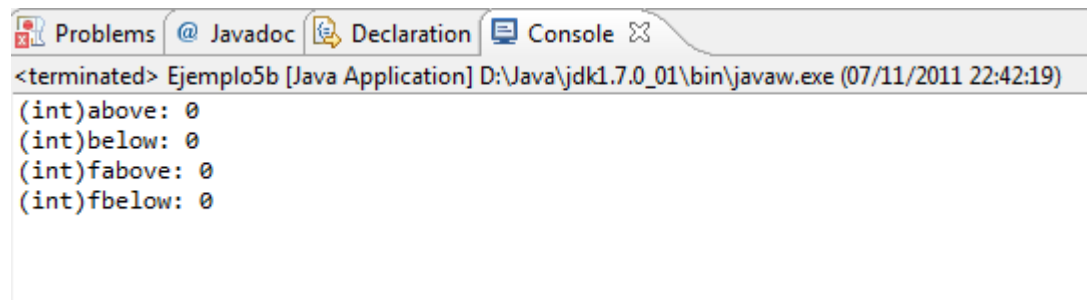
```
        System.out.println("(int)below: " + (int)below);
```

```
        System.out.println("(int)fabove: " + (int)fabove);
```

```
        System.out.println("(int)fbelow: " + (int)fbelow);
```

```
    }
```

```
}
```



The screenshot shows a Java IDE window with a tab labeled "Console". The console output displays the results of the program execution, showing that the integer casts of the floating-point values 0.7 and 0.4 result in 0.

```
<terminated> Ejemplo5b [Java Application] D:\Java\jdk1.7.0_01\bin\javaw.exe (07/11/2011 22:42:19)  
(int)above: 0  
(int)below: 0  
(int)fabove: 0  
(int)fbelow: 0
```

# Operadores

- Los operadores realizan funciones sobre uno, dos o tres operandos, por tanto tenemos:

OPERADORES	EJEMPLO
Unitarios	contador++
Binarios	contador + 1
Ternarios	$i < 10 ? i * 100 : i * 10$

- Los operadores siempre devuelven un valor que depende del operador y el tipo de los operandos.

# Operadores

- Los operadores se pueden dividir en las siguientes categorías:
  - Aritméticos
  - Relacionales
  - Condicionales
  - De desplazamiento
  - Lógicos
  - De asignación
  - Otros



# Operadores aritméticos

➤ Tenemos los siguientes operadores aritméticos:

OPERADOR	DESCRIPCIÓN	OBSERVACIÓN
+	Suma dos operandos (op1 + op2)	En el caso de Strings concatena
-	Resta dos operandos (op1 - op2)	
*	Multiplica dos operandos (op1 * op2)	
/	Divide dos operandos (op1 / op2)	
%	Calcula el resto de la división (op1 % op2)	

# Operandos aritméticos

➤ ¿De que tipo es el valor que devuelven?:

TIPO DATO	DESCRIPCIÓN
<i>long</i>	Cuando ninguno de los operandos es float o double y hay al menos uno que es long
<i>int</i>	Cuando ninguno de los operandos es float, double o long.
<i>double</i>	Cuando al menos uno de los operandos es double
<i>float</i>	Cuando ninguno de los operandos es double y hay al menos uno que es float

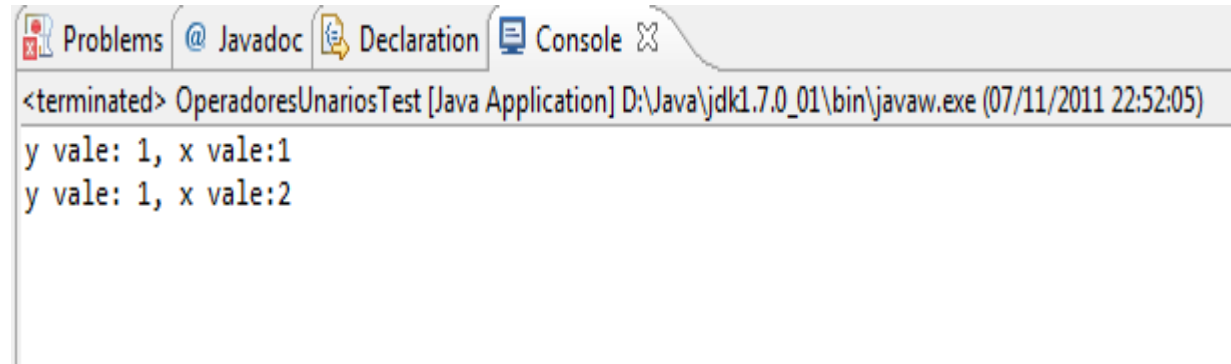
# Operadores aritméticos

➤ También existen operadores aritmeticos unitarios:

OPERADOR	DESCRIPCION
<i>+op</i>	Convierten a op en int en caso de que fuese byte, short o char
<i>-op</i>	Cambia el signo a op.
<i>++op</i>	Incrementa op en 1 (evaluando op despues de incrementarse)
<i>op++</i>	Incrementa op en 1 (evaluando op antes de incrementarse)
<i>--op</i>	Decrementa op en 1 (evaluando op despues de decrementarse)
<i>op--</i>	Decrementa op en 1 (evaluando op antes de decrementarse)

# Ejemplo

```
public class OperadoresUnariosTest {  
  
    public static void main(String[] args) {  
        int x = 0;  
        int y = 0;  
        y = ++x;  
        System.out.println("y vale: " + y + ", x vale:" + x);  
        y = x++;  
        System.out.println("y vale: " + y + ", x vale:" + x);  
    }  
}
```



