Tema 1
Fundamentos de Java

Prof. José de la Torre López
(adaptación Antonio Hernández)

1º Desarrollo de Aplicaciones Multiplataforma
Módulo formativo - Programación

¿Por dónde empezamos?

Algoritmo

"Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema."

Algoritmo - Ejemplo práctico

Hay una bombilla fundida en vuestro cuarto. Escribid una **secuencia de pasos ordenada** en los que se detalle **QUÉ** hay que hacer desde que descubrís la bombilla fundida hasta que se cambie la bombilla.

No siempre hay bombillas de repuesto en casa…

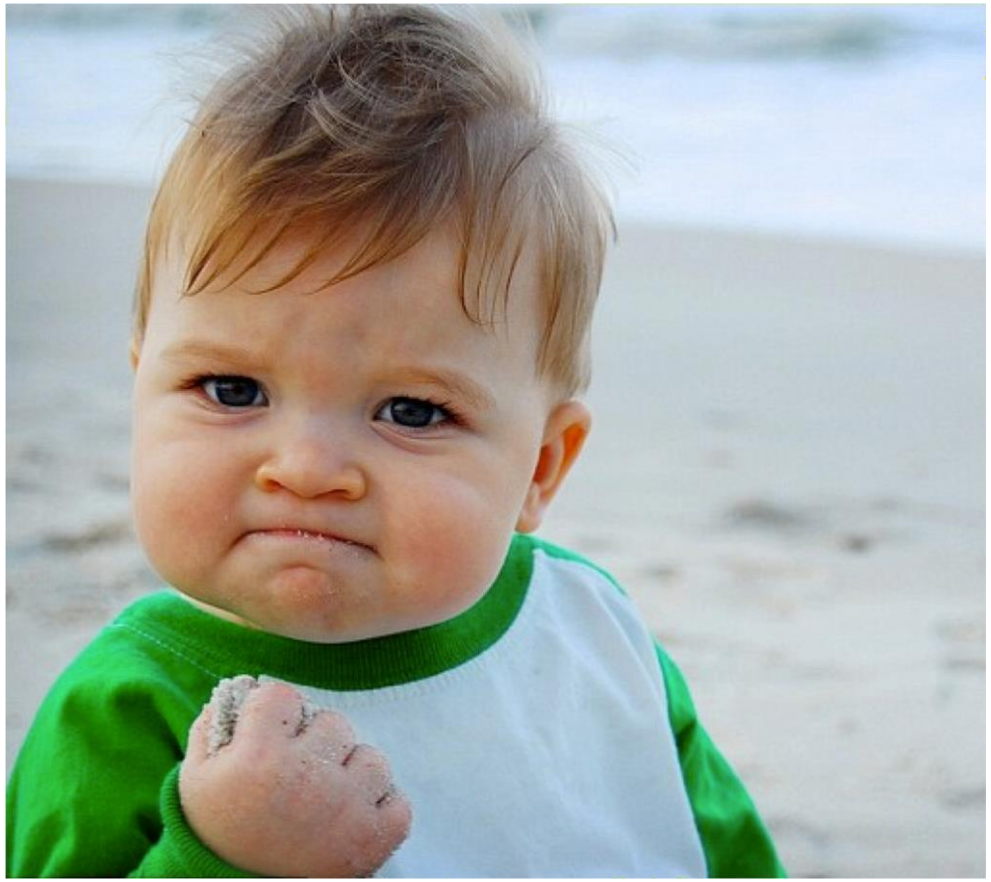
Algoritmo - Ejemplo práctico

1. Veo la bombilla fundida.
2. Le doy 3 veces al interruptor con fe ciega de que se ha ido la luz.
3. No se ha ido la luz… llamo a mi compi de piso con un grito.
4. Observo la bombilla como si fuese a cambiarse sola en plena oscuridad.

Algoritmo - Ejemplo práctico

5. Voy al cajón de las bombillas.
➔ ¿Hay bombillas de repuesto en casa?
➔ ¡Sí!
6a. La cambio sin electrocutarme.
➔ No…
6b. Voy a comprar al chino la más barata.
6. La cambio sin electrocutarme.

Enhorabuena, acabáis de iniciaros en el mágico mundo de la programación



Programa

Vulgarmente, es uno o varios **algoritmos** que están codificados en un lenguaje de programación de manera que el ordenador realice una o más tareas programadas.

