¿Que es Java?

Java es una tecnología pensada para desarrollo de aplicaciones de gran envergadura, altamente escalable con gran integración con otras tecnologías y muy robusta.

Sus principales características son:

- Lenguaje orientado a objetos.
- Sintaxis basada en C/C++.
- Multiplataforma.
- Manejo de memoria automático.
- Evolución permanente.

Idea original

- Año 1990: Sun Microsystems necesitaba una herramienta de programación.
- Para lograr esto involucraron a James Gosling quien paso a ser el padre-fundador de Java y el fue quien tuvo la visión de un lenguaje de programación que perseguía la idea de WORA(“Write once, run anywhere“).
- El proyecto fue técnicamente un éxito pero no fue lo mismo en el ámbito comercial.
- Año 1995: Java ve una posibilidad en el ámbito de internet y entra al negocio con los Applets.
- Java empieza a tomar fuerza y empieza a tomar mas potencia en otros contextos, gracias a su poder basado en WORA.

Historia

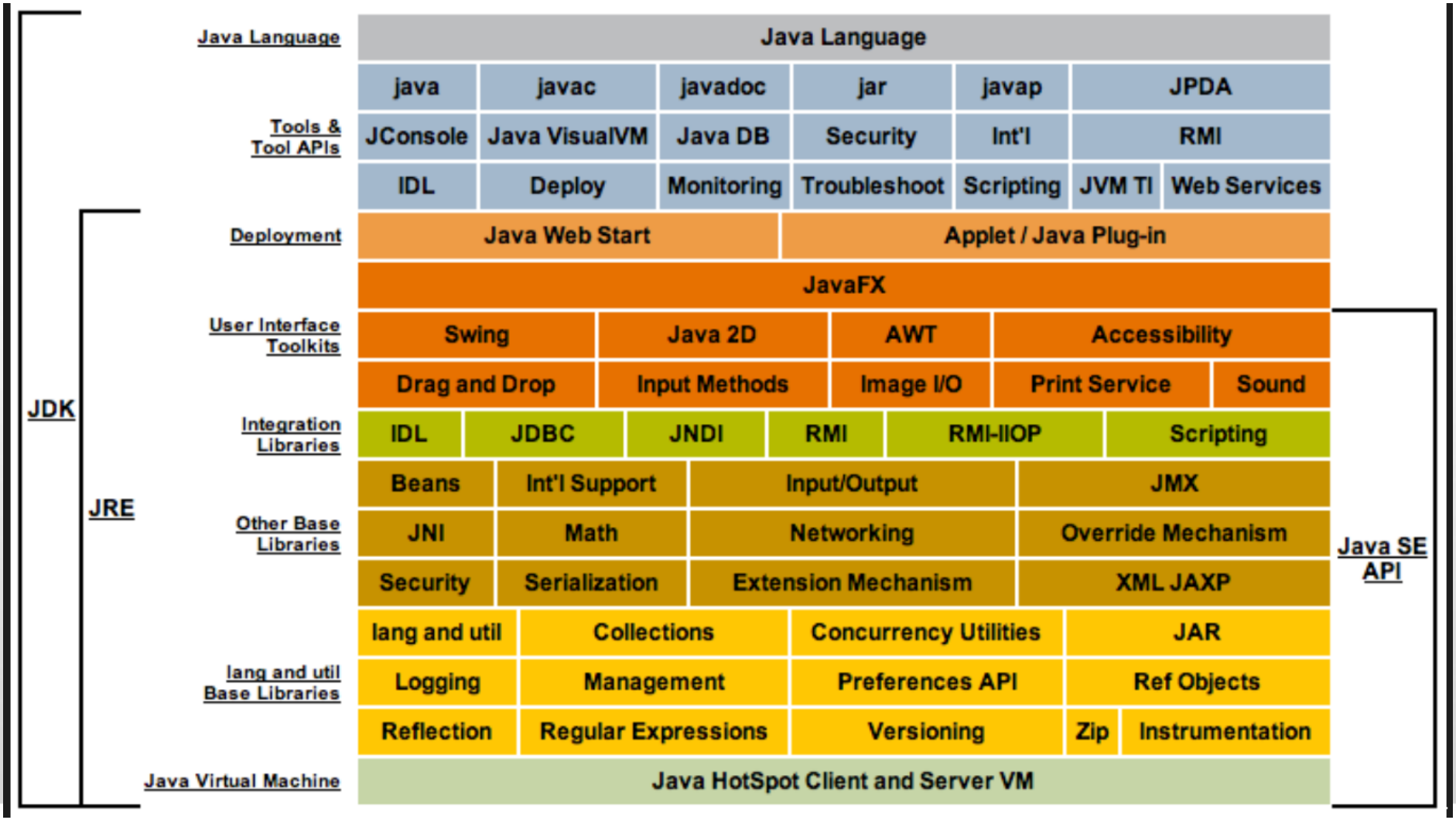
- **JDK 1.0** (1996): Primer lanzamiento del lenguaje Java.
- **JDK 1.1** (1997): Mejora de la versión anterior.
- **J2SE 1.2** (1998): Esta y las siguientes versiones fueron recogidas bajo la denominación Java 2 y el nombre "J2SE" (Java 2 Platform, Standard Edition), reemplazó a JDK para distinguir la plataforma base de J2EE (Java 2 Platform, Enterprise Edition) y J2ME (Java 2 Platform, Micro Edition). Incluyó distintas mejoras.
- **J2SE 1.3** (2000): Mejora de la versión anterior.
- **J2SE 1.4** (2002): Mejora de la versión anterior.
- **J2SE 5.0** (2004): Originalmente numerada 1.5, esta notación aún es usada en ocasiones. Mejora de la versión anterior.
- **Java SE 6** (2006): Sun cambió el nombre "J2SE" por Java SE y eliminó el ".0" del número de versión. Mejora de la versión anterior.
- **Java SE 7** (2011): Nueva versión que mejora la anterior.
- **Java SE 8**: Nueva versión que se espera para Marzo 2014.