The screenshot shows a Java IDE console window with the following tabs: Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of the program. The output consists of two lines: "y vale: 1, x vale:1" and "y vale: 1, x vale:2". The console title bar indicates the application is "OperadoresUnariosTest [Java Application]" and the command used is "D:\Java\jdk1.7.0\_01\bin\javaw.exe (07/11/2011 22:52:05)".

```
<terminated> OperadoresUnariosTest [Java Application] D:\Java\jdk1.7.0_01\bin\javaw.exe (07/11/2011 22:52:05)  
y vale: 1, x vale:1  
y vale: 1, x vale:2
```

# Operadores relacionales

➤ Tenemos los siguientes operadores relacionales:

OPERADOR	DESCRIPCION
>	Compara si un operando es mayor que otro (op1 > op2)
<	Compara si un operando es menor que otro (op1 < op2)
==	Compara si un operando es igual que otro (op1 == op2)
!=	Compara si un operando es diferente que otro (op1 != op2)
>=	Compara si un operando es mayor o igual que otro (op1 >= op2)
<=	Compara si un operando es menor o igual que otro (op1 <= op2)

# Operadores condicionales

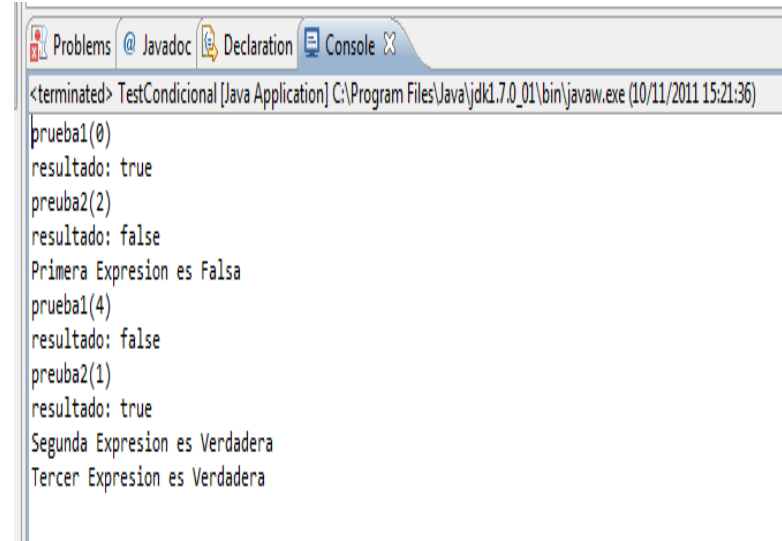
- Suelen combinarse con los relacionales para crear expresiones mas complejas.
- Tenemos los siguientes operadores condicionales:

OPERADOR	DESCRIPCION
<b>&amp;&amp;</b>	AND lógico, chequea si ambos operandos son verdaderos (op1 && op2)
<b>//</b>	OR lógico, chequea si uno de los operandos es verdadero (op1    op2)
<b>!</b>	NOT lógico, niega al operando (!op1)

# Ejemplo

```
public class TestCondicional {  
    static boolean prueba1(int val) {  
        System.out.println("prueba1(" + val + ")");  
        System.out.println("resultado: " + (val < 1));  
        return val < 1;  
    }  
    static boolean prueba2(int val) {  
        System.out.println("preuba2(" + val + ")");  
        System.out.println("resultado: " + (val < 2));  
        return val < 2;  
    }  
    static boolean prueba3(int val) {  
        System.out.println("prueba3(" + val + ")");  
        System.out.println("resultado: " + (val < 3));  
        return val < 3;  
    }  
    public static void main(String[] args) {  
        // Primer prueba Condicional  
        if(prueba1(0) && prueba2(2) && prueba3(2))  
            System.out.println("Primera Expresion es Verdadera");  
        else  
            System.out.println("Primera Expresion es Falsa");  
    }  
}
```

```
// Segunda Prueba Condicional  
if(prueba1(4) || prueba2(1))  
    System.out.println("Segunda Expresion es Verdadera");  
else  
    System.out.println("Segunda Expresion es Falsa");  
  
// Tercer Prueba Negacion  
if(65 != 65)  
    System.out.println("Tercer Expresion es Falsa");  
else  
    System.out.println("Tercer Expresion es Verdadera");  
}
```



The screenshot shows a Java IDE window with the 'Console' tab selected. The output of the program is as follows:

```
<terminated> TestCondicional [Java Application] C:\Program Files\Java\jdk1.7.0_01\bin\javaw.exe (10/11/2011 15:21:36)  
prueba1(0)  
resultado: true  
preuba2(2)  
resultado: false  
Primera Expresion es Falsa  
prueba1(4)  
resultado: false  
preuba2(1)  
resultado: true  
Segunda Expresion es Verdadera  
Tercer Expresion es Verdadera
```

# Operadores lógicos

➤ Tenemos los siguientes operadores lógicos:

&: AND lógico a nivel de bit (op1 & op2).

op1	op2	resultado
0	0	0
0	1	0
1	0	0
1	1	1



# Operadores lógicos

➤ |: OR lógico a nivel de bit (op1 | op2)

op1	op2	resultado
0	0	0
0	1	1
1	0	1
1	1	1

# Operadores lógicos

➤ $\wedge$ : XOR lógico a nivel de bit ( $op1 \wedge op2$ )

op1	op2	resultado
0	0	0
0	1	1
1	0	1
1	1	0

# Operadores de asignación

➤ Tenemos los siguientes operadores de asignación:

OPERADOR	DESCRIPCION
=	Guarda el valor del segundo operando en el primero (op1 = op2).
+=	Guarda la suma de los dos operandos en el primero (op1 += op2).
-=	Guarda la resta de los dos operandos en el primero (op1 -= op2).
*=	Guarda la multiplicación de los dos operandos en el primero (op1 *= op2).

