



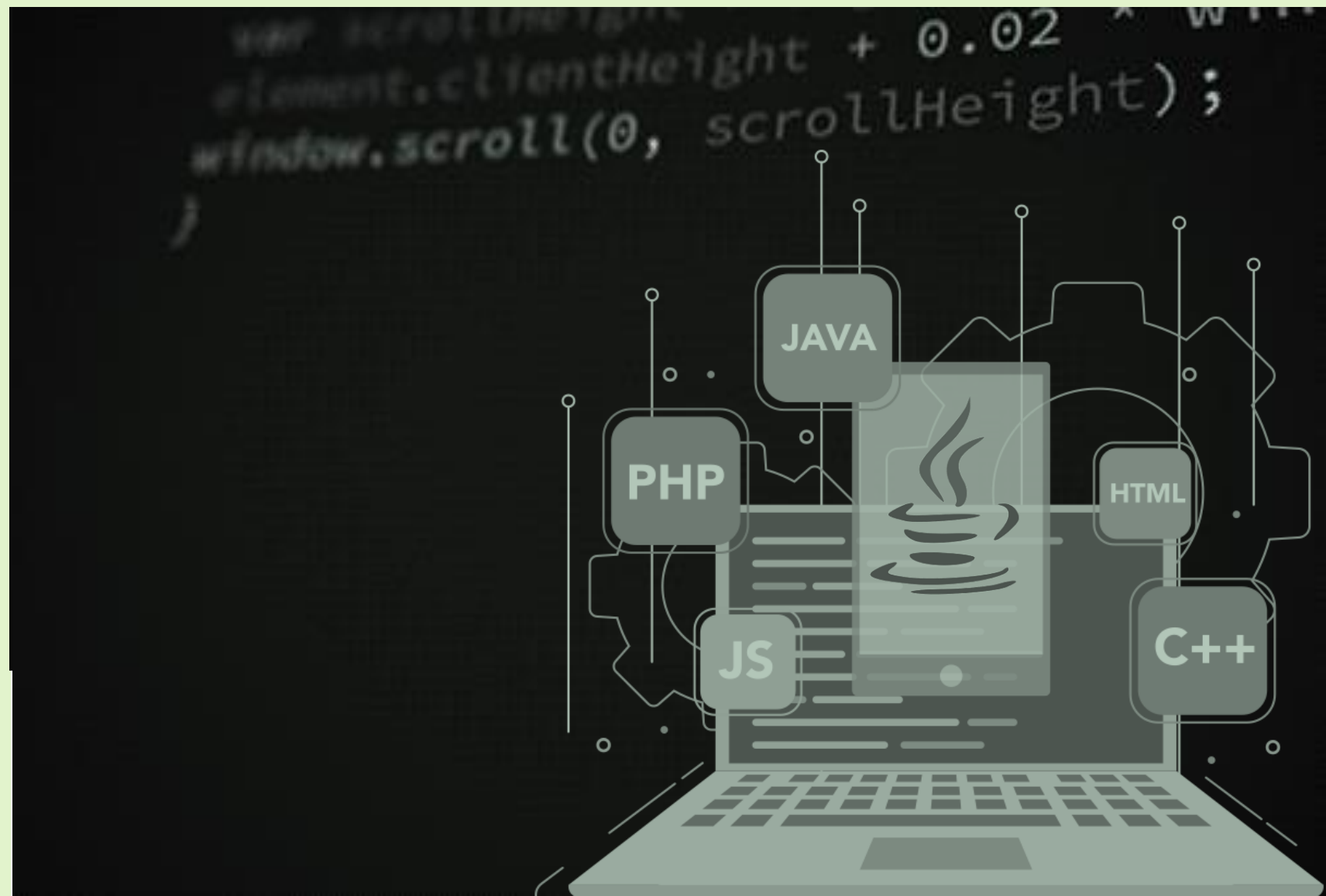
EFA
MORATALAZ

*1º CFGS Desarrollo de
Aplicaciones Web*

PROGRAMACIÓN

*DANIEL GONZÁLEZ-CALERO
JIMÉNEZ*

UT1 – ELEMENTOS DE UN PROGRAMA INFORMÁTICO





EFA
MORATALAZ

*1º CFGS Desarrollo de
Aplicaciones Web*

PROGRAMACIÓN

INDICE

UT1 – ELEMENTOS DE UN PROGRAMA INFORMÁTICO

1. CONCEPTO DE PROGRAMA Y LENGUAJE DE PROGRAMACIÓN.
2. EL LENGUAJE JAVA
3. JDK, JRE Y VARIABLES DE ENTORNO
4. BLOQUES PRINCIPALES DE UN PROGRAMA
5. TIPOS DE DATOS SIMPLES
6. VARIABLES Y CONSTANTES
7. OPERADORES
8. CONVERSIONES DE TIPOS
9. CLASE MATH

CONCEPTO DE PROGRAMA Y LENGUAJE DE PROGRAMACIÓN

1

PROGRAMA

Una serie de instrucciones ordenadas con una finalidad concreta que realizan una función determinada.

Para programar solo es necesario pensar **QUE** queremos hacer y **COMO** vamos a hacerlo, pensando siempre en el objetivo que queremos obtener.

Una vez que tengamos el QUE y el COMO solo tenemos que elegir un lenguaje de codificación para traducirlo.

```
1 public class Variables {  
2     public static void main(String[] args) {  
3         // DECLARA TUS VARIABLES  
4  
5         // Variables de tipo entero:  
6         int a;  
7         int edad;  
8         int numero;  
9     }  
10 }
```



EJEMPLOS

Hacer una tortilla

Instruc. 1: Comprar huevos
Instruc. 2: Comprar aceite
Instruc. 3: Comprar patatas
Instruc. 4: Pelar patatas
Instruc. 5: Batir huevos
Instruc. 6: Calentar aceite

...

OBJETIVO

Hacer una tortilla



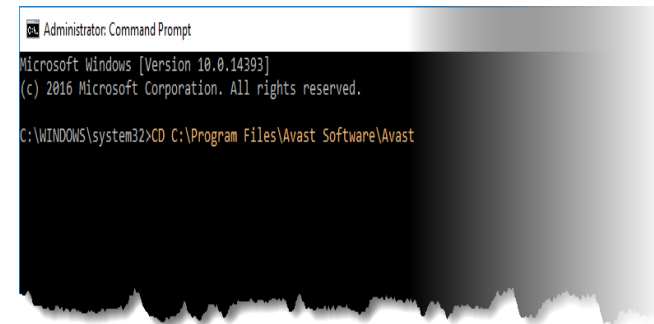
Mostrar datos de un alumno por pantalla

Instruc. 1: Pedir nombre
Instruc. 2: Pedir apellidos
Instruc. 3: Pedir edad
Instruc. 4: Mostrar nombre
Instruc. 5: Mostrar apellidos
Instruc. 6: Mostrar edad

...

