

```
int a = 7 & 4;
System.out.println(a);
int b = 3 & 4;
System.out.println(b);
```

 Funciona No funciona

4

Ø

```
int a = ~4;
System.out.println(a);
```

 Funciona No funciona

-5

```
int a = (~4 * 5)&1;
System.out.println(a);
```

 Funciona No funciona

1

- 10. Dentro de una clase *joven* tenemos las variables enteras *edad*, *nivel\_de\_estudios* e *ingresos*.

Necesitamos almacenar en la variable *booleana* *jasp* el valor:

- Verdadero. Si la *edad* es menor o igual a 28 y el *nivel\_de\_estudios* es mayor que tres, o bien, la *edad* es menor de 30 y los *ingresos* superan los 28.000 (euros).
- Falso. En caso contrario.

- Escribe el código necesario (2 líneas).

- 11. Realiza un programa con una variable entera *t* la cual contiene un tiempo en segundos y queremos conocer este tiempo pero expresado en horas, minutos y segundos.

- 12. (Ejercicio de dificultad alta) Realiza un programa que dado un importe en euros nos indique el mínimo número de billetes y la cantidad sobrante que se pueden utilizar para obtener dicha cantidad.

Por ejemplo:

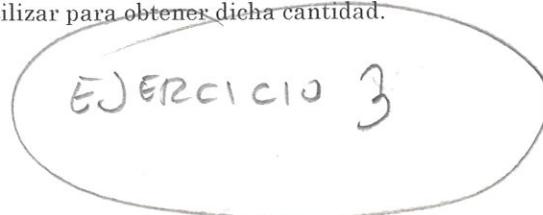
232 euros:

1 billete de 200.

1 billete de 20.

1 billete de 10

Sobran 2 euros.



#### Solución al ejercicio resuelto número 3:

Todas las afirmaciones son falsas.



## EJERCICIOS PROPUESTOS

- 1. Modifica el siguiente programa para hacer que compile y funcione:

```
class suma
{
    static int n1=50;
    public static void main(String [] args)
    {
        int n2=30, suma=0, n3;
        suma=n1+n2;
        System.out.println("LA SUMA ES: " + suma);
        suma=suma+n3;
        System.out.println(suma);
    }
}
```

- 2. ¿Por qué no compila el siguiente programa? Modificalo para hacer que funcione.

```
class suma
{
    public static void main(String [] args)
    {
        int n1=50,n2=30,
        boolean suma=0;
        suma=n1+n2;
        System.out.println("LA SUMA ES: " + suma);
    }
}
```

- 3. El siguiente programa tiene 3 fallos, averigua cuáles son y modifica el programa para que funcione.

```
class cuadrado
{
    public static void main(String [] args)
    {
        int numero=2,
        cuad=numero * número;
        System.out.println("EL CUADRADO DE "+NUMERO+" ES: " + cuad);
    }
}
```



## RESUMEN DEL CAPÍTULO



En este tema se introduce al alumno en los lenguajes de programación, y más concretamente, al lenguaje Java. Este tema es una primera toma de contacto del alumno con la programación y se hace desde una posición global viendo cómo funciona un programa muy sencillo (el famoso Hola Mundo) y desde los aspectos básicos como son las variables, los tipos de datos, los comentarios, los operadores, etc.

Es posible que el alumno en estos primeros temas vea conceptos que luego se estudien con más profundidad en apartados siguientes. El alumno en este tema deberá de entender la estructura, cómo funciona Java y poner énfasis en el estudio de los aspectos básicos del lenguaje, los distintos tipos de datos y en la utilidad y uso de los operadores y expresiones.

También es importante para el capítulo que el alumno sepa instalar y utilizar un IDE. La instalación, compilación y ejecución de programas es una cuestión básica que debe de manejar el alumno para el resto del libro.



## EJERCICIOS RESUELTOS



- 1. Realiza un método para la clase *Test* que genere letras de forma aleatoria. Como ejercicio complementario investiga el funcionamiento y uso de la función *Math.random()*.