Distribuciones de Java

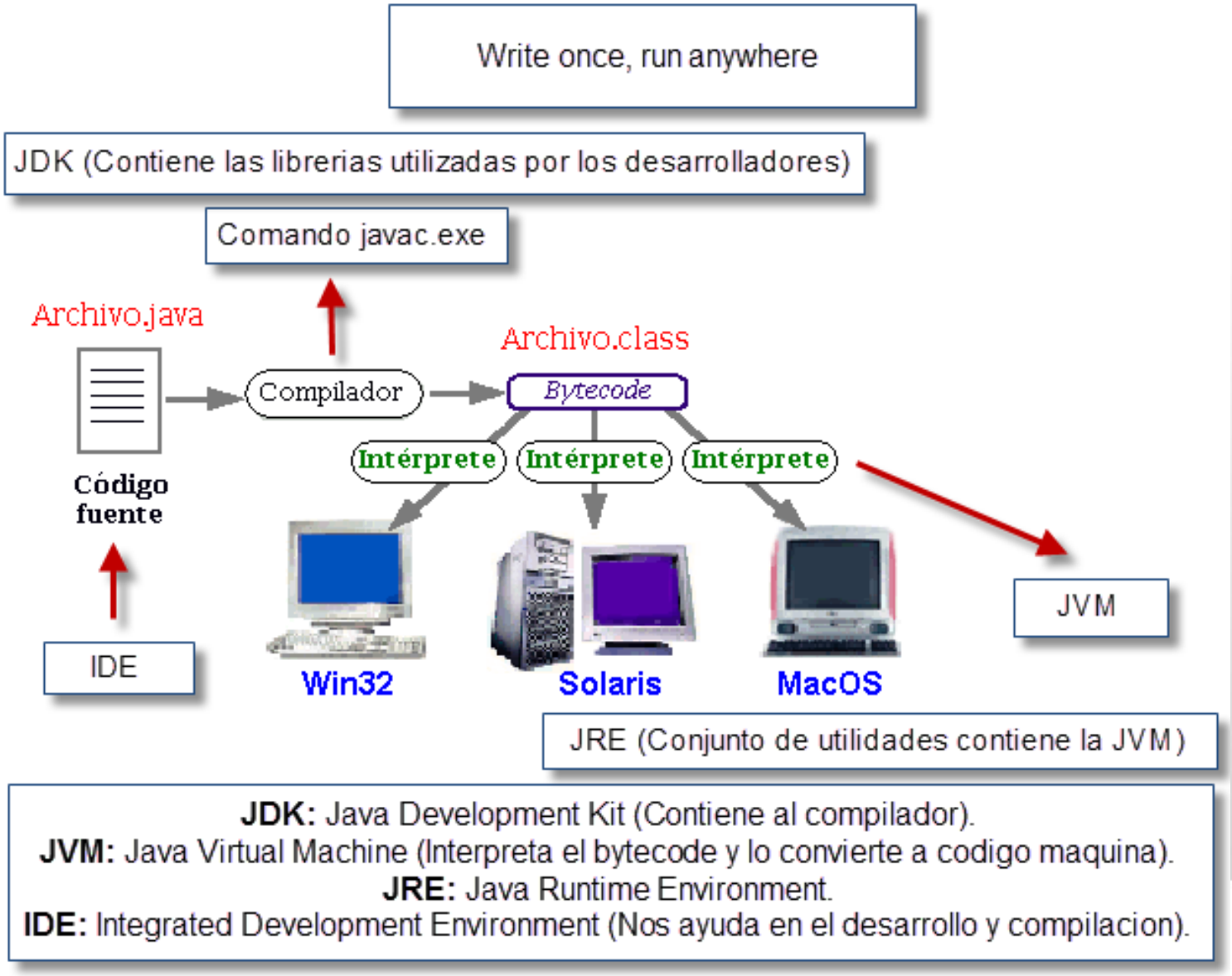
Java esta divido en 3 áreas bien definidas.

- JME (Mobile / Wireless): Esta área tiene como objetivo el desarrollo de aplicaciones móviles, tales como GPS, Handhelds (Por ejemplo la conocida Palm), celulares y otros dispositivos móviles programables. JME significa **Java Micro Edition**.
- JSE (Core / Desktop): Esta área tiene como objetivo el desarrollo de aplicaciones de escritorio, similares a las aplicaciones tipo ventanas creadas con Visual Basic o Delphi. Incluye la funcionalidad básica del lenguaje como manejo de clases, colecciones, entrada/salida, acceso a base de datos, manejo de sockets, hilos de ejecución, etc. JSE significa **Java Standard Edition**.
- JEE (Enterprise / Server): Esta área tiene como objetivo el desarrollo de aplicaciones empresariales, de gran tamaño. Contempla ambientes Webs, como los ambientes manejados por servidores de aplicaciones. Las tecnologías principales incluidas en esta área son Servlets, JSP y EJB, entre otras cosas. JEE significa **Java Enterprise Edition**.

