Capítulo 1

Elementos de un programa informático

Índice

Objetivos del capítulo:

- Conocer qué es un programa, un lenguaje de programación y las diferencias entre lenguajes de programación como Java y C o C++.
- Reconocer el aspecto de un programa básico en Java y sus características principales.
- Instalar y utilizar un IDE.
- Compilar y ejecutar programas sencillos en Java dentro y fuera de un Entorno de desarrollo.
- Conocer y utilizar fundamentos básicos del lenguaje Java como los tipos de datos, constantes, literales, variables, comentarios, operadores y expresiones.
- Identificar las ventajas y limitaciones de Java frente a otros lenguajes de programación.

1.1 Programa y lenguajes de programación

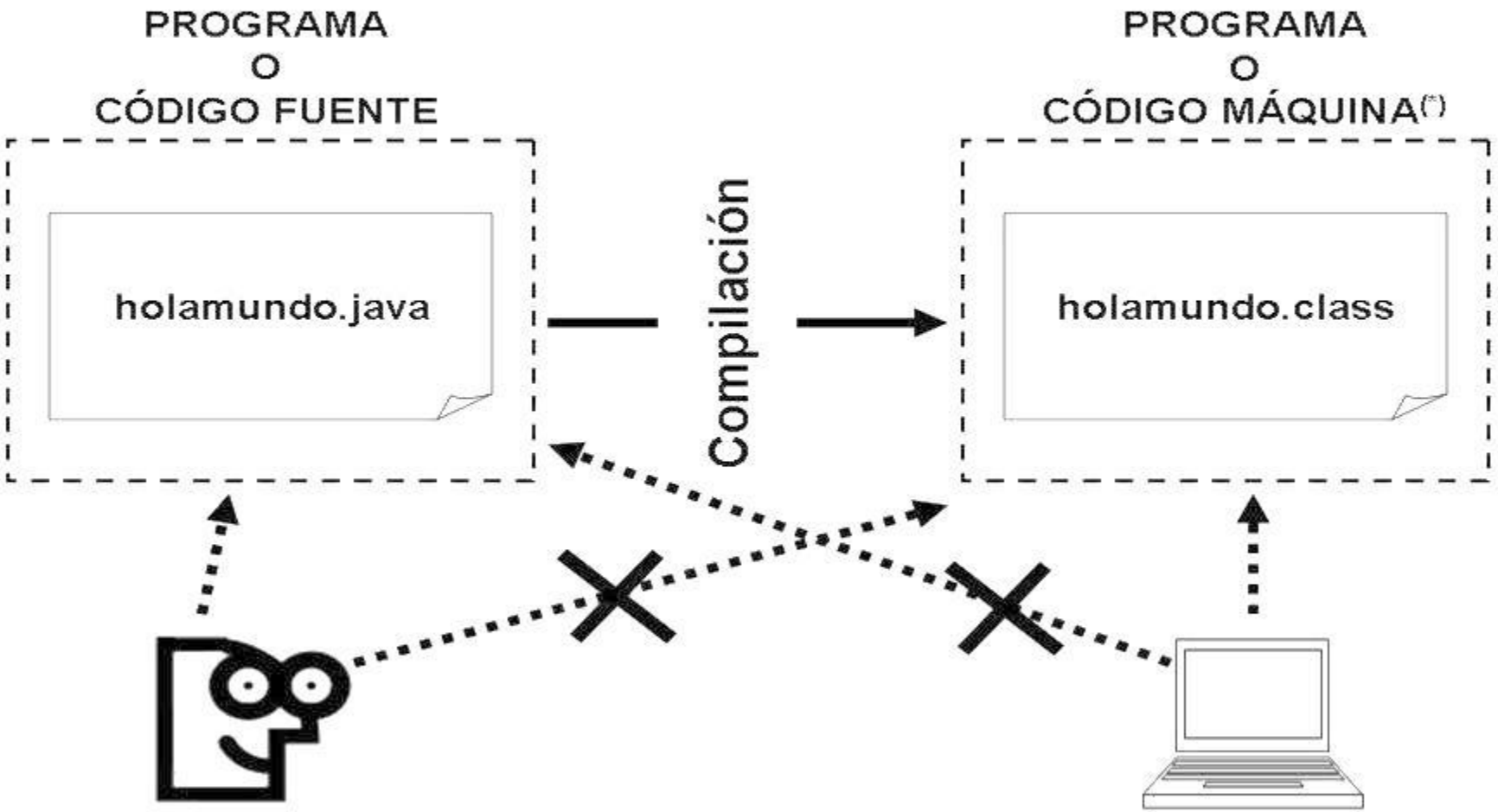
Definición de programa

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

Recuerda

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

Proceso de compilación



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

Compiladores e intérpretes

A diferencia de los compiladores, los intérpretes leen línea a línea el código fuente y lo ejecutan.
Este proceso es muy lento y requiere tener cargado en memoria el intérprete.