**Solución:**

```
class Test {  
    public static char getLetras() {  
        return (char)(Math.random()*26 + 'a');  
    }  
    public static void main(String[] args) {  
        System.out.println(getLetras());  
        System.out.println(getLetras());  
        System.out.println(getLetras());  
        System.out.println(getLetras());  
    }  
}
```

```
byte dato1 = 3; short dato2 = 5;
```

```
dato2 = dato1;
```

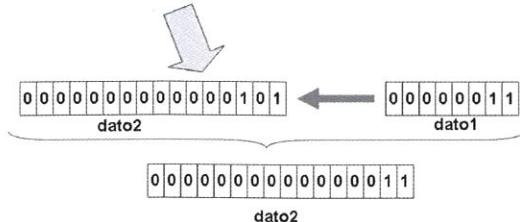


Figura 1.7. Ejemplo de conversión implícita

- **Conversiones explícitas.** En este caso es el programador el que fuerza la conversión mediante una operación llamada *cast* con el formato:

(tipo) expresión



#### Recuerda

Como puede ser comprensible no se pueden realizar conversiones entre enteros y *booleanos* o reales y *booleanos*.

Un ejemplo de conversión explícita sería el siguiente:

```
int idato=5;
byte bdato;
bdato = (byte)idato;
System.out.println(bdato); // sacará 5 por pantalla
```



#### Consejo

Intenta evitar las conversiones de tipos en la medida de lo posible... En algunas conversiones explícitas como ya supondrás pueden perder información en algunos casos.

En el siguiente ejemplo se puede observar la utilización de operadores de bits:

```
int num=5;
num = num << 1; // num = 10, equivale a num = num * 2
num = num >> 1; // num = 5, equivale a num = num / 2
```

### 1.7.6 OPERADORES DE ASIGNACIÓN

**Tabla 1.8.** Operadores de asignación

Operador	Uso	Operación
=	A = B	Asignación. Operador ya visto.
*=	A *= B	Multiplicación y asignación. La operación A*=B equivale a A=A*B.
/=	A /= B	División y asignación. La operación A/=B equivale a A=A/B.
%=	A %= B	Módulo y asignación. La operación A%=B equivale a A=A%B.
+=	A += B	Suma y asignación. La operación A+=B equivale a A=A+B.
-=	A -= B	Resta y asignación. La operación A-=B equivale a A=A-B.

En el siguiente ejemplo se puede observar la utilización de operadores de asignación:

```
int num=5;
num += 5; // num = 10, equivale a num = num + 5
```

### 1.7.7 PRECEDENCIA DE OPERADORES



#### Consejo

Utiliza paréntesis y de esa forma puedes dejar los programas más legibles y controlar las operaciones sin tener que depender de la precedencia.

### 1.7.2 OPERADORES RELACIONALES

Con los operadores relacionales se puede evaluar la igualdad y la magnitud. En la siguiente tabla A y B no son los operadores, sino que son los operandos como se puede ver:

**Tabla 1.4.** Operadores relacionales

Operador	Uso	Operación
<	A < B	A menor que B
>	A > B	A mayor que B
<=	A <= B	A menor o igual que B
>=	A >= B	A mayor o igual que B
!=	A != B	A distinto que B
= =	A == B	A igual que B

En el siguiente ejemplo se puede observar la utilización de operadores relacionales:

```
int m=2, n=5;
boolean res;
res =m > n;//res=false
res =m < n;//res=true
res =m >= n;//res=false
res =m <= n;//res=true
res =m == n;//res=false
res =m != n;//res=true
```

### 1.7.3 OPERADORES LÓGICOS

Con los operadores lógicos se pueden realizar operaciones lógicas. En la siguiente tabla A y B no son los operadores, sino que son los operandos como se puede ver:

**Tabla 1.5.** Operadores lógicos

Operador	Uso	Operación
&& o &	A&& B o A&B	A AND B. El resultado será true si ambos operandos son true y false en caso contrario.
o	A    B o A   B	A OR B. El resultado será false si ambos operandos son false y true en caso contrario.
!	!A	Not A. Si el operando es true el resultado es false y si el operando es false el resultado es true.
^	A ^ B	A XOR B. El resultado será true si un operando es true y el otro false, y false en caso contrario.

```
class suma
{
    static int n1=50; // variable miembro de la clase
    public static void main(String [] args)
    {
        int n2=30, suma=0; // variables locales
        suma=n1+n2;
        System.out.println("LA SUMA ES: " + suma);
    }
}
```

Como puede verse en el ejemplo anterior, las variables se declaran dentro de un bloque (por bloque se entiende el contenido entre las llaves {}) y son accesibles solo dentro de ese bloque.