Java packages



Panorama general



Primer programa

File → New Project → Java Category → Java Application

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5  package clase1;
6
7  /**
8   *
9   * @author ldebello
10  */
11 public class Clase1 {
12
13     /**
14      * @param args the command line arguments
15      */
16     public static void main(String[] args) {
17         // TODO code application logic here
18         System.out.println("Hola Mundo");
19     }
20 }
```

Comentarios

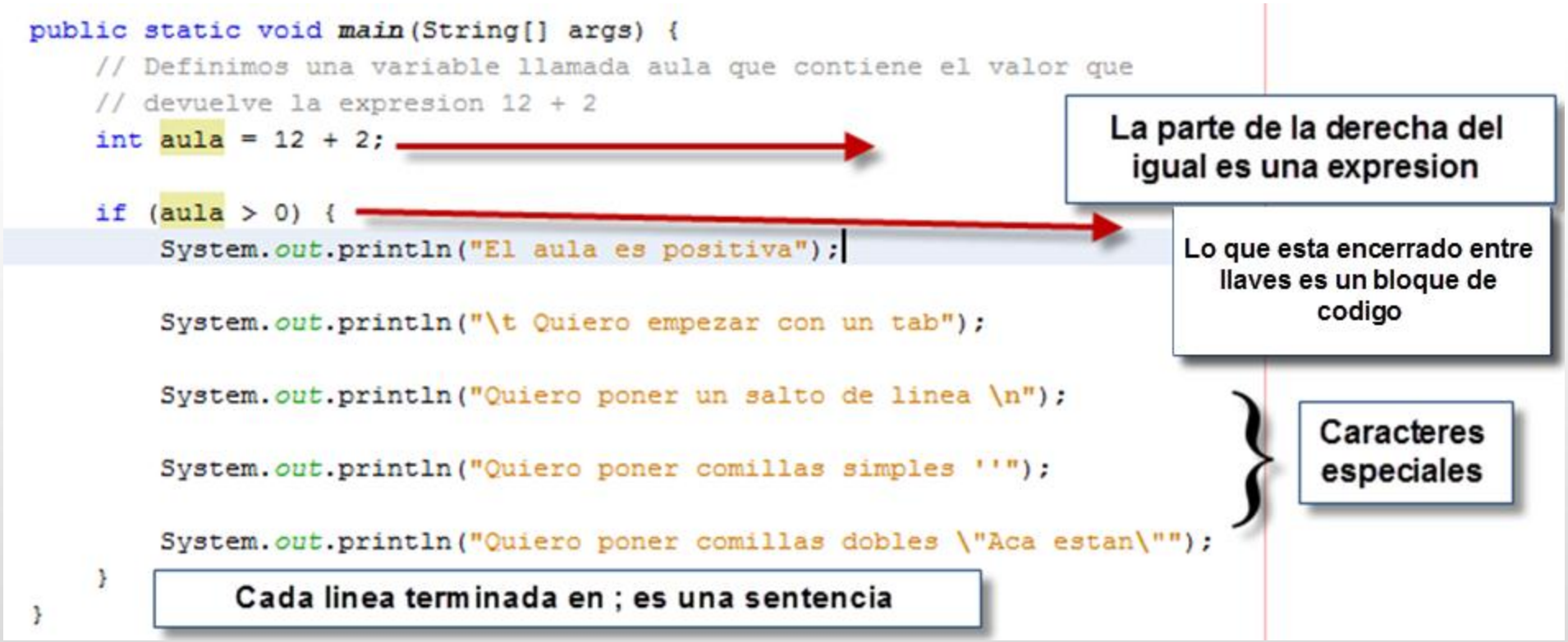
Los comentarios son de gran utilidad ya que nos dejan documentar lo que hace o quiere realizar nuestro código y si alguien mas agarra nuestro código lo puede entender de forma simple, los comentarios no son tenidos en cuenta por el compilador.

```
public static void main(String[] args) {
    // Este es un comentario de una sola linea.
    System.out.println("Hola Mundo 1");

    // Este es un comentario que continua en la linea de abajo
    // pero fijense que tuve que repetir la doble barra al principio.
    System.out.println("Hola Mundo 2");