4 Razones para aprender Java

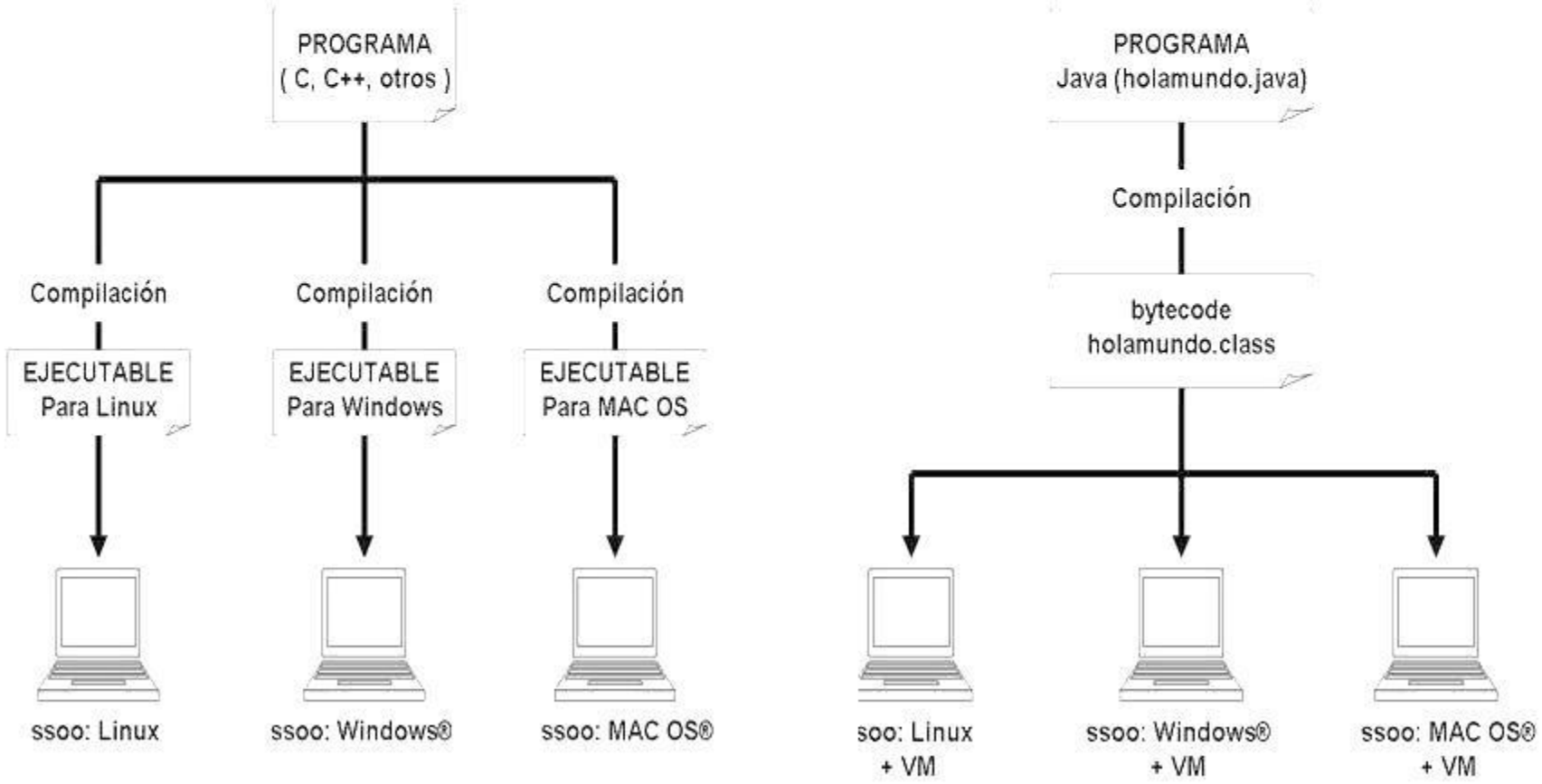
1. Por el futuro y presente que tiene.
2. Es un lenguaje sencillo.
3. Es un lenguaje orientado a objetos.
4. Es independiente de la plataforma.

1.1.1 **EL LENGUAJE JAVA**

- Es uno de los lenguajes más utilizados en la actualidad.
- Es un lenguaje de propósito general.
- Lenguaje de Internet (Applets, Servlets, páginas JSP o JavaScript).
- Lenguaje multiplataforma.
- Java utiliza una máquina virtual en el sistema destino
- Utiliza un código intermedio (*bytecode)* que puede ser ejecutado en cualquier sistema.

