



# Tema 4

---

## Sintaxis de Java



# Comentarios en Java

---

- Existen tres formas distintas de escribir los comentarios:

- // Comentarios en una línea

- /\* Comentarios de una línea o más líneas \*/

- /\*\* Comentarios de documentación, utilizado por la herramienta javadoc \*/

# Puntos y comas, bloques y espacios en blanco

---

- Una sentencia es una línea simple de código terminada en un punto y coma:
  - `System.out.println("Hola");`
- Un bloque es un conjunto de sentencias agrupadas entre llaves ({}):

```
while(true){  
    x = y + 1;  
    x = x + 1;  
}
```
- Los bloques pueden estar anidados “puede haber uno dentro de otro”, y Java nos permite que haya espacios entre los elementos de código fuente.

```
while(true)  
{  
    x = y + 1;  
    if(x<0)  
    {  
        x = x + 1;  
    }  
}
```



# Identificadores

---

- Los identificadores son los nombres unívocos que se le dan a las clases, métodos y variables.
- A la hora de crear identificadores hay que tener presente las siguientes reglas:
  - Deben empezar por una letra, subrayado (\_) o dólar (\$).
  - Después del primer carácter se pueden usar números.
  - Distinguen las mayúsculas y minúsculas.
  - Nunca pueden coincidir con una keyword o palabra reservada.

# Palabras clave de Java

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



# Ejemplos de identificadores

---

## ■ Estos identificadores serían válidos:

- ☐ Identificador
- ☐ nombreUsuario
- ☐ nombre\_usuario
- ☐ \_sys\_var2
- ☐ \$cambio
- ☐ if2



# Variables

---

- Una variable es un contenedor de datos identificado mediante un nombre (identificador).
- Por convenio, en Java, los nombres de las variables empiezan con una letra minúscula
- Si una variable está compuesta de más de una palabra, como **nombreDato** las palabras se ponen juntas y cada palabra después de la primera empieza con una letra mayúscula
- Dicho identificador se utilizará para referenciar el dato que contiene.
- Todas las variables deben llevar asociado un tipo que describe el tipo de dato que se guarda.
- Por tanto, una variable tiene:
  - Un tipo.
  - Un identificador.
  - 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.





# Tipos de datos

---

- 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 lógico

---

- Es el tipo: boolean
- Sus posibles valores son:
  - true
  - false
- Su valor por defecto es:
  - false
- Ejemplos:
  - `boolean switch1 = true;`
  - `boolean switch2;`



# Tipo de dato carácter

---

- Es el tipo: char
- Representa un carácter UNICODE
- Su tamaño es: 16 bits (2 bytes)
- Sus posibles valores son:
  - Un carácter entre comillas simples: 'a'
  - Un carácter especial con \ delante: '\n', '\t', ...
  - Un código UNICODE: '\uxxxx' (donde xxxx es un valor en hexadecimal).



# Tipo de dato carácter

---

- Su valor por defecto es:
  - `'\u0000'` -> null
- Ejemplos:
  - `char letra1 = 'a';`
  - `char letra2 = '\n';`
  - `char letra3 = '\u0041';`
  - `char letra4;`
- Existe un tipo complejo para las cadenas de caracteres: la clase `String`.



# Tipos de datos enteros

---

- Son los tipos: byte, short, int y long
- Sus tamaños son:
  - byte: 8 bits (1 byte), por tanto: -128 a 127.
  - short: 16 bits (2 bytes), por tanto: -32768 a 32767
  - int: 32 bits (4 bytes), por tanto: -2147483648 a 2147483647
  - long: 64 bits (8 bytes), por tanto: *-enorme a enorme*
- Sus posibles valores son:
  - Un valor decimal entero: 45 (por defecto int) o 45L (long).
  - Un valor octal: 055;//55 octal=45 decimal (en octal un número siempre empieza por cero, seguido de dígitos octales)
  - Un valor hexadecimal: 0x2D;// 2D hexadecimal=45 decimal (en hexadecimal un número siempre empieza por 0x seguido de dígitos hexadecimales)



# Tipos de datos reales

---

- Son los tipos: float y double
- Sus tamaños son:
  - float: 32 bits (4 bytes)
  - double: 64 bits (8 bytes)
- Sus posibles valores son:
  - Un valor decimal entero: 2 (por defecto int).
  - Un valor decimal real: 0.17 ó 6.02E23 (por defecto double).
  - Un valor decimal real: 0.17F ó 0.17D (redundante).