    /*
     * Este es un comentario de multiples lineas el
     * cual no necesita repetir la doble barra solo necesita
     * estar encerrado entre la barra y la barra invertida
     */
    System.out.println("Hola Mundo 3");
}
```

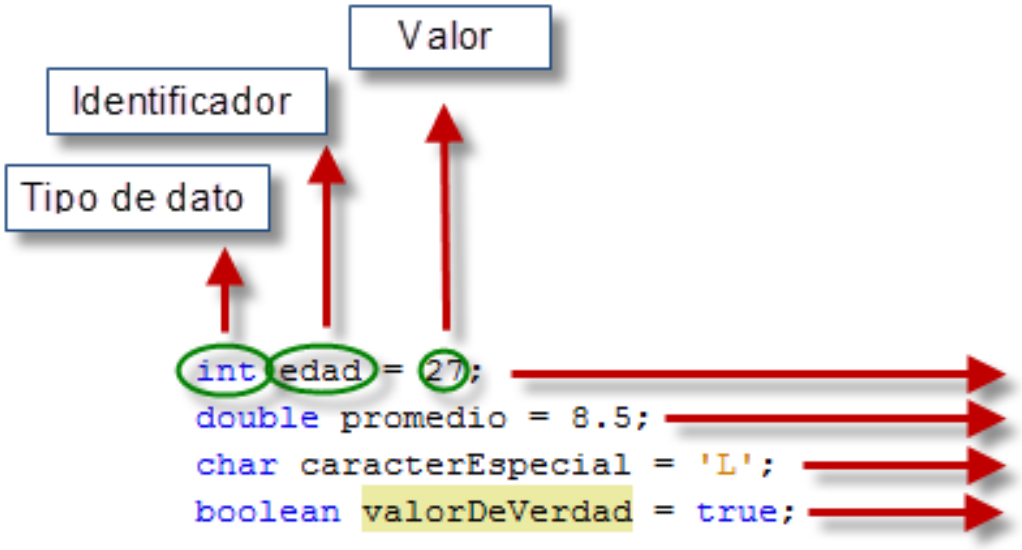
Sentencias / Bloques / Expresiones



Variables

Una variable(Contenedor de información) es la combinación de 3 cosas:

- Tipo de dato: Esto no se puede cambiar (Estático).
- Identificador: Es el nombre que identifica mi variable.
- Valor: Este parte puede variar su valor.



Memoria		Tipo dato
edad	27	Numero Entero
promedio	8.5	Numero Decimal
caracterEspecial	L	Carácter
valorDeVerdad	1	Booleano

char & String

```
public static void main(String[] args) {
    char caracterEspecial = 'L';