# Otros operadores

- Existen otros operadores en Java como:

OPERADOR	DESCRIPCION
<i>?:</i>	Se trata de una abreviatura de la estructura if-then-else (op1?op2:op3).
<i>[]</i>	Utilizado para declarar, crear y acceder a arrays.
<i>.</i>	Utilizado para acceder a atributos y métodos de objetos.
<i>(parametros)</i>	Utilizado para pasar párametros a un método.
<i>(tipo)</i>	Utilizado para realizar castings (conversiones).
<i>new</i>	Utilizado para crear o instanciar nuevos objetos.
<i>instanceof</i>	Chequea si el primer operando es una instancia del segundo operando

# Sentencias de control de flujo

- Sin las sentencias de control de flujo, el código java se ejecutaría linealmente desde la primera línea hasta la última.

- Existen cuatro tipos de sentencias:

Bucles: while, do while, for y for/in.

Bifurcaciones: if-then-else y switch-case.

Gestión de excepciones: try-catch-finally y throw.

De ruptura: break, continue, label: y return.

# Sentencia while

- La sentencia while se utiliza para ejecutar continuamente un bloque de código mientras la condición del while sea true.

```
while(expresión)
{
    sentencias;
}
```

# Sentencia do while

- La sentencia do-while es parecida a la sentencia while pero asegura que como mínimo el bloque de código se ejecute una vez.

**do**

{

sentencias;

}

**while**(expresión);

# Sentencia for

- La sentencia for facilita la ejecución de un bloque de código un número determinado de veces.

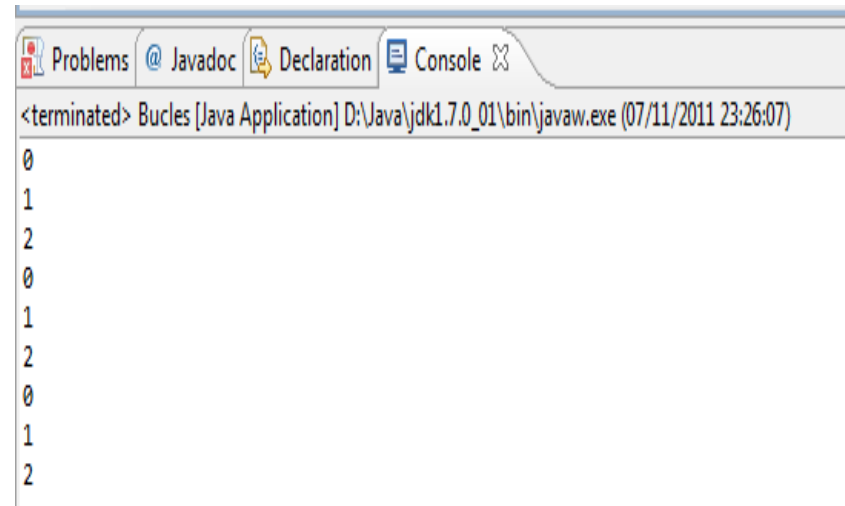
```
for(inicialización;terminación;incremento)
{
    sentencias;
}
```

Nota: Las variables definidas en la sentencia de inicialización son locales al bloque. Por tanto dejan de existir una vez se haya terminado el bucle.



# Ejemplo

```
public class Bucles {  
    public static void main(String[] args)  
    {  
        int cont1 = 0;  
        while(cont1<3)  
        {  
            System.out.println(cont1);  
            cont1++;  
        }  
        int cont2 = 0;  
        do  
        {  
            System.out.println(cont2);  
            cont2++;  
        }  
        while(cont2 < 3);  
        for(int cont3 = 0; cont3 < 3; cont3++)  
        {  
            System.out.println(cont3);  
        }  
    }  
}
```



The screenshot shows a Java IDE window with the 'Console' tab selected. The title bar indicates the application is 'Bucles [Java Application]' running on 'D:\Java\jdk1.7.0\_01\bin\javaw.exe' at '07/11/2011 23:26:07'. The console output displays the results of the three loops: the first while loop prints 0, 1, 2; the second do-while loop prints 0, 1, 2; and the third for loop prints 0, 1, 2.

```
<terminated> Bucles [Java Application] D:\Java\jdk1.7.0_01\bin\javaw.exe (07/11/2011 23:26:07)  
0  
1  
2  
0  
1  
2  
0  
1  
2
```

# Sentencia for/in

- Nos facilita la iteración de los elementos de cualquier tipo de colección: arrays, listas, etc..

```
for(inicialización;colección) //Nota:Se usa ":" en vez de ";".  
{  
    sentencias;  
}
```

Ejemplo:

```
public void listar(int[] param)  
{  
    for(int i: param)  
        system.out.println(i);  
}
```

# Sentencia for/in

- Basicamente, se trata de una simplificación a la hora de codificar.
- Es decir, al final, el compilador convierte el código a una sentencia for convencional:

```
public void listar(int[] param)
{
    for(int=0;i<param.length;i++)
        System.out.println(param[i]);
}
```

# Sentencia if-then-else

- La sentencia if-then-else permite elegir que bloque de código ejecutar entre dos posibilidades o más.