Las variables declaradas en el bloque de la clase como *n1* se consideran miembros de la clase, mientras que las variables *n2* y *suma* pertenecen al método *main* y solo pueden ser utilizados en el mismo. Las variables declaradas en el bloque de código de un método son variables que se crean cuando el bloque se declara, y se destruyen cuando finaliza la ejecución de dicho bloque.



#### **Inicialización de variables**

Las variables miembros de una clase se inicializan por defecto (las numéricas con 0 los caracteres con '\0' y las referencias a objetos y cadenas con *null*) mientras que las variables locales no se inicializan por defecto.



#### **Importante**

Una variable local no puede ser declarada como *static*.

#### **1.6.1 VISIBILIDAD Y VIDA DE LAS VARIABLES**

Visibilidad, *scope* o *ámbito* de una variable son sinónimos. Visibilidad es la parte del código de una aplicación donde la variable es accesible y puede ser utilizada.

**ACTIVIDADES**

- Se propone al alumno que investigue y recopile información sobre el juego de caracteres Unicode y ASCII con especial detenimiento en este último.

**1.4.1 ¿CÓMO SE UTILIZAN LOS TIPOS DE DATOS?**

A continuación, se muestran ejemplos de utilización de tipos de datos en la declaración de variables.

**Tabla 1.2.** Utilización de tipos de datos

Tipo de dato	Código
byte	byte a;
short	short b, c=3;
int	int d = -30; int e = 0xC125;
long	long b=434123 ; long b=5L ; /* la L en este caso indica Long*/
char	char car1='c'; char car2=99; /*car1 y car2 son lo mismo porque el 99 en decimal es la 'c' */
float	float pi=3.1416; float pi=3.1416F; /* la F en este caso indica Float*/ float medio=1/2F; /*0.5*/
double	double millón=1e6; /* 1x106 */ double medio1/2D; /*0.5 la D en este caso indica Double*/

## 1.5 CONSTANTES Y LITERALES

**1.5.1 LAS CONSTANTES****Cuestión de estilo**

Las constantes se declaran en mayúscula mientras que las variables se hacen en minúscula (esto se realiza como norma de estilo).

menos interesante; es NetBeans de la extinta SUN® ahora Oracle®. NetBeans es una aplicación de código abierto y muchos desarrolladores Java la utilizan. Como consejo, se recomienda Geany para pequeños proyectos y programas como NetBeans o Eclipse para proyectos más serios.

**Recuerda**

NetBeans y Eclipse son entornos de desarrollo libres.

**¿Es necesario un IDE para compilar y ejecutar Java?**

La respuesta es No. No es necesario compilar desde un IDE nuestro programa. En principio, si la variable PATH está correctamente configurada bastaría con ejecutar desde línea de comando y desde el mismo directorio donde se encuentra el fichero holamundo.java el siguiente comando:

```
$javac holamundo.java
```

Si el programa está correctamente escrito y el compilador no muestra ninguna salida de error aparecerá un fichero holamundo.class que será el bytecode o código que podrá ser ejecutado en cualquier máquina virtual Java.

**ACTIVIDADES**

- Investiga qué es el bytecode y qué es y cómo funciona una máquina virtual de Java.

Una vez tenemos el fichero .class hay que ejecutar el programa. Esto se realiza con el siguiente comando:

```
$java holamundo
```

O si se está en un entorno Windows®:

```
C:\> java holamundo
```

Y saldrá en la pantalla la cadena que queremos “Hola Mundo”.

**ACTIVIDADES**

- Instala Geany y el JDK en tu máquina. Una vez instaladas estas dos cosas configura las variables PATH y CLASSPATH en la máquina.
- Prueba a compilar dentro y fuera del IDE el programa holamundo y comprueba que funciona ejecutándolo.

### Mostrar texto por pantalla.

Parece intuitivo saber que el texto se mostrará por pantalla ejecutando la siguiente línea:

System.out.println ("Hola Mundo");  
↳ clase ↳ método .