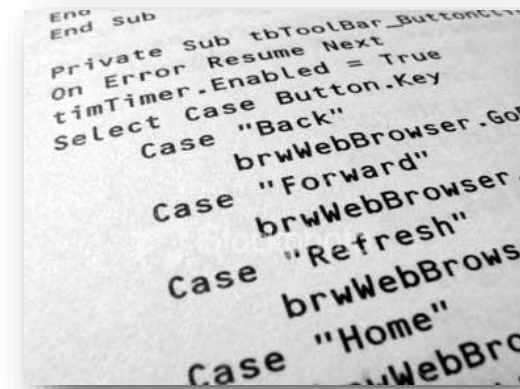
OBJETIVO:

Mostrar datos de un alumno por pantalla



CÓDIGO FUENTE:

- Conjunto de instrucciones que forman un programa informático
- Está escrito en un lenguaje de programación
- Los humanos son capaces de entender este código

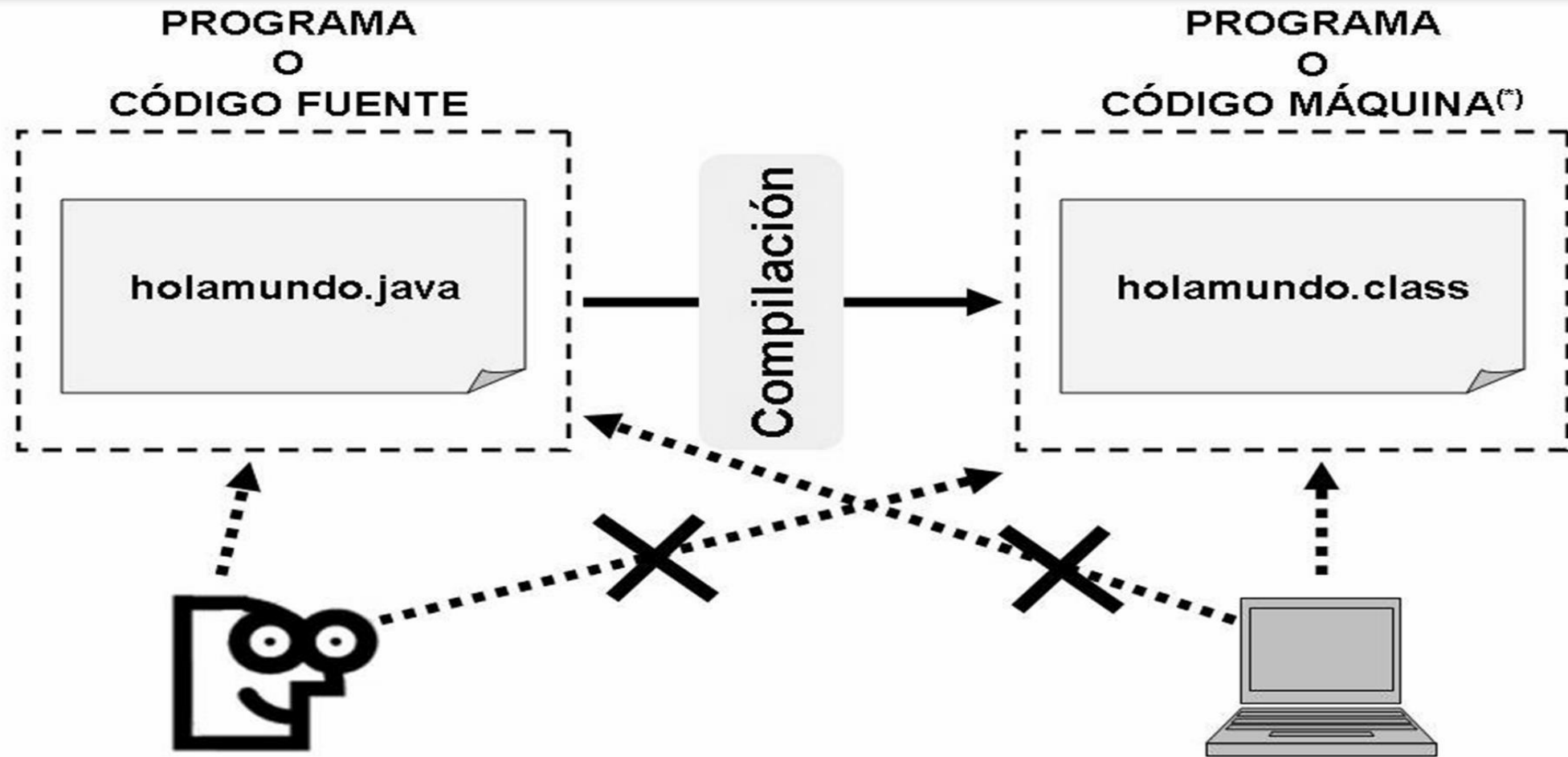


```
End Sub  
End Sub  
Private Sub tbToolBar_ButtonClick  
On Error Resume Next  
timTimer.Enabled = True  
Select Case Button.Key  
Case "Back"  
    brwWebBrowser.GoBack  
Case "Forward"  
    brwWebBrowser.GoForward  
Case "Refresh"  
    brwWebBrowser.Refresh  
Case "Home"  
    brwWebBrowser.Home
```

CÓDIGO MÁQUINA:

- Código fuente que ha sufrido un proceso de compilación
- Está escrito en binario
- Solo lo pueden entender las máquinas





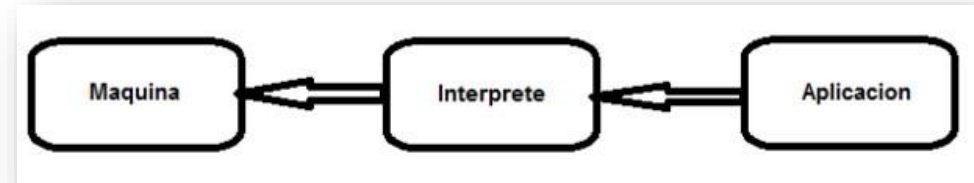
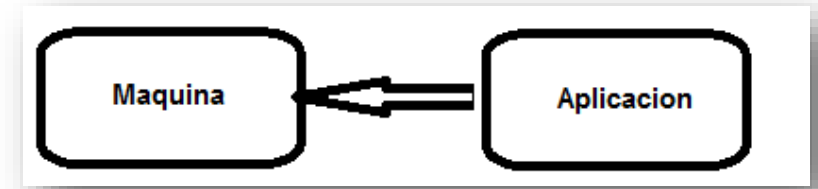
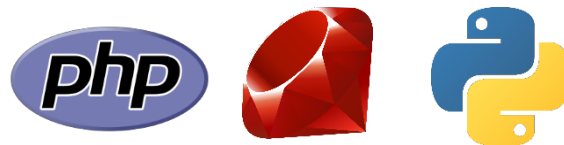
COMPILADOR:

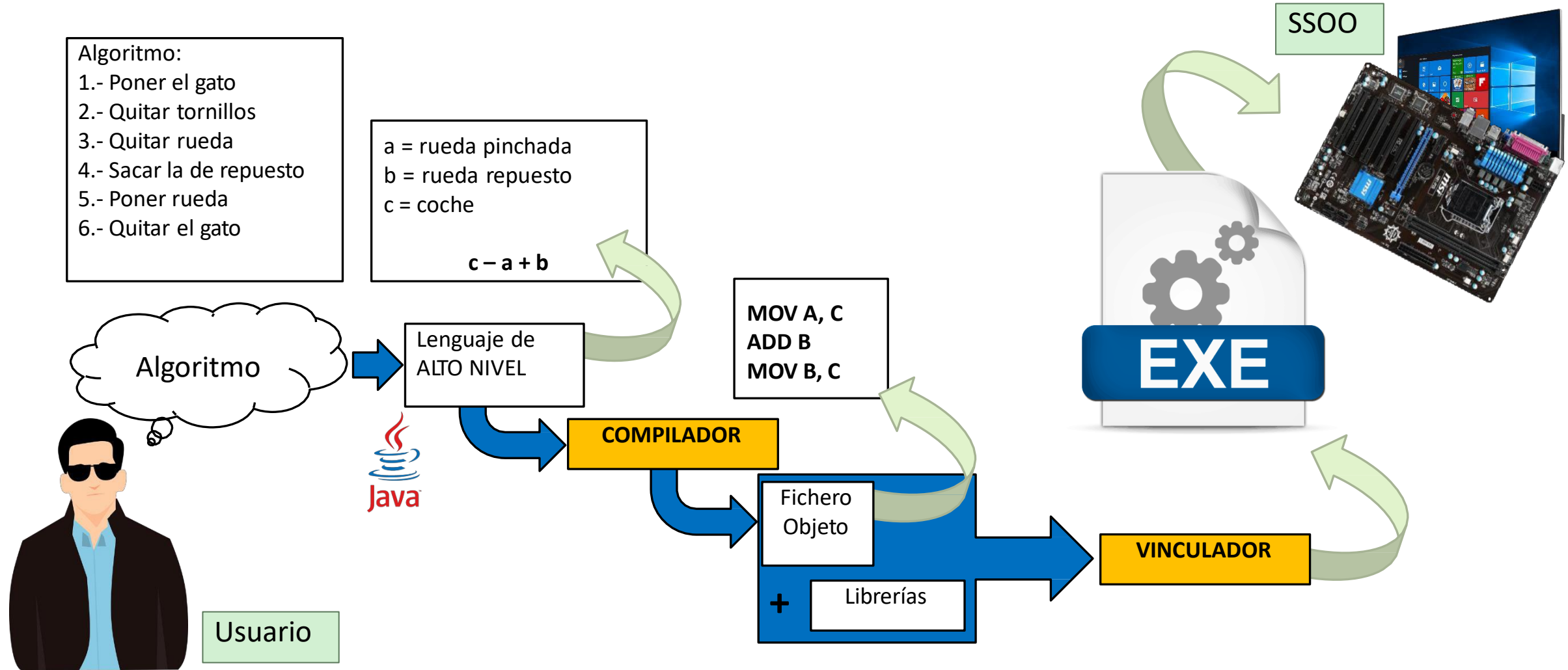
- Los compiladores son programas específicos para un lenguaje de programación
- Transforman el programa fuente en un programa ejecutable por la máquina destino
- No es posible compilar un programa escrito en lenguaje Java con un compilador de C porque éste no lo entendería.
- Lenguajes compilados:

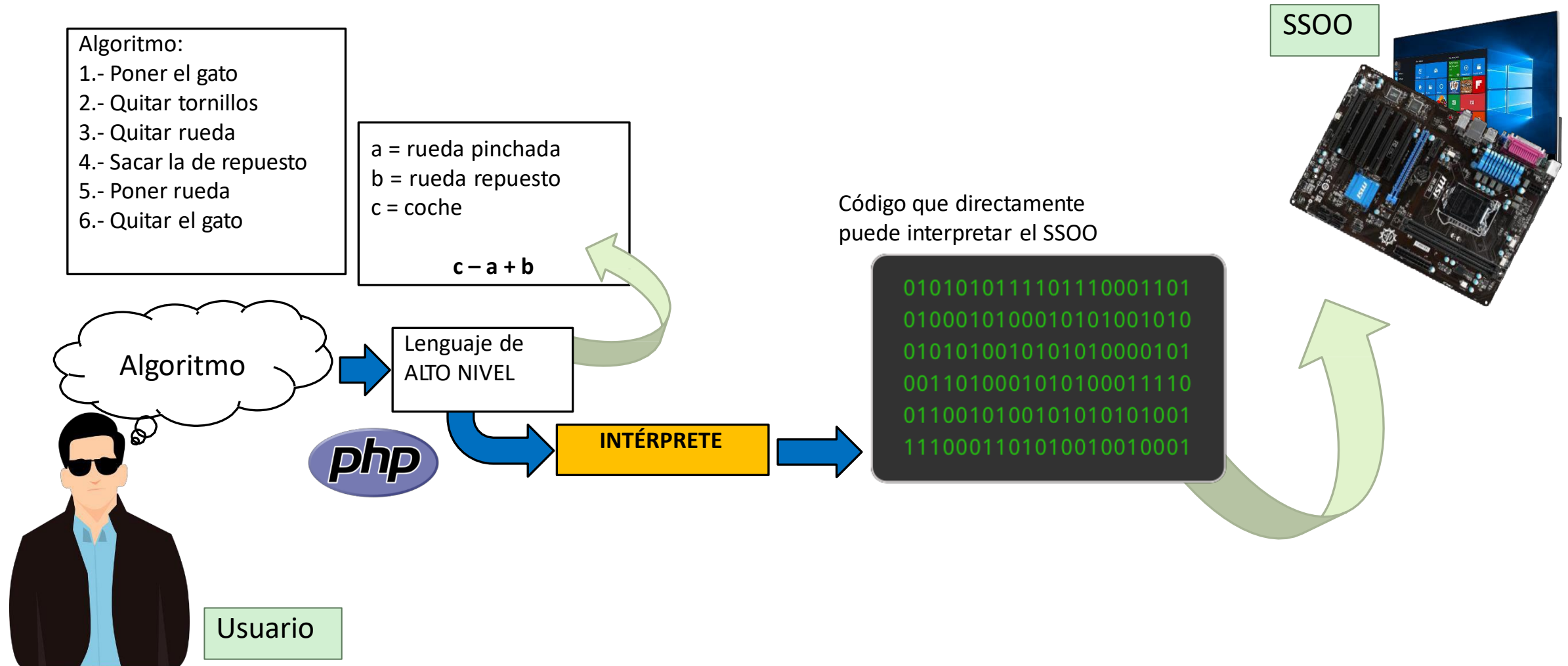


INTÉRPRETE:

- 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.
- La ventaja de los intérpretes es que la depuración y corrección de errores del programa es mucho más sencilla que con los compiladores.
- Lenguajes interpretados:





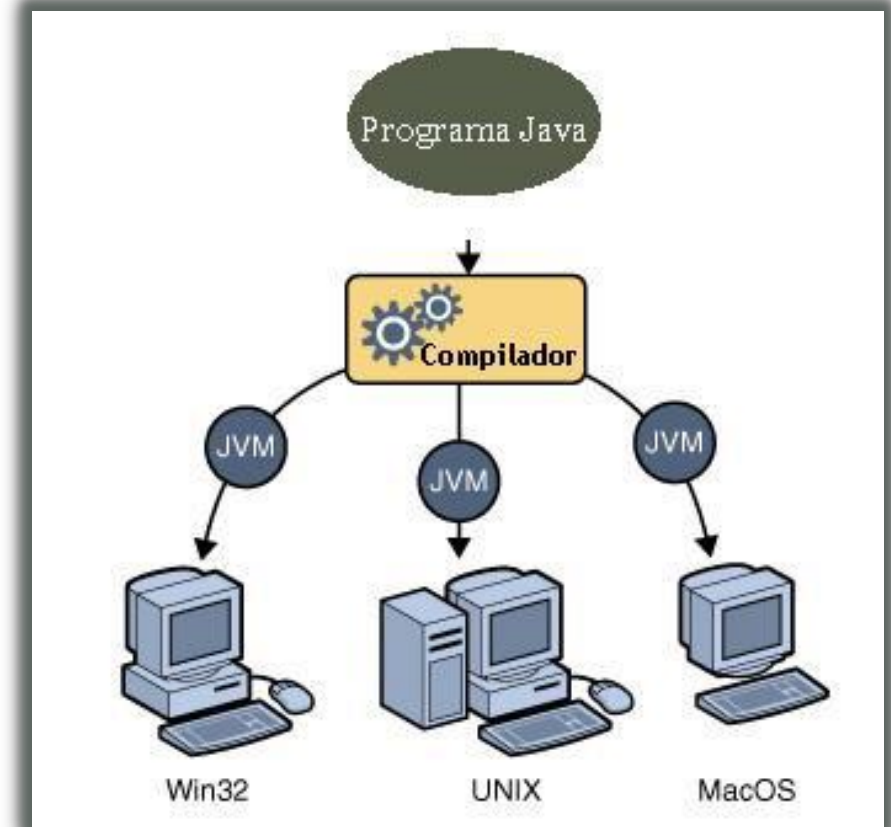


EL LENGUAJE JAVA

A large, stylized number 2 in a light green color, centered within a dark green rounded square. The number has a slight shadow effect, giving it a 3D appearance.



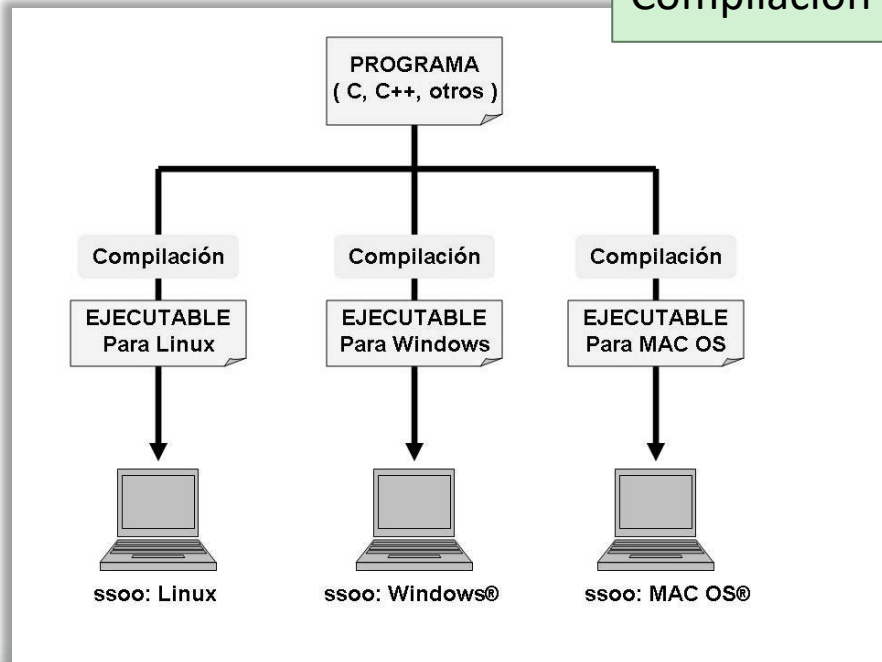
- Java es uno de los lenguajes más utilizados en la actualidad. ¿Porqué?
 - Por el futuro y presente que tiene
 - Es un lenguaje sencillo
 - Es un lenguaje orientado a objetos
 - Es independiente de la plataforma
- El éxito de Java radica en que es un **lenguaje multiplataforma**
- Java utiliza una máquina virtual en el sistema destino y por lo tanto no hace falta recompilar de nuevo las aplicaciones para cada sistema operativo.
- Este código intermedio o *bytecode* es independiente de la arquitectura y por lo tanto puede ser ejecutado en cualquier sistema.