**if**(expresión)

```
{  
    sentencias;  
}
```

**if**(expresión)

```
{  
    sentencias;  
}  
  
else  
  
{  
    sentencias;  
}
```

**if**(expresión)

```
{  
    sentencias;  
}  
  
else if(expresión)  
    sentencias;  
  
else  
  
{  
    sentencias;  
}
```

# Sentencia switch

- La sentencia switch es un caso particular de la sentencia if-then-else if-else, evalúa una expresión de tipo int o que pueda ser convertida a int de forma implícita.

```
switch(intExpresion)
{
    case intExpresion:
        sentencias;
        break;
    default:
        sentencias;
}
```

# Sentencia switch



- En la version 7 de Java, implementa la evaluación de expresiones de tipo String.

```
public class TestSwitch {  
    public void getTypeOfDayWithSwitchStatement(String  
        dayOfWeekArg) {  
        String typeOfDay;  
        switch (dayOfWeekArg) {  
            case "Monday":  
                typeOfDay = "Start of work week";  
                break;  
            case "Tuesday":  
            case "Wednesday":  
            case "Thursday":  
                typeOfDay = "Midweek";  
                break;
```

```
            case "Friday":  
                typeOfDay = "End of work week";  
                break;  
            case "Saturday":  
            case "Sunday":  
                typeOfDay = "Weekend";  
                break;  
            default:  
                throw new IllegalArgumentException  
                    ("Invalid day of the week: "+  
                    dayOfWeekArg);  
        }  
        System.out.println(typeOfDay);  
    }  
    public static void main(String[] args) {  
        TestSwitch test = new TestSwitch();  
        String dayOfWeekArg = "Friday";  
        test.getTypeOfDayWithSwitchStatement  
            (dayOfWeekArg);  
    }  
}
```

# Sentencia de ruptura

- ***break()***: Sirve para detener la ejecución tanto de los bucles como de la sentencia switch, y por tanto saltar a la siguiente línea de código después del bucle o switch.
- ***continue()***: Sirve para detener la ejecución del bloque de código de un bucle y volver a evaluar la condición de este.
- ***return()***: Sirve para finalizar la ejecución de un método.

# Ejercicio

- Identificar si este código compila bien. Si no compila solucionarlo. Si compila decir cuál sería la salida.

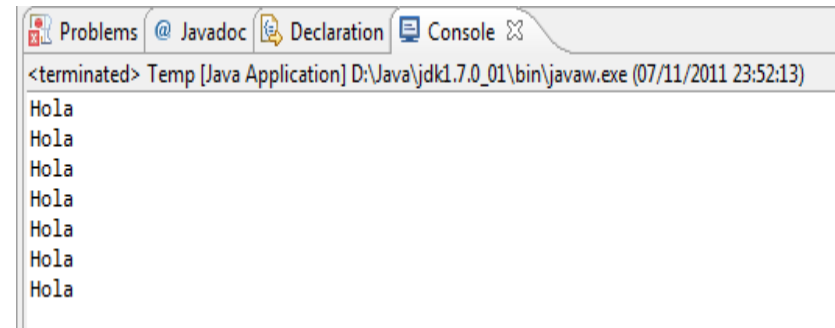
```
public class Temp {  
    public static void main(String[] args)  
    {  
        int x = 1;  
        while(x<10)  
        {  
            if(x>3)  
            {  
                System.out.println("Hola");  
            }  
        }  
    }  
}
```



# Ejercicio (solución)

- El código compila bien. Pero entra en un bucle infinito, habría que modificarlo con la línea roja y saldría la palabra “Hola” siete veces por pantalla.

```
public class Temp {  
    public static void main(String[] args)  
    {  
        int x = 1;  
        while(x<10)  
        {  
            x = x + 1;  
            if(x>3)  
            {  
                System.out.println("Hola");  
            }  
        }  
    }  
}
```



# Ejemplos

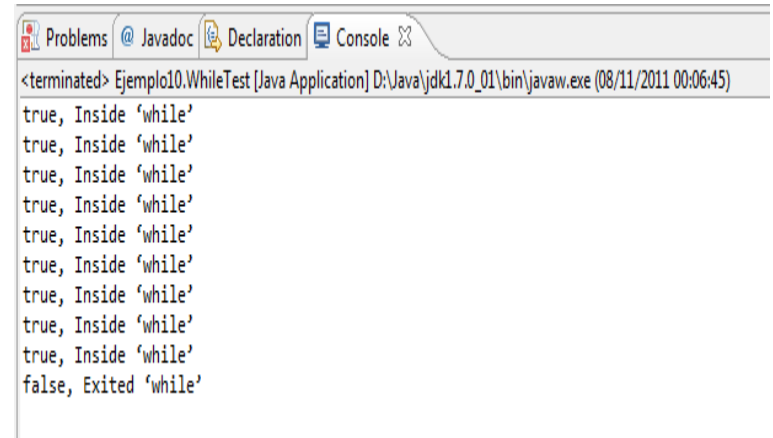
// if-Else

```
public class Ejemplo9 {  
    static int result = 0;  
    static void test(int testval, int target) {  
        if(testval > target)  
            result = +1;  
        else if(testval < target)  
            result = -1;  
        else  
            result = 0;  
    }  
    public static void main(String[] args) {  
        test(10, 5);  
        System.out.println(result);  
        test(5, 10);  
        System.out.println(result);  
        test(5, 5);  
        System.out.println(result);  
    }  
}
```