    // TODO code application logic here
    System.out.println("Hola Mundo");
}
```

- El tipo de dato char es utilizado para guardar un carácter y este puede ser cualquier carácter Unicode que nosotros querramos, siempre van encerrados entre comillas simples.
- El tipo de dato String es un objeto que representa una cadena de caracteres, en Java todos los objetos empiezan su nombre con mayúscula y siempre va entre comillas dobles.

Variables en Java

no puede comenzar con un número

puede comenzar con "_" o "$"

no puede utilizar caracteres "%" o "*" o "@" por que están reservados para otras operaciones

Puede incluir, pero no comenzar por un número.

No puede incluir el carácter espacio en blanco.

Distingue entre letras mayúsculas y minúsculas.

No se pueden utilizar las palabras reservadas como identificadores.

Palabras reservadas: Son todas las palabras propias del lenguaje y que no se pueden utilizar para declarar atributos, métodos, clases, etc. Son escritas en minúsculas:

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

* not used
** added in 1.2
*** added in 1.4
**** added in 5.0

Tipos de datos primitivos

Pueden ser de 8 tipos: **char**, **boolean**, **byte**, **short**, **int**, **long**, **double**, or **float**.

Los 6 tipos numéricos, están compuestos por bytes de 8 bits y pueden ser positivos o negativos. El primer bit es usado para representar el signo, donde un 1 significa negativo y un 0 significa positivo

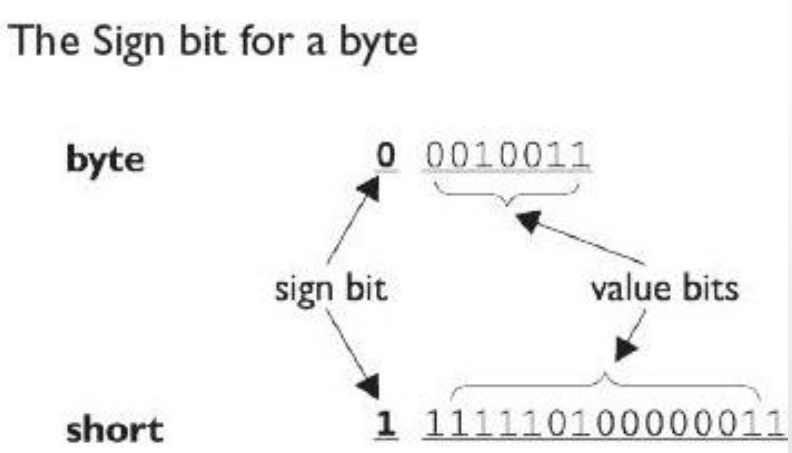
Type	Bits	Bytes	Minimum Range	Maximum Range
byte	8	1	-2^7	2^7-1
short	16	2	-2^15	2^15-1
char	16			
int	32	4	-2^31	2^31-1
long	64	8	-2^63	2^63-1
float	32	4	n/a	n/a
double	64	8	n/a	n/a

La cantidad de negativos es igual a la cantidad de positivos, el cero "0" es considerado como positivo, es por esa razón que se resta -1 en el rango máximo.

La formula para hallar el rango mínimo es $-2^{(bits\ -1)}$ y para el rango máximo es $2^{(bits\ -1)}$ -1. En ambos se resta **1 bits** que es el que representa el signo (positivo o negativo).

Los de tipo **boolean** no tienen rango, solo aceptan **true** o **false**.

Los de tipo **char** (1 caracter), es de 16 bits Unicode character. puede tener 2^{16} posibles valores, el rango es de 0 a 65535, osea es un poco mas grande que un **short**. Al ser de tipo char, no cuenta con signo de positivo o negativo, por tal razón no se le resta -1 a la potencia.



sign bit: 0 = positive
1 = negative

value bits:

byte: 7 bits can represent 2^7 or 128 different values:
0 thru 127 -or- –128 thru –1

short: 15 bits can represent 2^{15} or 32768 values:
0 thru 32767 -or- –32768 thru –1

Constantes en Java

Una constante(Contenedor de información) es la combinación de 3 cosas:

- Tipo de dato: Esto no se puede cambiar (Estático).
- Identificador: Es el nombre que identifica mi variable (Por convención se escriben en mayúscula).
- Valor: Se asigna el valor una única vez y este ya no puede cambiar durante la vida del programa.

```
final double PI = 3.131592653589;
```

Operadores

Un operador es un símbolo especial que le indica al compilador que debe efectuar una determinada operación.

- Operadores aritméticos.
- Operadores de asignación.
- Operador incremental/decremental.
- Operador de concatenación.
- Operadores lógicos.
- Operadores relacionales.
- Operadores binarios.

Los veremos en detalle junto con la estructura de control IF

Los operadores están definidos en tres categorías:

- Operadores Unarios: Solo tienen un único operando.
- Operadores Binarios: Tienen dos operando.
- Operadores Ternarios: Tienen tres operando (En Java existe uno solo de este tipo).

Operadores aritméticos

```
public static void main(String[] args) {
    // Defino dos variables del tipo entero, donde una contiene el valor 15
    // y la otra el valor 10.
    int numero1 = 15;
    int numero2 = 10;

    // Suma
    System.out.println(numero1 + numero2);

    // Resta
    System.out.println(numero1 - numero2);

    // Multiplicacion
    System.out.println(numero1 * numero2);

    // Division ¿Cuanto va a imprimir por pantalla?
    System.out.println(numero1 / numero2);