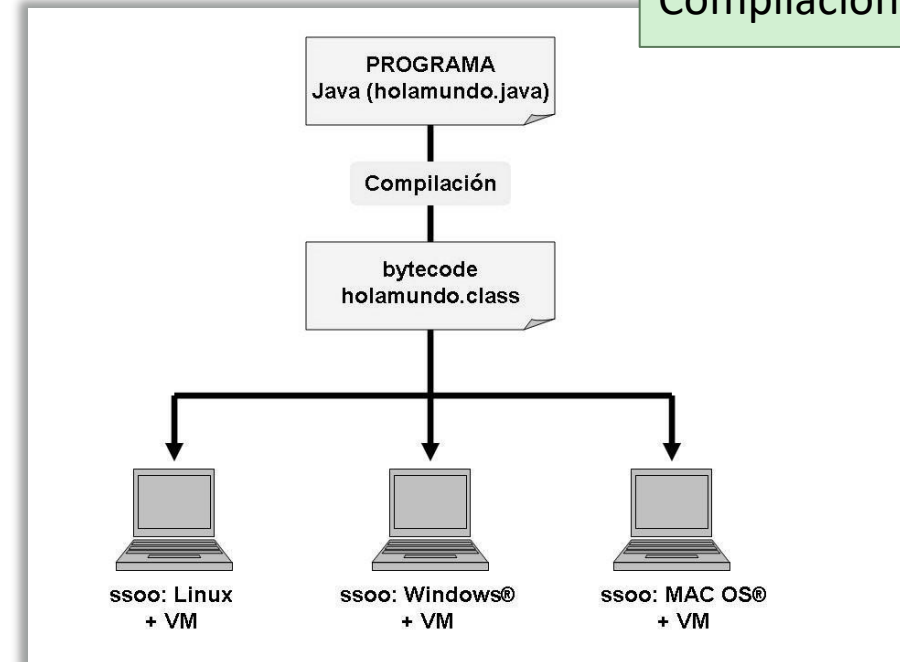


- En Java, una vez compilado el programa, se puede ejecutar en cualquier plataforma solamente con tener instalada la máquina virtual (*Java Virtual Machine – JVM*) de Java
- Sin embargo en C, C++ u otro lenguaje, se debe recompilar el programa para el sistema destino con la consiguiente pérdida de flexibilidad

Compilación en C

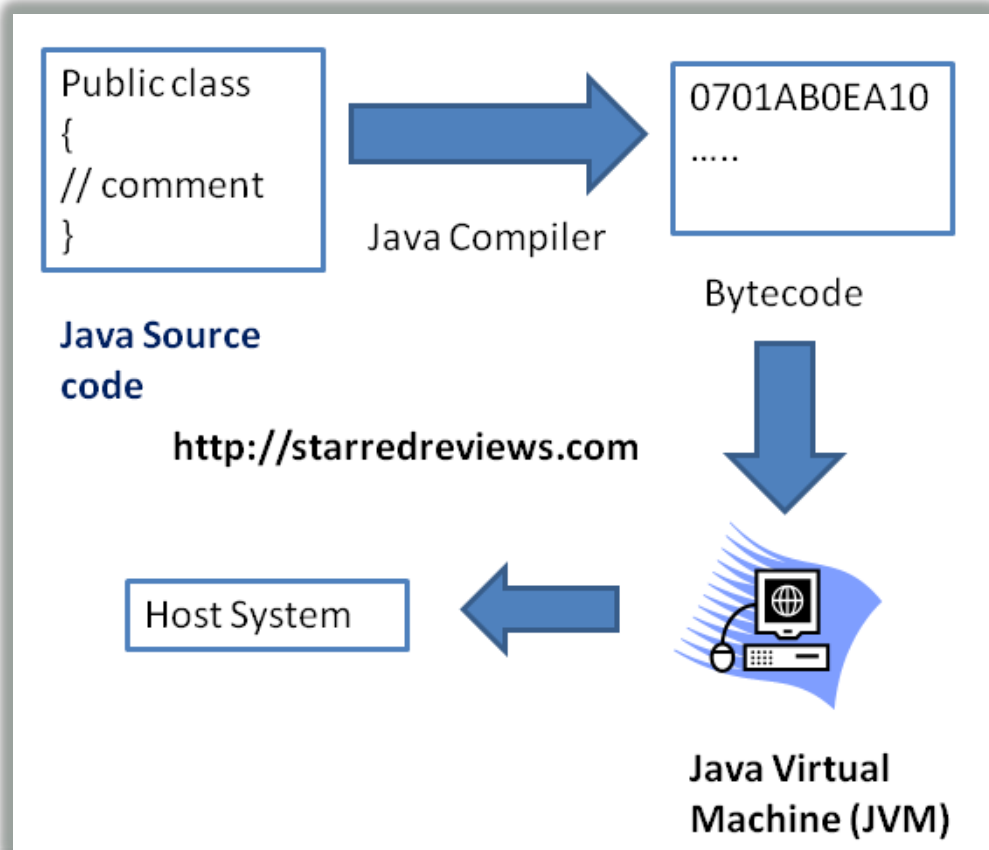


Compilación en Java

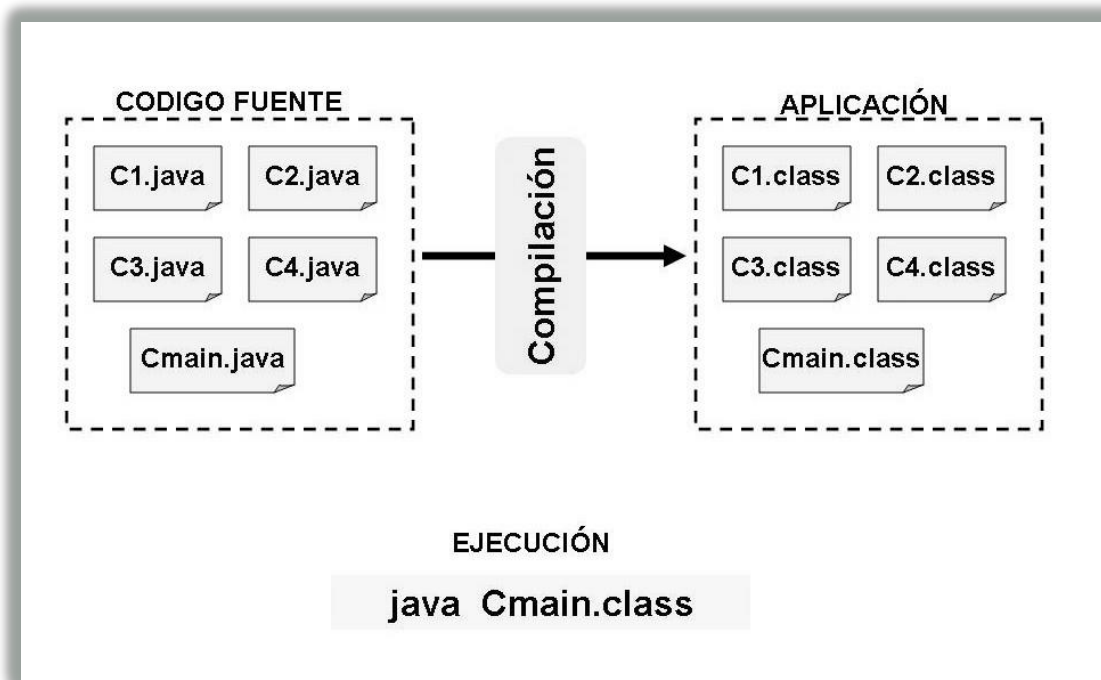




- Los programas o aplicaciones en Java se componen de una serie de **ficheros .class** que son ficheros en *bytecode*



- 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.