# Tipos de datos reales

---

- Su valor por defecto es:
  - 0.0 (cero)
- Ejemplos:
  - **float** unFloat = 0.17F;
  - **double** unDouble;
  - **double** otroDouble = -12.01E30;



# Tipos de dato enumeración

---

- El tipo es: enum
- Se trata de un tipo de dato complejo algo especial
- Implementa una clase que tiene un atributo que puede tomar varios valores y solo esos.
- Ejemplo:
  - **enum** Semaforo { VERDE, AMBAR, ROJO }
  - **enum** TurnoClase{ MAÑANA, TARDE }



# Ejemplo

```
import java.util.*;

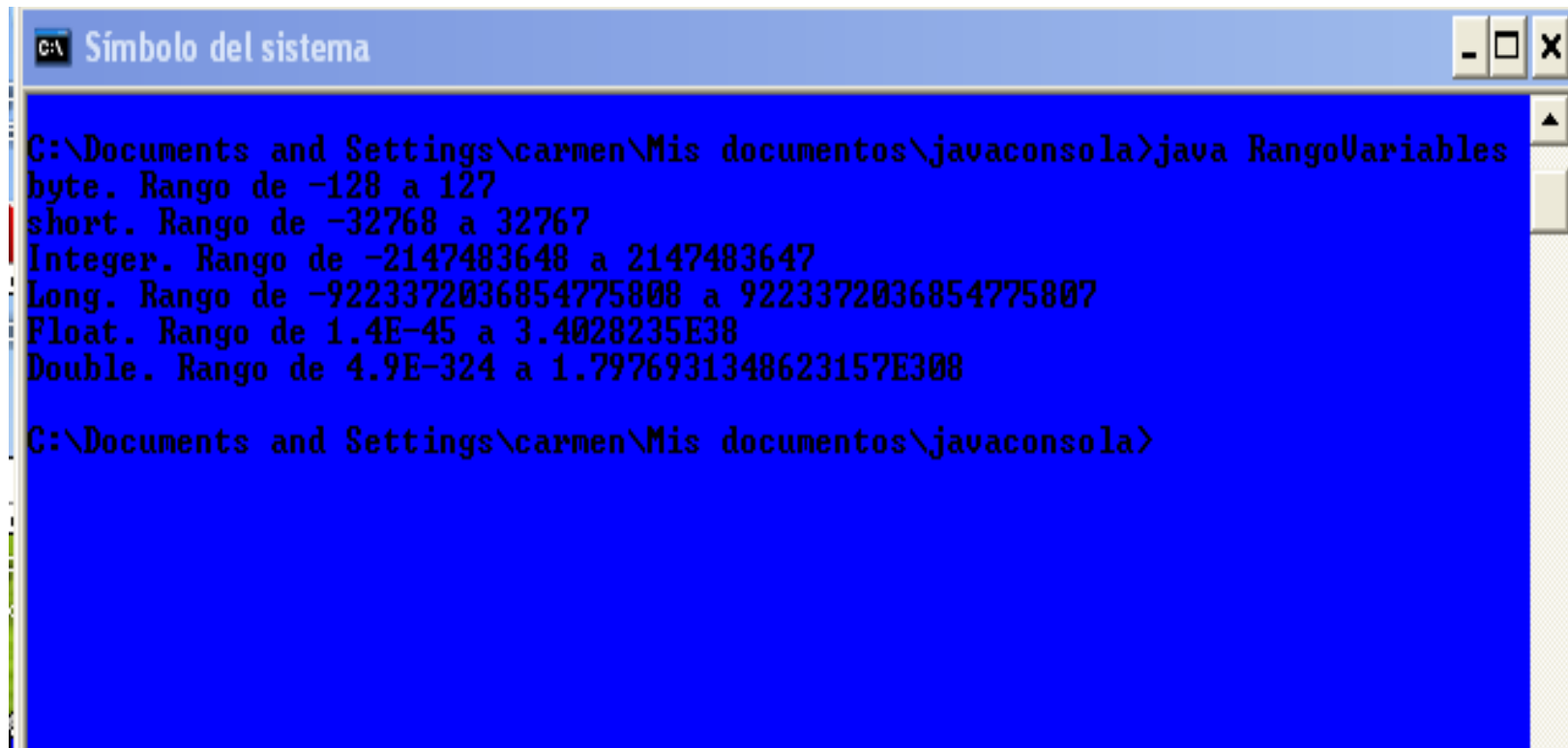
/*Programa que imprime los rangos de los tipos primitivos de Java.*/

public class RangoVariables
{
    public static void main (String args [])
    {
        // Enteros
        System.out.println("byte. Rango de " + Byte.MIN_VALUE + " a " + Byte.MAX_VALUE);
        System.out.println("short. Rango de " + Short.MIN_VALUE + " a " + Short.MAX_VALUE);
        System.out.println("Integer. Rango de " + Integer.MIN_VALUE + " a " + Integer.MAX_VALUE);
        System.out.println("Long. Rango de " + Long.MIN_VALUE + " a " + Long.MAX_VALUE);
        //Reales
        System.out.println("Float. Rango de " + Float.MIN_VALUE + " a " + Float.MAX_VALUE);
        System.out.println("Double. Rango de " + Double.MIN_VALUE + " a " + Double.MAX_VALUE);
        //Caracteres
        // System.out.println("Char. Rango de " + Char.MIN_VALUE + " a " + Char.MAX_VALUE);
    }
}
```