1.1.1 EL LENGUAJE JAVA

- Java vs otros lenguajes de programación



JAMES GOSLING

Trabajó para Sun Microsystems® y fue el diseñador de Java en 1990.

El primer nombre de Java fue OAK y tuvo como referentes C y C++.

SUN® desarrolló Java para que fuese utilizado en microelectrónica y sistemas embebidos.

Agradecimientos a Kyle y Kelly Adams por la foto

1.1.2 **EL JDK**

- No contiene ninguna herramienta gráfica
- Contiene aplicaciones de consola y herramientas de compilación, documentación y depuración.
- Incluye el JRE (*Java Runtime Environment*).
- Herramientas de consola (java, javac, javap, jdb, javadoc y appletviewer)

1.2 ESTRUCTURA Y BLOQUES FUNDAMENTALES DE UN PROGRAMA

La clase holamundo

```
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");
    }
}
```

1.3 ENTORNOS INTEGRADOS DE DESARROLLO

Trabajando con un IDE

- Herramienta con el cual poder desarrollar y probar proyectos.
- Primero configurar la ruta del JDK (*Java Development Kit*).
- Configurar la variable PATH en Windows® o las variables JAVA_HOME y JAVA en Linux.
- No es necesario un IDE para compilar y ejecutar Java.

1.4 TIPOS DE DATOS SIMPLES

Tipo de datos	**Información representada**	**Rango**
byte	Datos enteros	-128 ←→ +127
short	Datos enteros	-32768 ←→ +32767
int	Datos enteros	-2147483648 ←→ +2147483647
long	Datos enteros	-9223372036854775808 ←→ +9223372036854775807
char	Datos enteros y caracteres	0 ←→ 65535
float	Datos en coma flotante de 32 bits	Precisión aproximada de 7 dígitos
double	Datos en coma flotante de 64 bits	Precisión aproximada de 16 dígitos
boolean	Valores booleanos	true/false

1.4.1 ¿CÓMO SE UTILIZAN LOS 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; /* $1x10^6$ */ double medio1/2D; /*0.5 la D en este caso indica Double*/
boolean	boolean adivinanza=true;

1.5 CONSTANTES Y LITERALES

1.5.1 **LAS CONSTANTES**

- Formato:

final [static] <tipo> <nombre> = <valor>;

final static double PI=3.141592;

Las constantes se declaran en mayúscula mientras que las variables se hacen en minúscula (por estilo).

Las constantes

Las constantes se utilizan en datos que nunca varían (IVA, PI, etc.). Utilizando constantes y no variables nos aseguramos que su valor no va a poder ser modificado nunca.

También utilizar constantes permite centralizar el valor de un dato en una sola línea de código (si se quiere cambiar el valor del IVA se hará solamente en una línea en vez de si se utilizase el literal 18 en muchas partes del programa).

1.6 **VARIABLES**

Las variables

Una variable es una zona de memoria donde se puede almacenar información del tipo

```
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);
    }
}
```

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.

1.6.1 **VISIBILIDAD Y VIDA DE LAS VARIABLES**

En Java las variables no pueden declararse fuera de una clase.

- Visibilidad o scope es la parte del código de una aplicación donde la variable es accesible y puede ser utilizada.

1.7 OPERADORES Y EXPRESIONES

1.7.1 OPERADORES ARITMÉTICOS

Operador	Uso	Operación
+	A + B	Suma
-	A - B	Resta
*	A * B	Multiplicación
/	A / B	División
%	A % B	Módulo o resto de una división entera

1.7.1 **OPERADORES ARITMÉTICOS**

```
int  n1=2, n2;
n2=n1 * n1;       // n2=4
n2=n2-n1;         // n2=2
n2=n2+n1+15; // n2=19
n2=n2/n1;         // n2=9
n2=n2%n1;         // n2=1
```

1.7.2 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

1.7.2 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

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.

1.7.3 **OPERADORES LÓGICOS**

```
int  m=2, n=5;
boolean res;
res =m > n && m >= n;//res=false
res =!(m < n || m != n);//res=false
```

1.7.4 OPERADORES UNITARIOS

Operador	Uso	Operación
~	~A	Complemento a 1 de A
-	-A	Cambio de signo del operando
--	A--	Decremento de A
++	A++	Incremento de A
!	! A	Not A (ya visto)

1.7.4 OPERADORES UNITARIOS

```
int  m=2, n=5;
m++; //  m=3
n--;  //    n=4
```

1.7.5 OPERADORES DE BITS

Operador	Uso	Operación
&	A & B	AND lógico. A AND B.
|	A | B	OR lógico. A OR B.
^	A ^ B	XOR lógico. A XOR B.
<<	A << B	Desplazamiento a la izquierda de A B bits rellenando con ceros por la derecha.
>>	A >> B	Desplazamiento a la derecha de A B bits rellenando con el BIT de signo por la izquierda.
>>>	A >>> B	Desplazamiento a la derecha de A B bits rellenando con ceros por la izquierda.