JDK, JRE Y VARIABLES DE ENTORNO





➤ JVM – Java Virtual Machine

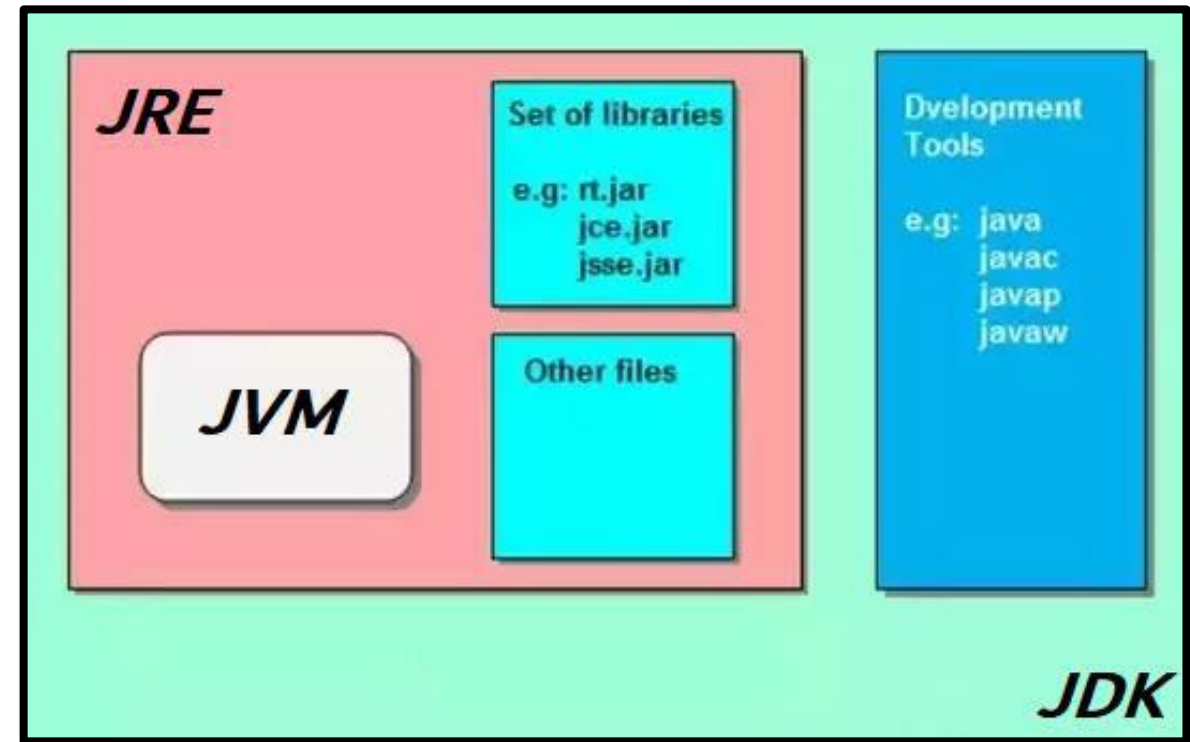
Máquina virtual que es capaz de interpretar y ejecutar las instrucciones en un código binario especial o bytecode generado por el compilador

➤ JRE – Java Runtime Environment

Constituye el entorno mínimo en un sistema operativo para poder ejecutar aplicaciones basadas en Java. Incluye la JVM y librerías necesarias para ello.

➤ JDK – Java Development Kit

Es necesario para poder desarrollar aplicaciones mediante lenguaje de programación Java.
JDK incluye a JRE





El JDK contiene, entre otras, las siguientes herramientas de consola:

- **java** - Es la máquina virtual de Java.
- **javac** - Es el compilador de Java.
- **javap** - Es un desensamblador de clases.
- **jdb** - El depurador de consola de Java
- **javadoc** - Es el generador de documentación.
- **appletviewer** - Visor de *Applets*.

Y muchas más opciones. Siéntete libre de probar todas las opciones!

Ejemplo: (aunque nosotros no trabajaremos con el cmd y lo haremos con un IDE)

```
@MAC:~$ java -version
java version "1.6.0_32"
OpenJDK Runtime Environment (IcedTea6 1.13.4) (6b32-1.13.4-4ubuntu0.12.04.2)
OpenJDK 64-Bit Server VM (build 23.25-b01, mixed mode)
```



A la hora de trabajar con Java, es importante tener presente que hay dos variables de entorno cuyos valores tienen que estar correctamente configurados para permitir llevar a cabo la compilación, ejecución y generación de paquetes *jar* de una forma mucho más cómoda:

- **PATH** - Es la variable del sistema utilizada por el sistema operativo para buscar los ejecutables de las diferentes aplicaciones desde el intérprete de comandos o el terminal de Linux.
- **CLASSPATH** - Es la variable del sistema utilizada por el entorno de ejecución de Java (JRE) para buscar los paquetes y las clases definidas por el usuario.

Se tiene que configurar estas variables para que el funcionamiento y ejecución del IDE sea perfecto y más aún si la compilación se hace desde línea de comandos.

Para configurar este tipo de variables hay que entrar en la configuración del sistema operativo, concretamente en las opciones avanzadas del sistema.



- Videotutorial para la instalación del IDE de ECLIPSE y la previa instalación del JDK para desarrollar

Como instalar y configurar Eclipse y JDK 8

The Java 8 logo, featuring the Java logo and the text "Java 8 Advanced Streams".

- Instalar Eclipse en windows 10 para desarrolladores Java.
- Instalar JDK 8 y configurarlo en nuestro eclipse.