# Ejemplos

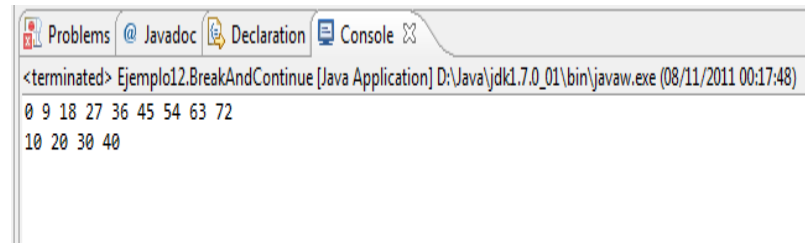
```
public class Ejemplo10 {  
  
    public static class WhileTest {  
        static boolean condition() {  
            boolean result = Math.random() < 0.99;  
            System.out.print(result + ", ");  
            return result;  
        }  
        public static void main(String[] args) {  
            while(condition())  
                System.out.println("Inside 'while'");  
            System.out.println("Exited 'while'");  
        }  
    }  
}
```



```
<terminated> Ejemplo10.WhileTest [Java Application] D:\Java\jdk1.7.0_01\bin\javaw.exe (08/11/2011 00:06:45)  
true, Inside 'while'  
true, Inside 'while'  
true, Inside 'while'  
true, Inside 'while'  
true, Inside 'while'  
true, Inside 'while'  
true, Inside 'while'  
true, Inside 'while'  
true, Inside 'while'  
true, Inside 'while'  
false, Exited 'while'
```

# Ejemplos

```
public class BreakAndContinue {
    public static void main(String[] args) {
        for(int i = 0; i < 100; i++) {
            if(i == 74) break; // Sale de la iteracion
            if(i % 9 != 0) continue; // Continua con la proxima iteracion
            System.out.print(i + " ");
        }
        System.out.println();
        int i = 0;
        // Un "infinito loop":
        while(true) {
            i++;
            int j = i * 27;
            if(j == 1269) break; // Sale de la iteracion
            if(i % 10 != 0) continue; // Inicio de la iteracion
            System.out.print(i + " ");
        }
    }
}
```



# ARRAYS

# Arrays

- Un array es una estructura de datos que permite albergar varios elementos del mismo tipo.
- La longitud de un array se establece durante su creación.
- Una vez establecida la longitud de un array, ya no se puede modificar.
- Un elemento de un array, es el valor de una de sus posiciones, y se identifica mediante un índice.

# Arrays

- Un array en Java, es un tipo de clase especial que hereda implícitamente de la clase `java.lang.Object`.
- La declaración de un array se realiza mediante el tipo de dato que va a albergar y los corchetes `[]`.

- Declaración de un array:

*modificador\_acceso\_tipo[] nombre[=valor\_inicial];*

- Ejemplo:

**private** int[] numeros;

**private** String[] cadenas;

# Arrays

- Los arrays pueden albergar tanto tipos de datos primitivos como complejos.
- Un array sin inicializar, por defecto vale null.
- La creación de un array se realiza mediante la keyword: new, como cualquier otra clase.
- Creación de un array:

```
modificador_acceso_tipo[] nombre = new tipo[longitud];
```

- Ejemplo:

```
private int[] numeros = new int[5];
```

```
private String[] cadenas = new String(4);
```



# Arrays

- Una vez creado un array, todas sus posiciones son inicializadas al valor por defecto del tipo de variable que albergue.
- Es decir, 0 o 0.0 si trabaja con un número, *false* si se trata de boolean y *null* si se trata de tipo de dato complejo.
- Existe una forma de crear un array inicializando todas sus posiciones a un valor determinado, igualándolo a un listado de elementos separados por comas ente {}.
- El tamaño de un array igual al número de elementos del listado.

# Arrays

- Ejemplos:

```
private int[] numeros = {1,2,3,4,5};
```

```
private String[] cadenas = {'hola','adios'};
```

```
private Integer[] ints = {new Integer(12), new Integer(98)};
```

- Para el acceso a un elemento de un array se utiliza el nombre del array seguido de unos [] con la posición a la que queremos acceder.

- La primera posición de un array es la 0.

- Ejemplo:

```
numeros[2] = 3; int a = numero[0]; if(numeros[4] == 5) .....
```

# Arrays

## ➤ En memoria.....

```
public class Colecciones
```

```
{
```

```
  public static void main(String[] args)
```

```
{
```

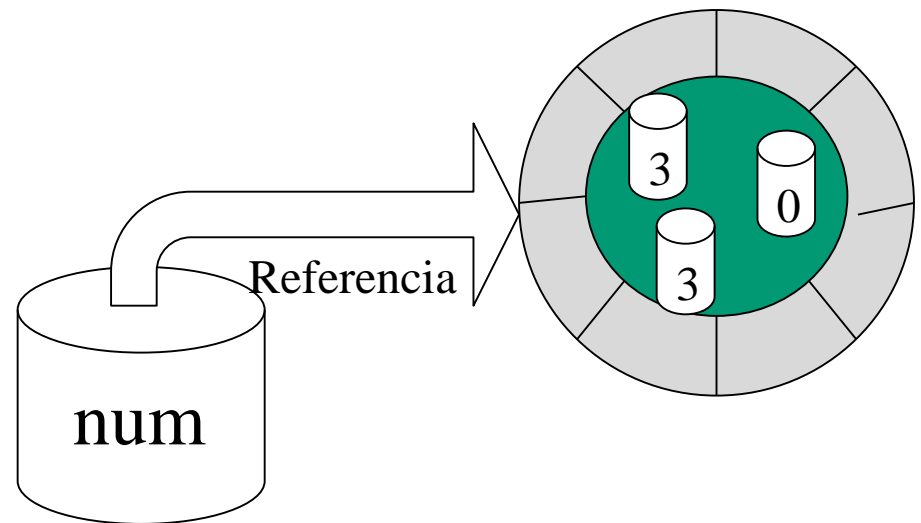
```
    int[] num = new int[3];
```

```
    num[0] = 3;
```

```
    num[1] = 8;
```

```
}
```

```
}
```



# Arrays

➤ En memoria.....

```
public class Colecciones
```

```
{
```

```
  public static void main(String[] args)
```

```
  {
```