# Ejemplo

```
public class TiposEnumerados1 {  
    //definimos un tipo enumerado  
    //los tipos enumerados deben definirse siempre fuera  
    //del main y, más en general, fuera de cualquier método  
    public enum Semana {LUNES, MARTES, MIERCOLES, JUEVES,VIERNES, SABADO, DOMINGO};  
  
    public static void main(String[] args) {  
        //definimos una variable que pertenece al tipo enumeradoSemana  
        //y le damos el valor que representa el día martes  
        Semana hoy = Semana.MARTES;  
        //si el día cayese en el fin de semana no hay que trabajar  
        //obsérvese como gracias a la numeración del programa es fácil de entender  
        if (hoy == Semana.DOMINGO || hoy == Semana.SABADO)  
            System.out.println("Hoy toca descansar");  
        else  
            System.out.println("Hoy toca trabajar");  
        }  
    }  
}
```

## Ejecución del ejemplo



```
C:\Documents and Settings\carmen\Mis documentos\javaconsola>java RangoVariables
byte. Rango de -128 a 127
short. Rango de -32768 a 32767
Integer. Rango de -2147483648 a 2147483647
Long. Rango de -9223372036854775808 a 9223372036854775807
Float. Rango de 1.4E-45 a 3.4028235E38
Double. Rango de 4.9E-324 a 1.7976931348623157E308

C:\Documents and Settings\carmen\Mis documentos\javaconsola>
```

# Conversión de tipos (casting)

Java es un lenguaje fuertemente tipado, no se pueden declarar variables de un tipo y asignarles directamente valores de otro tipo. Se debe hacer una conversión:

- **Implicita** ( en tiempo de compilación)

```
int a; byte b=7;
```

```
a=b; (cuando el tipo al que se asigna es mayor)
```

- **Con casting** (en tiempo de ejecución)

```
int a=7; byte b;
```

```
b=(byte) a;
```