Para sacar información por pantalla en Java se utiliza la clase **System** que puede ser llamada desde cualquier punto de un programa, la cual tiene un atributo **out** que a su vez tiene dos métodos muy utilizados: **print()** y **println()**. La diferencia entre estos dos últimos métodos es que en el segundo se añade un retorno de línea al texto introducido. Como se puede ver la orden termina en ; (todas las ordenes en Java terminan en ; salvo los cierres de llaves a los cuales no hace falta ponérselo pues se sobreentiende que se finaliza la orden).

## 1.3 ENTORNOS INTEGRADOS DE DESARROLLO

Un IDE o Entorno Integrado de Desarrollo es una herramienta con el cual poder desarrollar y probar proyectos en un lenguaje determinado.



#### Recuerda

JDK o Java Development Kit es el software necesario para poder desarrollar y ejecutar programas java. También se denomina SDK (*Standard Development Kit*) o incluso J2SE (*Java 2 platform Standard Edition*).

Lo primero que hay que hacer cuando se instala un IDE es configurar como mínimo la ruta del JDK (*Java Development Kit*). Si no se tiene el JDK no se podrá trabajar con Java, luego habrá que instalarlo primero. En Ubuntu Linux basta con ejecutar desde consola el siguiente comando:

```
$ sudo apt-get install sun-java6-jdk
```



#### Importante

Cuando se instale un entorno integrado de desarrollo hay que asegurarse que las opciones que indican las rutas de las bibliotecas, el JDK y demás recursos son correctas. Si no se hace esto el programa nunca podrá ejecutar ni compilar programas.

### 1.1.3 LOS PROGRAMAS EN JAVA

Los programas o aplicaciones en Java se componen de una serie de ficheros .class que son ficheros en bytecode que contienen las clases del programa. Estos ficheros no tienen por qué estar situados en un directorio concreto, sino que pueden estar distribuidos en varios discos o incluso en varias máquinas.

La aplicación se ejecuta desde el método principal o *main()* situada en una clase. A partir de aquí se van creando objetos a partir de las clases y se va ejecutando la aplicación. El *main()* es un método estático (ya se explicará esto más adelante) el cual puede empezar a crear los objetos, incluidos los de su propia clase.

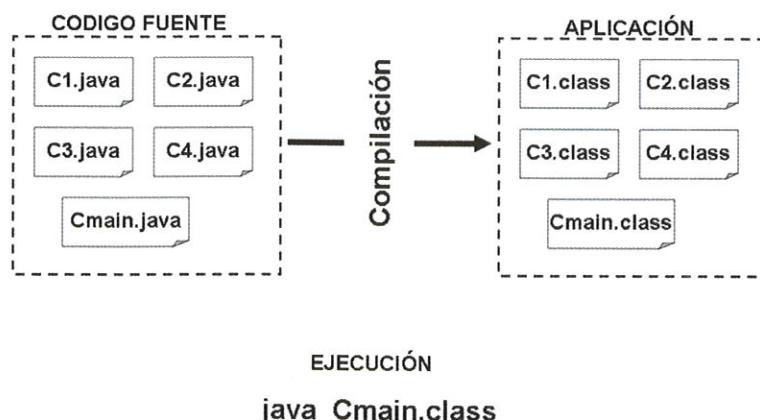


Figura 1.4. Proceso de compilación de un programa

## 1.2 ESTRUCTURA Y BLOQUES FUNDAMENTALES DE UN PROGRAMA

En este apartado se va a ver el programa de inicio por excelencia en cualquier lenguaje de programación y se comentará cada una de sus líneas. El proceso de compilación y ejecución se explica en el siguiente apartado.

```
public class holamundo {
/* programa holamundo*/
public static void main(String[] args) {
    /* lo único que hace este programa es mostrar
       la cadena "Hola Mundo" por pantalla*/
    System.out.println("Hola Mundo");
}
}
```

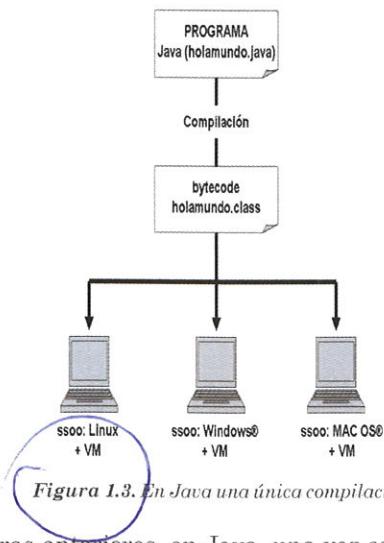


Figura 1.3. En Java una única compilación

Como puede apreciarse en las figuras anteriores, en Java, una vez compilado el programa, se puede ejecutar en cualquier plataforma solamente con tener instalada la máquina virtual (*Virtual Machine – VM*) de Java. Sin embargo en C, C++ u otro lenguaje, deberemos recomilar el programa para el sistema destino con la consiguiente pérdida de flexibilidad.