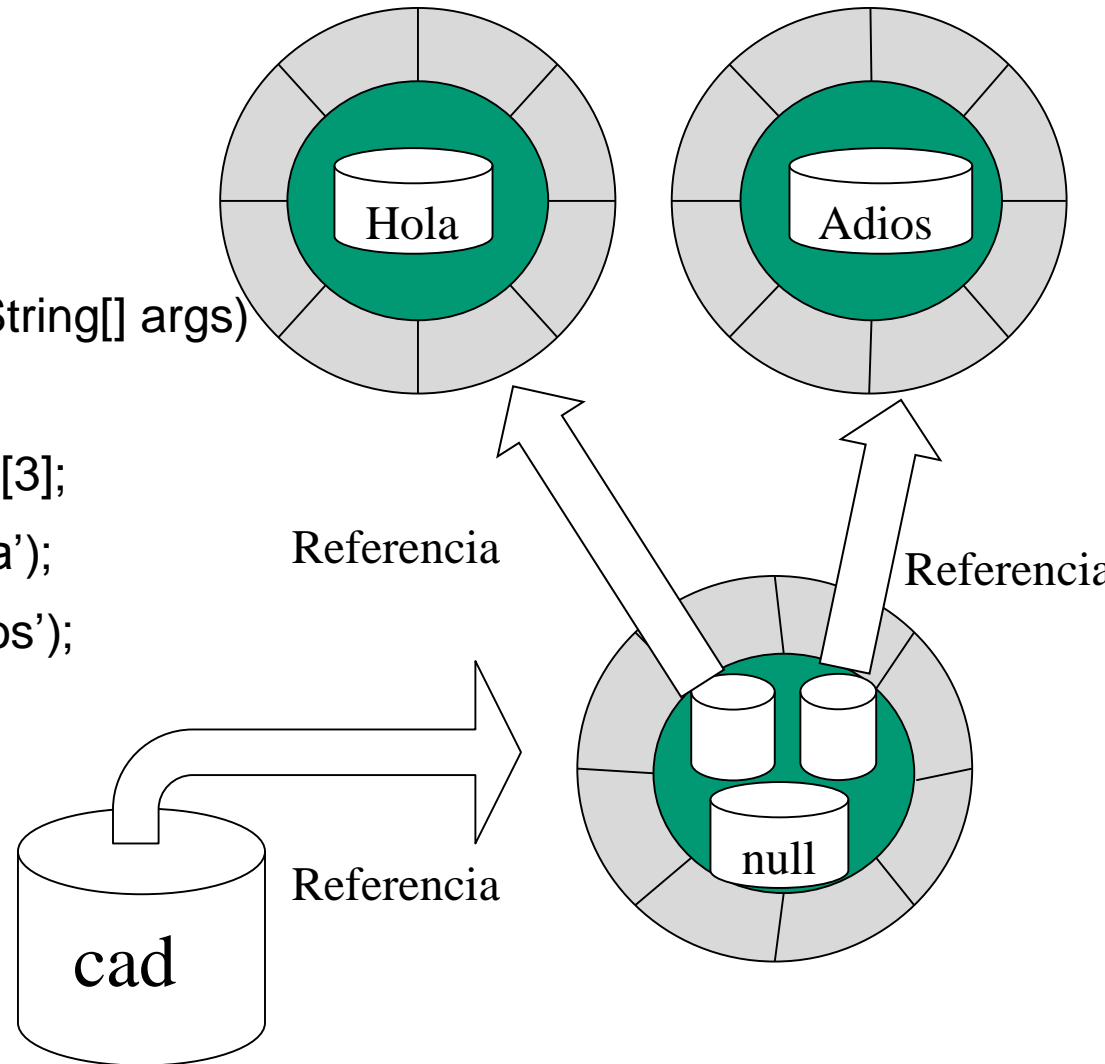
```
    String[] cad = new String[3];
```

```
    cad[0] = new String('Hola');
```

```
    cad[1] = new String('Adios');
```

```
  }
```

```
}
```



# Arrays

- Para conocer la longitud de un array, podremos acceder a su atributo público: `length`.
- El índice de un array es de tipo `int`.
- Al no ser dinámico, no podemos:
  - Eliminar posiciones.
  - Insertar posiciones.
- El borrado será algo lógico, como igualar las posiciones a `null`, a `-1`, etc ... dependerá del desarrollador.

# Arrays

- Es imposible acceder a una posición fuera del array, llegado el caso se lanzará una excepción:

`ArrayIndexOutOfBoundsException`

# Ejemplo

```
public class Colecciones {  
    public static void main(String[] args) {  
        //Creacion e inicializacion  
        String[] saludos = new String[4];
```

//Inserción

```
saludos[0] = new String("Hola");  
saludos[1] = new String("Adios");  
saludos[2] = new String("Hello");  
saludos[3] = new String("GoodBye");
```

//Extracción

```
String extraccion = saludos[2];  
System.out.println(extraccion);
```

//Borrado

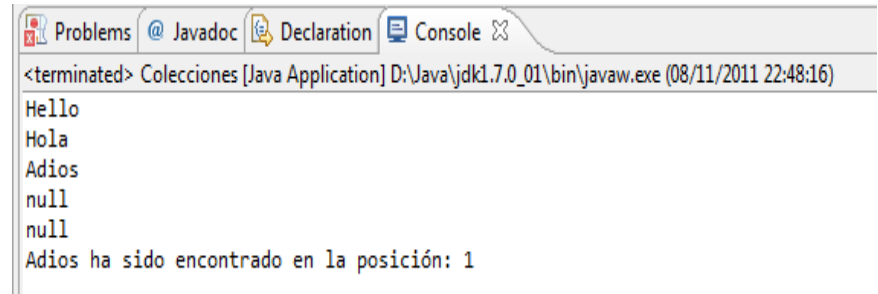
```
saludos[2] = null;  
saludos[3] = null;
```