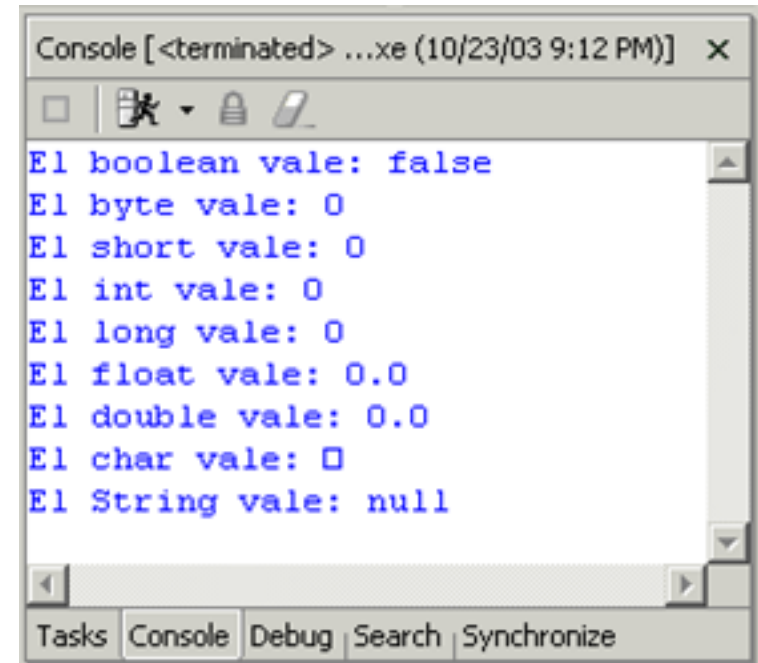
# Variables primitivas vs. complejas

---

- Una variable de tipo primitivo contiene el dato directamente
  - `byte a = 10;`
- Una variable de tipo complejo contiene una referencia (puntero) a la zona de memoria donde está el objeto:
  - `String s = new String("Hola");`

# 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 vale: " + unInt);  
        System.out.println("El long vale: " + unLong);  
        System.out.println("El float vale: " + unFloat);  
        System.out.println("El double vale: " + unDouble);  
        System.out.println("El char vale: " + unChar);  
        System.out.println("El String vale: " + unString);  
    }  
}
```



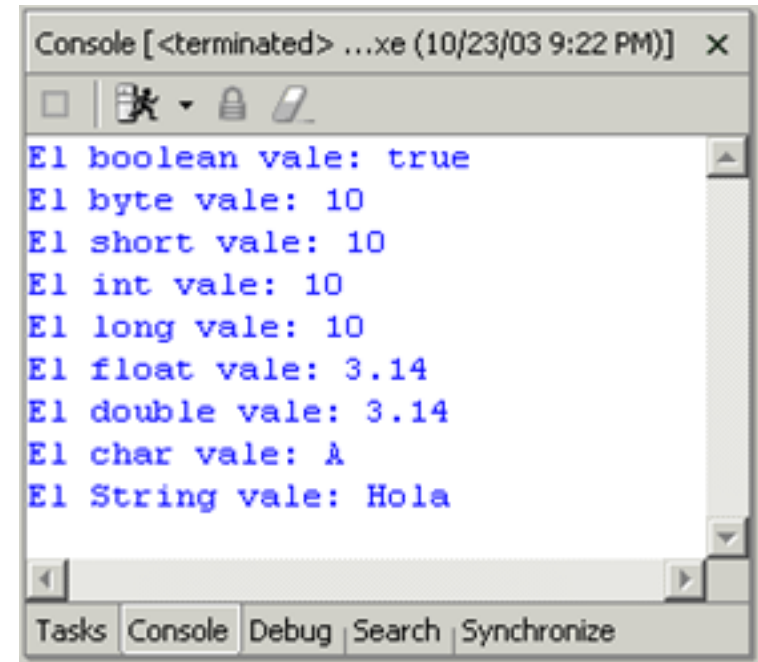
The screenshot shows a console window titled "Console [<terminated> ...xe (10/23/03 9:12 PM)]". The output of the program is displayed in blue text:

```
El boolean vale: false  
El byte vale: 0  
El short vale: 0  
El int vale: 0  
El long vale: 0  
El float vale: 0.0  
El double vale: 0.0  
El char vale:    
El String vale: null
```

At the bottom of the console window, there are tabs for "Tasks", "Console", "Debug", "Search", and "Synchronize".

# Ejemplo

```
public class VariablesTest2
{
    public static void main(String[] args)
    {
        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");
        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 vale: " + unInt);
        System.out.println("El long vale: " + unLong);
        System.out.println("El float vale: " + unFloat);
        System.out.println("El double vale: " + unDouble);
        System.out.println("El char vale: " + unChar);
        System.out.println("El String vale: " + unString);
    }
}
```



The screenshot shows a console window titled "Console [<terminated> ...xe (10/23/03 9:22 PM)]". The output of the program is displayed in blue text:

```
El boolean vale: true
El byte vale: 10
El short vale: 10
El int vale: 10
El long vale: 10
El float vale: 3.14
El double vale: 3.14
El char vale: A
El String vale: Hola
```

At the bottom of the console window, there are tabs for "Tasks", "Console", "Debug", "Search", and "Synchronize".

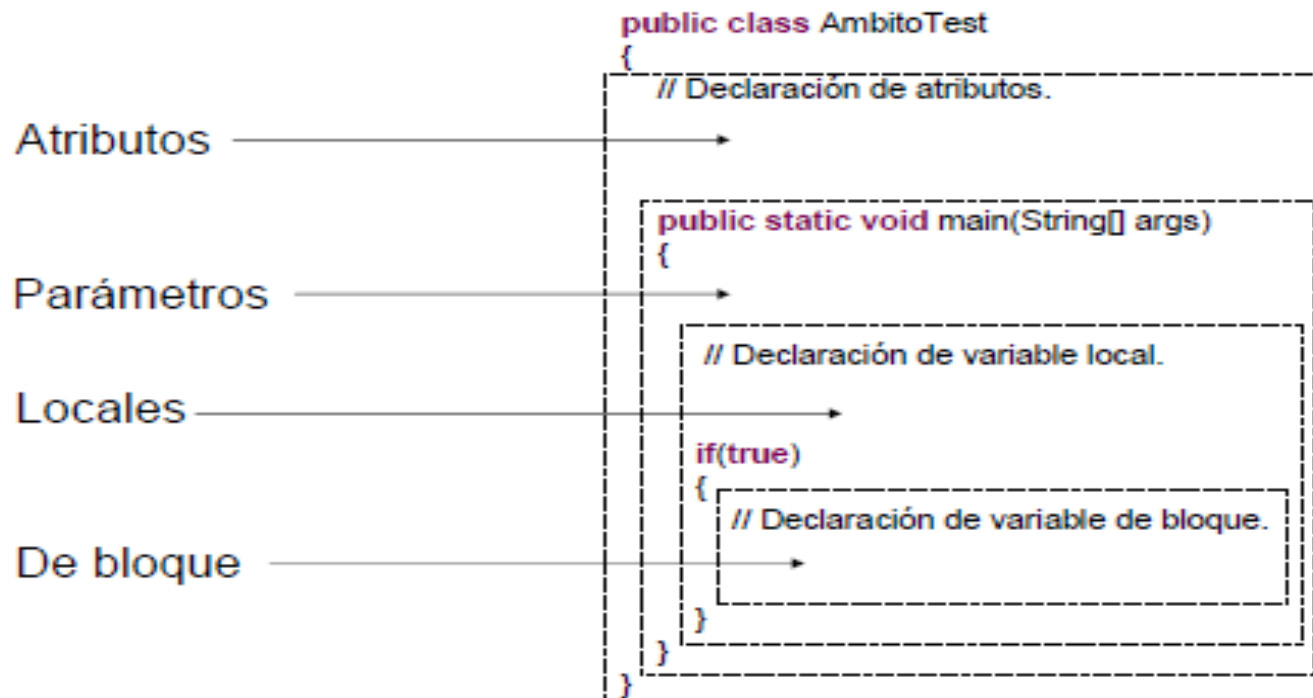
# Ámbito de las variables

---

- 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 ámbito.
- Ámbitos
  - Atributos (o variables miembro)
  - Parámetros de método
  - Variables locales: siempre hay que inicializarlas
  - Variables de bloque: siempre hay que inicializarlas

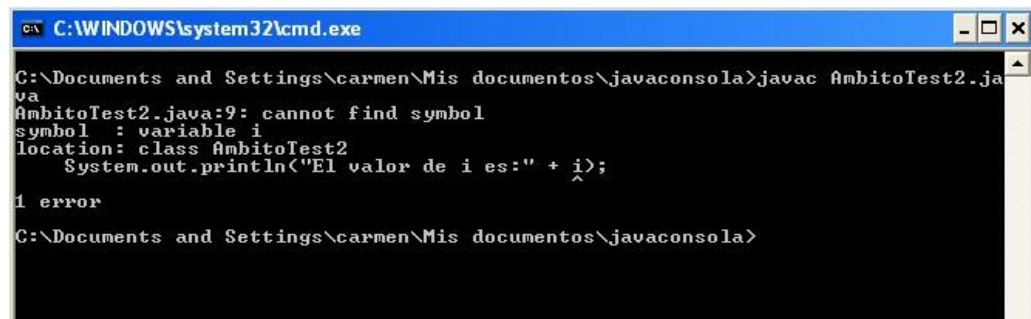


# Ámbito de las variables



# Ejemplo

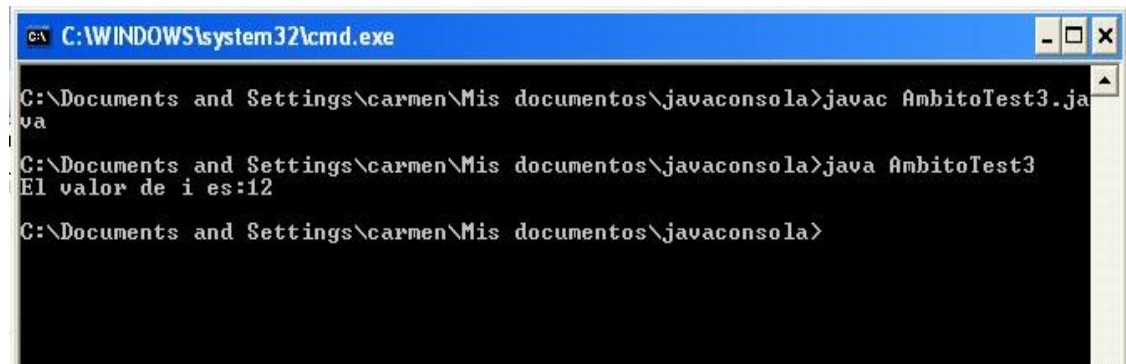
```
public class AmbitoTest2
{
    public static void main(String[] args)
    {
        if (true)
        {
            int i=12;
        }
        System.out.println("El valor de i es:" + i);
    }
}
```



A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows the command prompt at "C:\Documents and Settings\carmen\Mis documentos\javaconsola>". The user has entered "javac AmbitoTest2.java". The output shows a compilation error: "AmbitoTest2.java:9: cannot find symbol", "symbol : variable i", "location: class AmbitoTest2", and "System.out.println(\"El valor de i es:\" + i);". The error message "1 error" is displayed at the bottom of the output. The cursor is positioned at the end of the command line.

# Ejemplo