Guware



Antes de empezar a programar, es necesario conocer los diferentes tipos de archivos generados que se van a manejar y las extensiones asociadas:

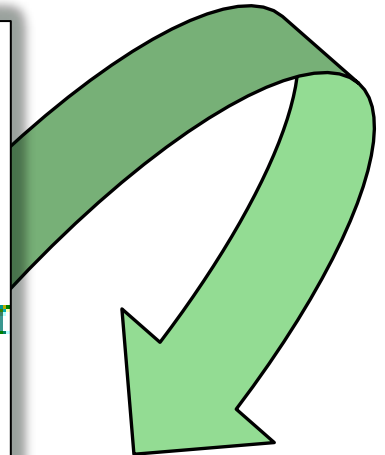
- Cuando se escribe *código fuente* en Java, este se almacena en un archivo de texto plano con la extensión **.java**
- Como resultado de compilar exitosamente un programa escrito en un **.java** obtendremos y archivo contenedor de código intermedio (bytecode) con extensión **.class**
- A diferencia de otras tecnologías en las que el resultado de una correcta compilación es un archivo ejecutable, Java genera un archivo independiente por cada clase del proyecto **.class**. Esto puede hacer muy incómodo el trabajo con programas que contengan muchas clases, por eso ofrece la posibilidad de empaquetarlos todos en un archivo **.jar**
- Posteriormente para facilitar la implantación de las aplicaciones, Java ofrece:
 - **.war** (*Web Application Archive*): contiene todos los archivos que forman una aplicación web desarrollada con Java. Constituye una aplicación web.
 - **.ear** (*Enterprise Archive*): empaquetan módulos en un archivo que facilita el despliegue de las aplicaciones en un único paso sobre un servidor. Es decir, el **.ear** contiene un **.war** y unos descriptores de despliegue llamados **META-INF**

BLOQUES PRINCIPALES DE UN PROGRAMA



- Programa de inicio por excelencia en cualquier lenguaje de programación

```
1 public class Holamundo {  
2     /* programa Holamundo */  
3     public static void main(String[] args) {  
4         /* lo único que hace este programa es r  
5         System.out.println("Hola Mundo");  
6     }  
7 }
```



```
1 public class Holamundo {  
2     ...  
3     ...  
4 }
```

- En java generalmente, cada clase es un fichero distinto
- Las clases tienen el mismo nombre que su fichero .java y es importante que mayúsculas y minúsculas coincidan

- El código Java en las clases se agrupa en **métodos o funciones**.
- Cuando Java va a ejecutar el código de una clase, lo primero que hace es buscar el método **main** de dicha clase para ejecutarlo.
- El método **main** tiene las siguientes particularidades:
 - Es público (**public**). Esto es así para poder llamarlo desde cualquier lado.
 - Es estático (**static**). Al ser static se le puede llamar sin tener que instanciar la clase.
 - No devuelve ningún valor (**void**).
 - Admite una serie de parámetros (**String [] args**) que en este ejemplo concreto no son utilizados.
 - El método **main** abarca todo el código contenido entre las **llaves**.

```
1 public static void main(String [ ] args) {  
2     ...  
3     ...  
4 }
```

- La clase **PrintStream** permite dar formato a la salida de un texto.
- Cuando la aplicación java arranca, automáticamente se crea un objeto **System.out** que permite establecer comunicación entre la aplicación java y el sistema operativo . Gracias a esto, ya podemos utilizar los métodos de la clase PrintStream:
 - **print()** → Imprime la secuencia o variables que se le indican entre paréntesis.
 - **println()** → Funciona igual que el anterior pero le añade un salto de línea al principio.
- 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).

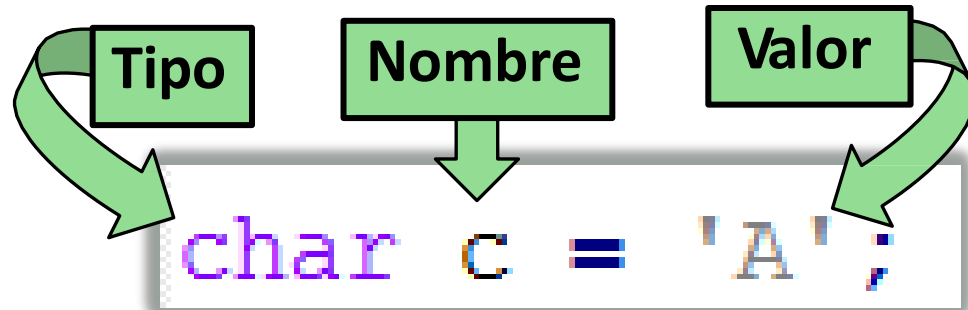
```
1 System.out.println("Hola mundo");
```


TIPOS DE DATOS SIMPLES

5

Tipos de datos	Información representada	Rango	Descripción
Byte	Datos enteros	-128 a 127	Se utilizan 8 bits (1 byte) para almacenar el dato.
Short	Datos enteros	-32.768 a 32.767	Dato de 16 bits de longitud
Int	Datos enteros	-2.147 M a 21.147 M	Dato de 32 bits de longitud
Long	Datos enteros	-9.223.372 B a 9.223.372 B	Dato de 64 bits de longitud
Char	Caracteres	0 a 65535	Este rango es para representar números UNICODE, los ASCII se representan con valores de 0 al 127.
Float	Con coma de 32 bits	Precisión de 7 dígitos	Dato en coma flotante de 32 bits
Double	Con coma de 64 bits	Precisión de 16 dígitos	Dato en coma flotante de 64 bits
Boolean	Valores Booleanos	True / false	Utilizado para evaluar si el resultado es verdadero o falso
String	Cadena de caracteres	2.147 M caracteres	Utilizado para representar cadenas de texto

- Los tipos de datos se utilizan generalmente al declarar **variables** y son necesarios para que el compilador conozca de antemano el tipo de información que va a contener una variable.



```
1 // declarando un número entero con valor 5
2 int x = 5;
3
4 // declarando un número en coma flotante de 64 bits con valor 3,14
5 double y = 3,14;
6
7 // declarando un char con valor A
8 char c = 'A';
```

VARIABLES Y CONSTANTES



- Una variable es una zona de memoria donde se puede almacenar información del tipo que desee el programador.
- Al crear variables se reserva memoria para albergar datos
- Todas las variables que están dentro de un bloque son visibles y existen **dentro** de dicho bloque

```
1 public class Suma{
2     int n1 = 50;    // variable con visibilidad toda la clase
3     public static void main(String [] args){
4         int n2 = 30, suma = 0; // variables con visibilidad en el método main
5         suma = n1 + n2;
6         System.out.println("LA SUMA ES: " + suma);
7     }
8 }
```

- Las constantes se utilizan en datos que **nunca varían**
 - Ejemplo: *IVA, PI, etc.*
- Las constantes se declaran en **mayúscula**
 - *esto se realiza como norma de estilo (no es obligatorio)*
- Las constantes se declaran siguiendo el siguiente formato:

```
1  /* final [static] <tipo de dato> <nombre constante> = <valor>; */
2  final double PI = 3.141592;
3  final int IVA = 21;
```