//Recorrido

```
for(int i=0;i<saludos.length;i++)  
    System.out.println(saludos[i]);
```

//Busqueda

```
for(int i=0;i<saludos.length;i++)  
{  
    if(saludos[i] != null && saludos[i].equals("Adios"))  
    {  
        System.out.println("Adios ha sido encontrado  
                            en la posición: " + i);  
        break;  
    }  
}  
}
```



```
Problems  @ Javadoc  Declaration  Console  X  
<terminated> Colecciones [Java Application] D:\Java\jdk1.7.0_01\bin\javaw.exe (08/11/2011 22:48:16)  
Hello  
Hola  
Adios  
null  
null  
Adios ha sido encontrado en la posición: 1
```

# Sentencia for/in

- Nos facilita la iteración por los elementos de cualquier tipo de colección: arrays, listas, etc..

```
for(inicialización;colección) //Nota:Se usa ":" en vez de ";".  
{  
    sentencias;  
}
```

Ejemplo:

```
public void listar(int[] param)  
{  
    for(int i: param)  
        system.out.println(i);  
}
```



# Sentencia for/in

- Basicamente, se trata de una simplificación a la hora de codificar.
- Es decir, al final, el compilador convierte el código a una sentencia for convencional:

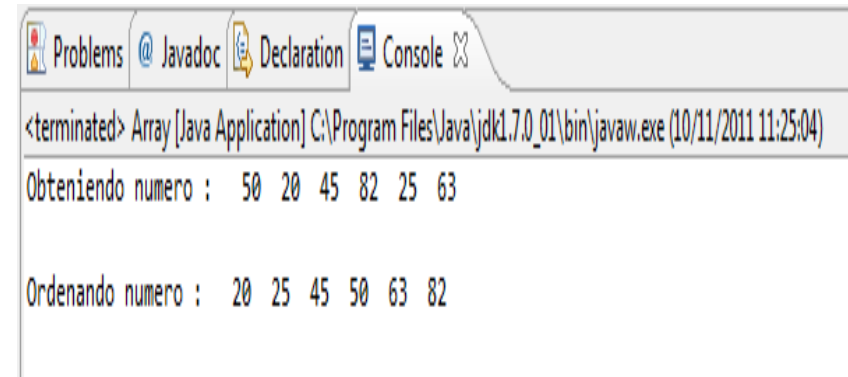
```
public void listar(int[] param)
{
    for(int=0;i<param.length;i++)
        System.out.println(param[i]);
}
```

# Ejemplo

```
import java.util.Arrays;

public class Array {

    public static void main(String[] args){
        int num[] = {50,20,45,82,25,63};
        int l = num.length;
        int i;
        System.out.print("Obteniendo numero : ");
        for (i = 0; i < l; i++) {
            System.out.print(" " + num[i]);
        }
        System.out.println("\n");
        System.out.print("Ordenando numero : ");
        Arrays.sort(num);
        for(i = 0; i < l; i++){
            System.out.print(" " + num[i]);
        }
    }
}
```



The screenshot shows a Java IDE console window with the following tabs: Problems, Javadoc, Declaration, and Console. The console output is as follows:

```
<terminated> Array [Java Application] C:\Program Files\Java\jdk1.7.0_01\bin\javaw.exe (10/11/2011 11:25:04)
Obteniendo numero : 50 20 45 82 25 63
Ordenando numero : 20 25 45 50 63 82
```

# Arrays multidimensionales

- Un array multidimensional es un array de arrays
- Si fuese de tres dimensiones entonces tendríamos un array de arrays de arrays.

- Creación de un array bidimensional:

`modificador_acceso tipo[][] nombre = new tipo [long][long];`

- Ejemplo:

`private int[][] numeros = new int[4][2];`

# Arrays multidimensionales

- Pero podemos tener arrays bidimensionales no cuadradas, es decir que la segunda dimensión tenga longitud diferente dependiendo de la primera dimensión.

- Creacion de un array bidimensional:

modificador\_acceso tipo[][] nombre = new tipo[long][][];

- Ejemplo:

```
private int[][] numeros = new int[4][];
```

```
numeros[0] = new int[2];
```

```
numeros[1] = new int[10];
```

```
numeros[3] = new int[1];
```

# Arrays multidimensionales

- Al igual que ocurriera en los arrays de una dimensión, también se pueden inicializar en la creación con un listado de valores.
- Ejemplos:

```
private int[][] numeros = {{1,2,3},{123}};
```

```
private String[][] dias = {{'Lunes','Martes'}, {'Miercoles'}};
```

# Arrays

➤ En memoria.....

```
public class Colecciones
```

```
{
```

```
  public static void main(String[] args)
```

```
  {
```

```
    int[] num = new int[2][2];
```

```
    num[0][0] = 0;
```