    // Resto de la division
    System.out.println(numero1 % numero2);
}
```

Operadores asignación I

```
public static void main(String[] args) {
    // Defino dos variables del tipo entero, donde una contiene el valor 15
    // y la otra el valor 10.
    int numero1 = 15;
    int numero2 = 10;

    // Defino una variable que se llama resultado y la inicio en 0
    int resultado = 0;

    // Asigno el resultado de la suma para guardarlo en esta variable
    resultado = numero1 + numero2;

    // Como se puede ver las variables numero1 y numero2 mantiene su valor
    // original, si yo no necesitaria los valores originales y solo el resultado
    // de la suma podriamos ejecutar la siguiente sentencia.
    // numero1 += numero2;

    // Imprimo el valor de la variable llamada numero1
    System.out.println(numero1);

    // Imprimo el valor de la variable llamada numero2
    System.out.println(numero2);

    // Imprimo el valor de la variable llamada resultado
    System.out.println(resultado);
}
```

Operadores asignación II

Como se vio en el slide anterior el operador de asignación "+=" nos brinda un camino reducido para hacer la suma y asignar el resultado a una de las variables que estamos sumando, lo mismo se puede hacer para lo demás operadores aritméticos y para otros operadores lógicos.

Operadores de asignación

Operación	Operador	Utilización	Operación equivalente
Suma	+=	A += B	A = A + B
Resta	-=	A -= B	A = A - B
Multiplicación	*=	A *= B	A = A * B
División	/=	A /= B	A = A / B
Resto de división	%=	A %= B	A = A % B
Desplazamiento a la izquierda	<<=	A <<= B	A = A << B
Desplazamiento a la derecha	>>=	A >>= B	A = A >> B
Desplazamiento a la derecha sin signo	>>>=	A >>>= B	A = A >>> B
AND de bits	&=	A &= B	A = A & B
OR de bits	|=	A |= B	A = A | B
XOR de bits	^=	A ^= B	A = A ^ B

Operador incremental/decremental

```java
public static void main(String[] args) {
    // Defino una variable del tipo numero entero la cual se llama "numeroInicial"
    // y la inicio con el valor 1.
    int numeroInicial = 1;

    // Imprimo el valor de la variable llamada "numeroInicial"
    System.out.println(numeroInicial);

    // Le sumo 1 usando el operador suma (+)
    numeroInicial = numeroInicial + 1;

    // Imprimo el valor de la variable llamada "numeroInicial"
    System.out.println(numeroInicial);

    // Uso el operador incremental (Esto incrementa en 1 el valor)
    numeroInicial++;

    // Imprimo el valor de la variable llamada "numeroInicial"
    System.out.println(numeroInicial);

    // Uso el operador para decrementar (Esto decrementa en 1 el valor)
    numeroInicial--;

    // Imprimo el valor de la variable llamada "numeroInicial"
    System.out.println(numeroInicial);
}
```

Operador de concatenación

```
public static void main(String[] args) {
    // Defino una variable llamada numeroFavorito y la inicio con 11
    int numeroFavorito = 11;

    // Imprimo el valor de la variable llamada "numeroFavorito"
    System.out.println(numeroFavorito);

    // Imprimo el resultado de sumar mi numero favorito con si mismo
    System.out.println(numeroFavorito + numeroFavorito);