```
public class AmbitoTest3
{
    static int i=5;
    public static void main(String[] args)
    {
        int i=12;
        System.out.println("El valor de i es:" + i);
    }
}
```



```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\carmen\Mis documentos\javaconsola>javac AmbitoTest3.java

C:\Documents and Settings\carmen\Mis documentos\javaconsola>java AmbitoTest3
El valor de i es:12

C:\Documents and Settings\carmen\Mis documentos\javaconsola>
```



# Operadores

Los operadores realizan algunas funciones en uno o dos operandos.

- **operadores unarios** requieren un solo operando.

Ej: `s++;`

- **operadores binarios** requieren dos operandos

Ej: `operador =` es un operador binario que asigna el valor del operando derecho al operando izquierdo.

# Operadores aritméticos.

Resta o signo menos -

Suma +

Multiplicación \*

División /

resto división entera(módulo) %

Decremento --

Incremento ++

**Nota** + concatena para String

```
int a=10, b=3, c;
```

```
float x=2.0, y;
```

```
y=x+a; /* resultado 12.0 tipo float */
```

```
c=a/b; /* resultado 3 de tipo int */
```

```
c=a%b; /* resultado 1 de tipo int */
```

```
y=a/b; /* resultado es 3 tipo float */
```

```
y=x/a; /* a se transforma a float resultado  
0.2 float */
```

## Producen resultado numérico

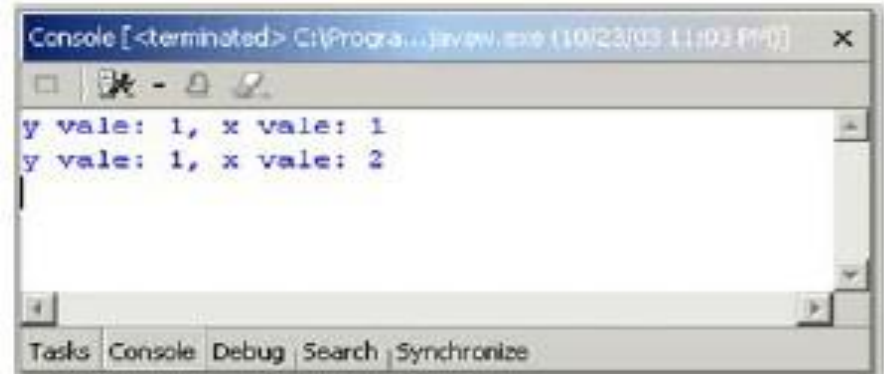
```
X=10;
```

```
Y=++x; /* y=11 x = 11 */ primero incremento luego asignación
```

```
Y=x++; /* y=10 x = 11 */ primero asignación luego incremento
```

# 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);
    }
}
```



# Operador de Asignación

Asignación.(simples y abreviados)

- $=$              $x=3$
- $+=$             $x+=3$  es lo mismo que  $x=x+3$
- $-=$             $x-=3$  es lo mismo que  $x=x-3$
- $*=$             $x*=3$  es lo mismo que  $x=x*3$
- $/=$             $x/=3$  es lo mismo que  $x=x/3$
- $\%=$             $x\%=3$  es lo mismo que  $x=x\%3$



# Operadores relacionales

## ■ Relacionales

- ☐ < menor que
- ☐ <= menor o igual que
- ☐ > mayor que
- ☐ >= mayor o igual que
- ☐ == igual
- ☐ != distinto

## ■ Resultados: Booleano 1 – 0 (verdadero o falso)



# Operadores Lógicos

- $\&\&$  y o AND
- $\parallel$  OR
- $!$  NOT  
(monario)

A	B	!A	A&&B	A  B
V	V	0	1	1
V	F	0	0	1
F	V	1	0	1
F	F	1	0	0



# Operadores para Tratamiento de Bits

- Se utilizan para realizar operaciones a nivel de bit, y los operandos deben ser de tipo entero o de tipo char
- No se pueden utilizar sobre los tipos: float, double, long double y void.
- Se utilizan con mucha frecuencia en:
  - ☐ controladores de dispositivos
  - ☐ programas de modem
  - ☐ rutinas de archivos de disco
  - ☐ rutinas de impresoras
  - ☐ comunicaciones series, para calcular la paridad

# Operadores para Tratamiento de Bits

## 1) Operadores para desplazamiento de bit

- << Desplazamiento a la izqda

$A \ll b$  Desplaza los bits del número entero A, b posiciones a la izquierda.

*Cada posición desplazada a la izquierda equivale a multiplicar por 2 el número A.*

- >> Desplazamiento a la derecha

$A \gg b$  Desplaza los bits del número entero A, b posiciones a la derecha.

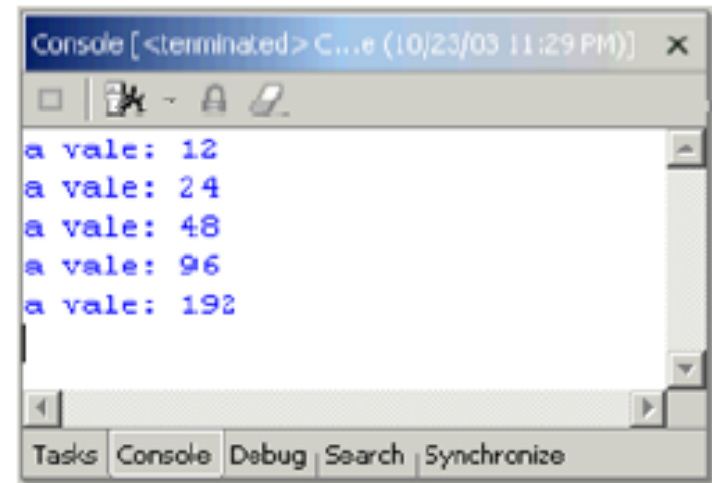
*Cada posición desplazada a la derecha equivale a dividir por 2 el número A.*