❖ final

- Indica que es una constante

❖ static

- Implica que solo existirá una copia de dicha constante, aunque existan varias instancias de objetos

OPERADORES



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

```


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  int m = 2, n = 5;
2  boolean res;
3  res = m > n;           //res=false
4  res = m < n;           //res=true
5  res = m >= n;          //res=false
6  res = m <= n;          //res=true
7  res = m == n;          //res=false
8  res = m != n;          //res=true

```

Operador	Uso	Operación
~	~A	Complemento a 1 de A
-	-A	Cambio de signo del operando
--	A--	Decremento de A
++	A++	Incremento de A

```

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

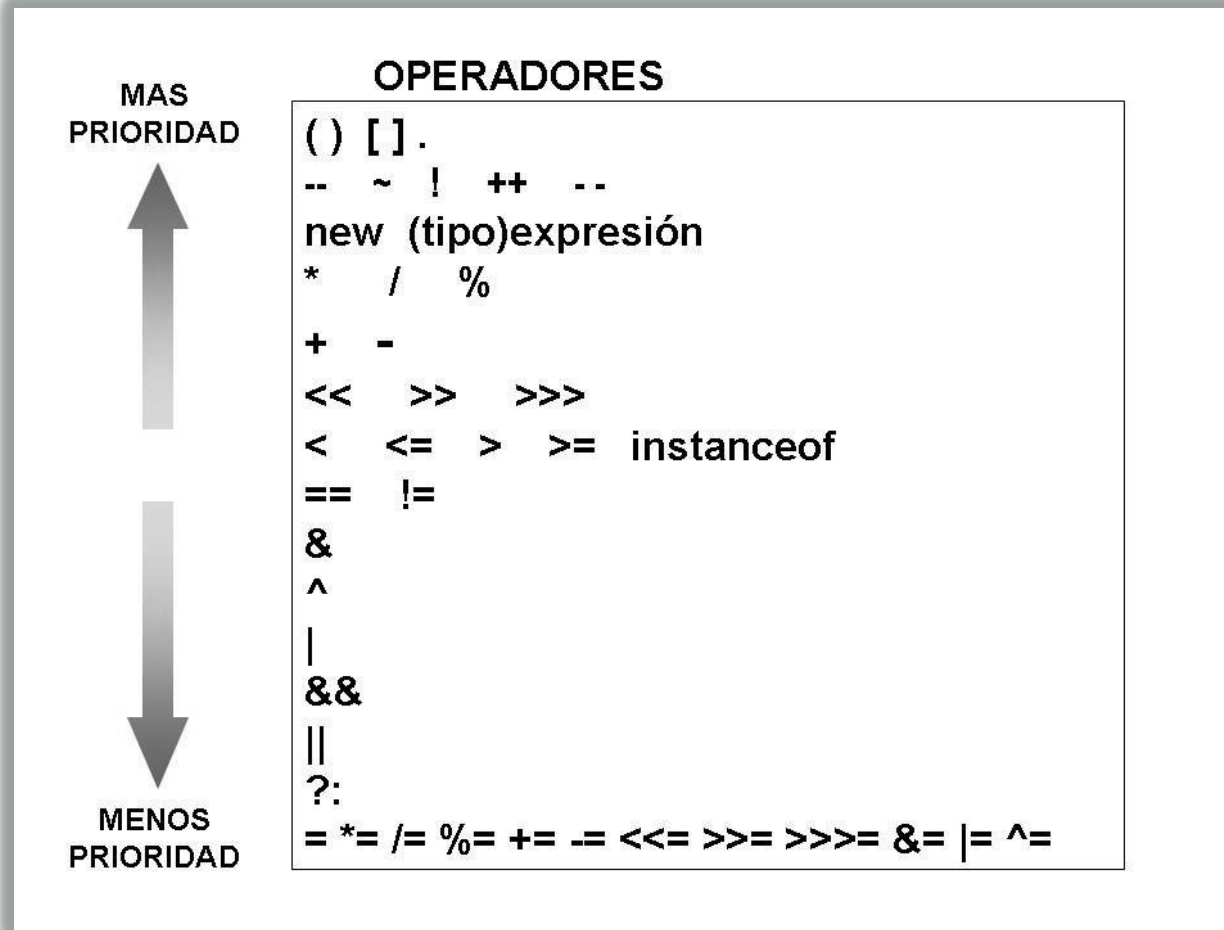
```

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

```



```
1 int a = 4;
2 a = a + 3 * 2;
3 System.out.println(a); // ¿Resultado?
```

CONVERSIONES DE TIPOS



- El programador puede forzar la conversión mediante una operación llamada **cast o casting**

(tipo) expresión

- El programador debe ser consciente de la pérdida de información que se produce

```
1  int i_dato = 5;  
2  byte b_dato;  
3  b_dato = (byte)i_dato;  
4  System.out.println(b_dato);    // Imprime 5
```

- No se pueden realizar conversiones entre enteros y booleanos o reales y booleanos

CLASE MATH



- Es una clase que nos proporciona una serie de funciones y variables matemáticas que tienen mucha utilidad en Java. Esta clase viene incluida en las nuevas versiones y no hace falta importarla.
- Para utilizarla tenemos que poner:

Math.método(parámetros);

- Constante definidas en dicha clase:

CONSTANTE	DESCRIPCIÓN
PI	Devuelve el valor de PI. Es un double. 3,14159
E	Devuelve el valor de E. Es un double. 2,71828

MÉTODO	DESCRIPCIÓN	PARÁMETROS	TIPO DE SALIDA
abs	Devuelve el valor absoluto de un número	Int, double, float o long	El mismo que introduces
exp	Devuelve el exponencial de un número	Double	Double
log	Devuelve el logaritmo natural en base e de un nº	Double	Double
max, min	Devuelve el mayor o el menor de dos valores	2 parámetros: int, double, float	El mismo que de entrada
sqrt	Devuelve la raíz cuadrada de un número	Double o float	Long o int
ceil, floor	Devuelve el entero más cercano por arriba o por debajo	Double	Double
random	Devuelve un número aleatorio entre 0 y 1. Se pueden cambiar el rango de generación.	Ninguno	Double
round	Devuelve el entero más cercano	Double o float	Long o int
sin, cos, tan	Devuelve el seno, coseno o tangente de un ángulo	Double	Double
arcos, asin, atan	Devuelve el arco-seno, arco-coseno o arco-tangente de un ángulo en radianes	Double	Double
pow	Devuelve un número elevado a un exponente	2 Double	Double