Por lo tanto, Java es un compilador y a la vez un intérprete. El compilador compila a bytecode y el intérprete se encargará de ejecutar ese código intermedio en la máquina real.



#### Recuerda

Java es multiplataforma y programas en Java pueden ser ejecutados en Windows®, GNU/Linux y Mac OS X entre otros sistemas.

James Gosling trabajaba para Sun Microsystems® y fue el diseñador de Java en 1990. El primer nombre que tuvo Java fue OAK y tuvo como referentes C y C++ (de hecho se parece mucho a ellos en el aspecto, pero la filosofía de funcionamiento es totalmente distinta). SUN® desarrollo este lenguaje en principio con otra orientación, la idea es que fuese utilizado en microelectrónica y sistemas embebidos. Lo que nunca se pensó SUN® es la repercusión y evolución que tendría más tarde este lenguaje.



#### Cuatro razones para aprender Java

1. Por el futuro y presente que tiene.
2. Es un lenguaje sencillo.
3. Es un lenguaje orientado a objetos.
4. Es independiente de la plataforma.

La información de este capítulo muchas veces es un resumen y en ocasiones no trata en profundidad ciertos aspectos. No obstante, el alumno en la sección de bibliografía puede encontrar libros y páginas aconsejadas en los que puede ampliar o contrastar la información en este libro proporcionada.

## 1.1 PROGRAMA Y LENGUAJES DE PROGRAMACIÓN

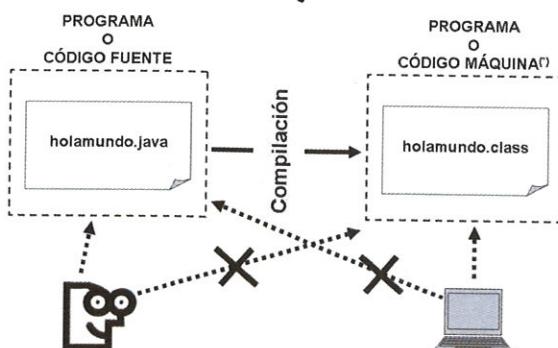


### Definición de programa

Un programa es una serie de órdenes o instrucciones ordenadas con una finalidad concreta que realizan una función determinada.

Todo el mundo estamos familiarizados con la ejecución de programas (editores de textos, navegadores, juegos, reproductores de música o películas, etc.). Por regla general, cuando queremos ejecutar un programa se lo indicamos al sistema haciendo doble click sobre él e incluso algunos usuarios más avanzados ejecutan comandos desde un intérprete de comandos o consola. Si una vez has tenido la curiosidad de abrir un programa con un bloc de notas o editor de texto te habrás dado cuenta que aparece algo horrible en el editor, una serie de símbolos ininteligibles (por los humanos). Eso es porque los programas están en binario, que es el lenguaje que entienden las máquinas. Entonces te preguntarás: si al final de este libro seré capaz de escribir programas, ¿podré entender esos códigos? La respuesta es No. En este libro vamos a aprender un lenguaje de programación para escribir programas de manera entendible por los humanos que luego traduciremos al lenguaje máquina entendible por los ordenadores mediante otros programas llamados intérpretes o compiladores.

En la siguiente figura se verá todo esto de modo más gráfico:



(\*) En Java es bytecode. Interpretable por la máquina virtual de Java.

*Figura 1.1. Programas en código fuente y máquina*

Como se puede observar, el código fuente es el que escribe el programador que luego lo compila a código máquina. Compilar equivale a transformar el programa inteligible por el programador al programa inteligible por la máquina. El código fuente o programa fuente está escrito en un lenguaje de programación y el compilador es un programa que se encarga de transformar el código fuente en código máquina.