    // Imprimo un cartel + el valor de la variable llamada "numeroFavorito"
    System.out.println("Mi numero favorito es: " + numeroFavorito);
}
```

El operador de concatenación es el mismo que la suma, y el resultado varia en base al tipo de los operando (Cuando un operador tiene dos funciones se dice que esta sobrecargado).

Numero + Numero = Resultado de la suma.

Caracteres + Numero = Concatenación entre los caracteres y el Numero.

Otro tipo de dato + Cualquier cosa = Nos marca error.

Operadores lógicos I

Operadores Lógicos: Nos permiten construir expresiones lógicas.

- **&&:** Devuelve true si ambas expresiones son verdaderas.
- **||:** Devuelve true si al menos alguna de las expresiones es verdadera.
- **!:** Niega la expresión que se le pasa.
- **^:** Devuelve true cuando una expresión es verdad y la otra falsa.
- **&:** Devuelve true si ambas expresiones son verdaderas, evalúa ambas.
- **|:** Devuelve true si al menos alguna de las expresiones es verdadera, evalúa ambas.

Expresion1	Expresion2	Expression1 && Expression2	Expression1 || Expression2	!Expression1	Expression1 ^ Expression2	Expression1 & Expression2	Expression1 | Expression2
False	False	False	False	True	False	False	False
False	True	False	True	True	True	False	True
True	False	False	True	False	True	False	True
True	True	True	True	False	False	True	True

Operadores lógicos II

```java
public static void main(String[] args) {
    // Defino una variable llamada esAlto y la inicio en verdad
    boolean esAlto = true;

    // Pregunto, si la variable llamada esAlto tiene valor de verdad
    // si es verdad imprimo el mensaje
    if (esAlto) {
        System.out.println("Es alto 1");
    }

    // Esta linea hace exactamente lo mismo que la de arriba pero pone la
    // comparacion de forma explica.
    // Un solo igual es el operador asignacion, pero dos iguales seguidos es
    // el de comparacion.
    if (esAlto == true) {
        System.out.println("Es alto 2");
    }

    // Aca preguntamos si el valor de esAlto es false
    if (esAlto == false) {
        System.out.println("Es alto 3");
    }

    // Defino una variable llamada ganaMasQueYo y la inicio en verdad
    boolean ganaMasQueYo = true;

    // Defino una variable llamada sabeMasQueYo y la inicio en verdad
    boolean sabeMasQueYo = false;

    if (ganaMasQueYo == true && sabeMasQueYo == false) {
        System.out.println("Es mi Jefe");
    }
}
```

Operadores relacionales

Equality and Relational Operators

==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

```java
public static void main(String[] args) {
    // Defino una variable que se llama "numero" y la inicializo
    // con el numero 10
    int numero = 10;

    // Pregunto si el numero ES IGUAL a 10
    if (numero == 10) {
        System.out.println("El numero es 10");
    }

    // Pregunto si el numero NO ES IGUAL a 10
    if (numero != 10) {
        System.out.println("El numero no es igual 10");
    }

    // Pregunto si el numero ES MAYOR que 10
    if (numero > 10) {
        System.out.println("El numero es mayor que 10");
    }

    // Pregunto si el numero ES MAYOR O IGUAL que 10
    if (numero >= 10) {
        System.out.println("El numero es mayor o igual que 10");
    }

    // Pregunto si el numero ES MENOR que 10
    if (numero < 10) {
        System.out.println("El numero es menor que 10");
    }

    // Pregunto si el numero ES MENOR O IGUAL que 10
    if (numero <= 10) {
        System.out.println("El numero es menor o igual que 10");
    }
}
```

Operadores binarios

```
public static void main(String[] args) {
    // Definimos una variable del tipo byte que contiene al 8
    byte numero = 8;

    // Numero 8. Formato binario: 00001000
    System.out.println(numero);

    // Complemento de 8 (Da por resultado -9). Formato binario: 11110111
    System.out.println(~numero);

    // Signed left shift de 8 (Da por resultado 16). Formato binario: 00010000
    System.out.println(numero<<1);

    // Signed right shift de 8 (Da por resultado 4). Formato binario: 00000100
    System.out.println(numero>>1);
}
```

Bitwise and Bit Shift Operators

~	Unary bitwise complement
<<	Signed left shift
>>	Signed right shift
>>>	Unsigned right shift
&	Bitwise AND
^	Bitwise exclusive OR
|	Bitwise inclusive OR