1.7.5 **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
```

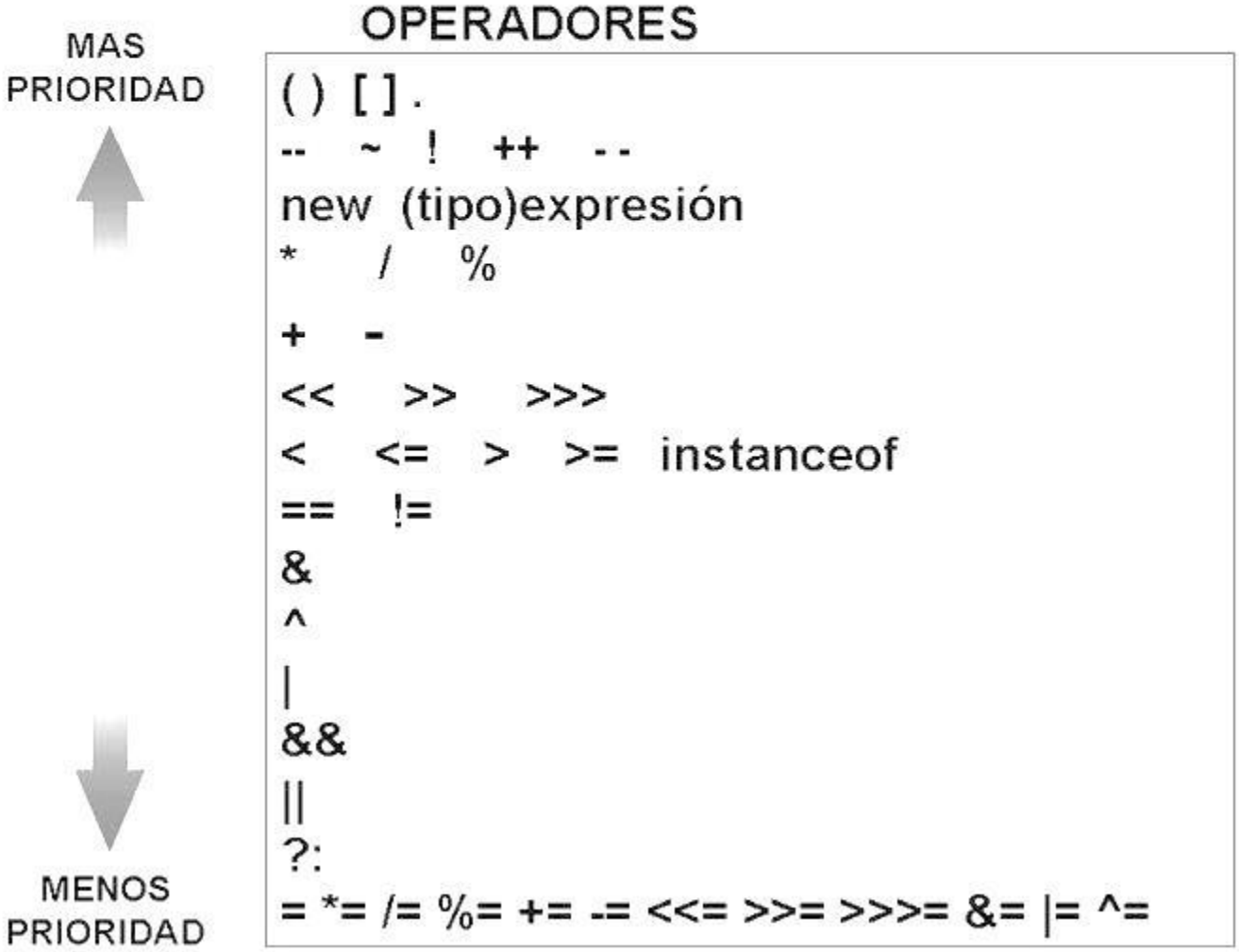
1.7.6 OPERADORES DE ASIGNACIÓN

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.

1.7.6 **OPERADORES DE ASIGNACIÓN**

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

1.7.7 PRECEDENCIA DE OPERADORES



1.8 CONVERSIONES DE TIPOS (CAST)

Conversiones de tipos

- **Conversiones implícitas**. Automático. Requiere que la variable destino (la colocada a la izquierda) tenga más precisión que la variable origen (situada a la derecha).
- **Conversiones explícitas**. Forzado por el programador 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*.