- >>> Desplazamiento a la derecha sin signo

- <<< Desplazamiento a la izqda sin signo

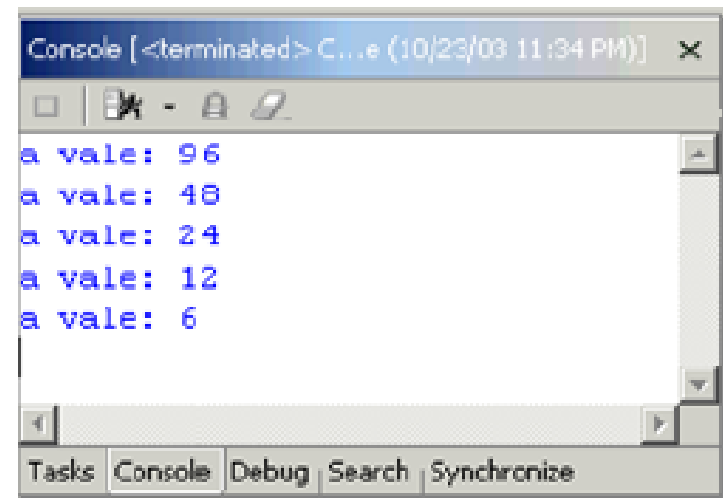
# Ejemplo

```
public class Multiplicador
{
    public static void main(String[] args)
    {
        int a = 6;
        a = a << 1;
        System.out.println("a vale: " + a);
        a = a << 1;
        System.out.println("a vale: " + a);
        a = a << 1;
        System.out.println("a vale: " + a);
        a = a << 1;
        System.out.println("a vale: " + a);
    }
}
```



# Ejemplo

```
public class Dividor
{
    public static void main(String[] args)
    {
        int a = 192;
        a = a >> 1;
        System.out.println("a vale: " + a);
        a = a >> 1;
        System.out.println("a vale: " + a);
        a = a >> 1;
        System.out.println("a vale: " + a);
        a = a >> 1;
        System.out.println("a vale: " + a);
        a = a >> 1;
        System.out.println("a vale: " + a);
    }
}
```





# Operadores para Tratamiento de Bits

## 2) operadores lógicos

- $\&$   $\Rightarrow$  y lógica bit a bit (El resultado es 1 si ambos bits lo son)
- $|$   $\Rightarrow$  o lógica bit a bit (El resultado es 0 si ambos bits lo son)
- $\wedge$   $\Rightarrow$  o exclusiva bit a bit (El resultado es 1 si ambos bits son diferentes)
- $\sim$   $\Rightarrow$  Complemento a 1 (Cambia un bit a su opuesto)



# Otros Operadores

- **Operador de molde o cast:** convierte el tipo de dato de un operando o una expresión, sin que varíe en su declaración  
(tipo) operando ó expresión
- **Operador Condicional :** Se utiliza para realizar una operación alternativa mediante una condición
  - expresión 1 ? expresión 2 : expresión 3
- **new :** Se utiliza para crear objetos nuevos
- **instanceof:** Se utiliza para comprobar si un objeto pertenece a una clase concreta en cuyo caso devuelve true o false sino pertenece.

```
public class Relacional {
    public static void main(String arg[]) {
        double op1,op2;
        op1=1.34;
        op2=1.35;
        System.out.println("op1="+op1+" op2="+op2);
        System.out.println("op1>op2 = "+(op1>op2));
        System.out.println("op1<op2 = "+(op1<op2));
        System.out.println("op1==op2 = "+(op1==op2));
        System.out.println("op1!=op2 = "+(op1!=op2));
        char op3,op4;
        op3='a'; op4='b';
        System.out.println("'a'>'b' = "+(op3>op4));
    }
}
```

**Salida por pantalla:**

```
op1=1.34 op2=1.35
op1>op2 = false
op1<op2 = true
op1==op2 = false
op1!=op2 = true
'a'>'b' = false
```