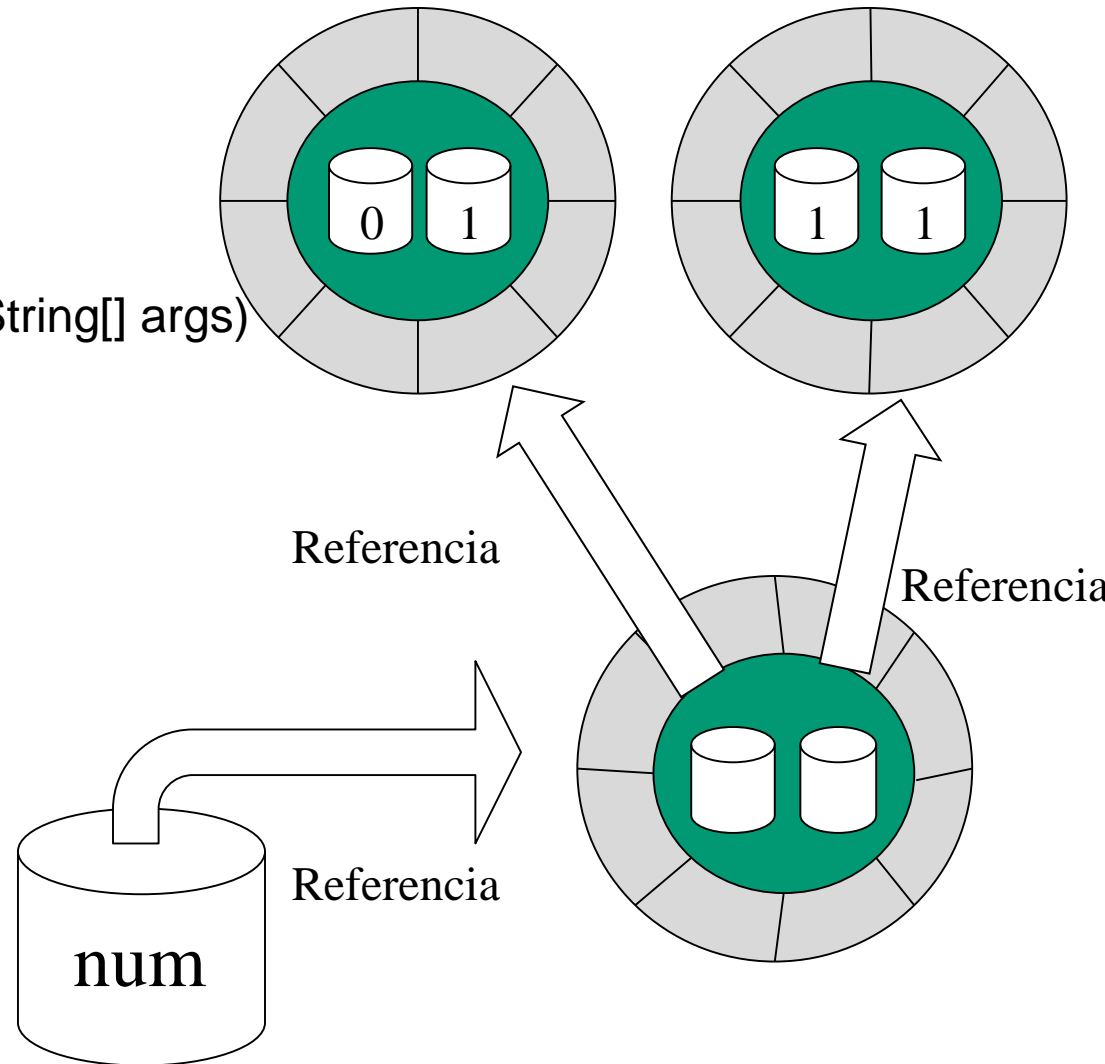
```
    num[0][1] = 1;
```

```
    num[1][0] = 1;
```

```
    num[1][1] = 1;
```

```
  }
```

```
}
```



# El método main usa arrays

- El método main recibe un array de Strings que contiene los argumentos enviados en el arranque.

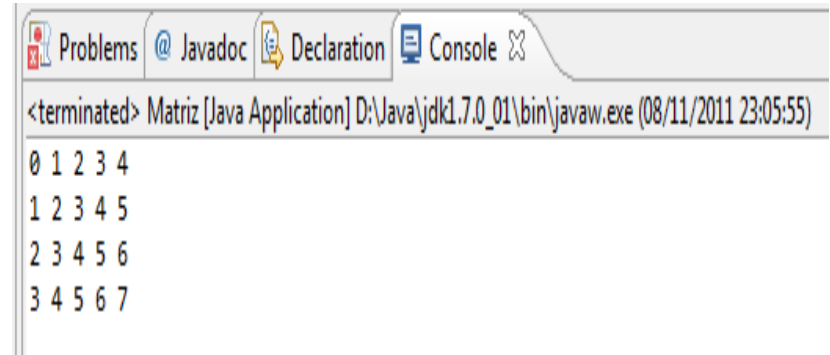
```
public static void main(String[] args)
```

```
public class Test
{
    public static void main(String[] args)
    {
        for(int i=0;i<args.length;i++)
            System.out.println("Argumento " + i + ":" + args[i]);
    }
}
```

```
C:\trabajo>Java Test arg1 arg2
Argumento 0: arg1
Argumento 1: arg2
```

# Ejemplo

```
public class Matriz {  
    public static void main(String[] args) {  
        int[][] matriz = new int[4][];  
        //Rellenar la matriz  
        for(int i=0; i<matriz.length;i++) {  
            matriz[i] = new int[5];  
            for(int j=0;j<matriz[i].length;j++)  
                matriz[i][j] = i + j;  
        }  
        //Mostrar la matriz  
        for(int i=0;i<matriz.length;i++) {  
            for(int j=0; j<matriz[i].length;j++)  
                System.out.print(matriz[i][j] + " ");  
            System.out.println();  
        }  
    }  
}
```



```
<terminated> Matriz [Java Application] D:\Java\jdk1.7.0_01\bin\javaw.exe (08/11/2011 23:05:55)  
0 1 2 3 4  
1 2 3 4 5  
2 3 4 5 6  
3 4 5 6 7
```



# Conclusiones

- En la presente sesión, se detalló la tecnología Java con sus 3 plataformas.
- Además, se identificó la sintaxis básica del lenguaje de programación Java, revisando las estructuras de control, repetitivas, entre otros.