Lenguajes de programación

1ª generación	Código máquina 01101010 10101110…
2ª generación	Ensamblador

Venga va, un poco de ensamblador

```
DATOS SEGMENT
saludo db "Hola mundo!!!","$"
DATOS ENDS CODE SEGMENT
assume cs:code,ds:datos START PROC
mov ax,datos
mov ds,ax
mov dx,offset saludo mov ah,9
int 21h
mov ax,4C00h
int 21h
START ENDP CODE ENDS
END START
```

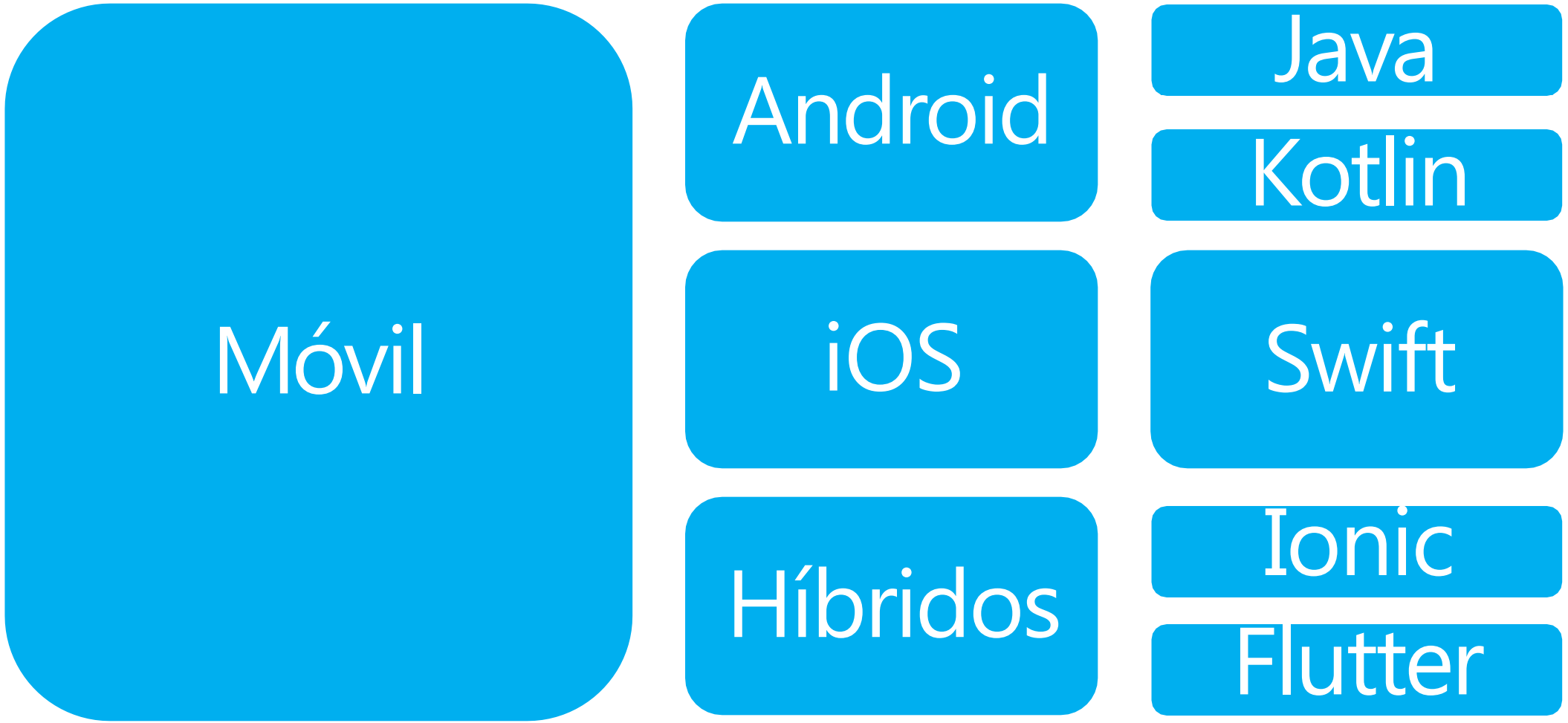
Lenguajes de programación

3ª generación	Fortran Lisp Pascal
4ª generación	POO

Lenguajes de programación

Web cliente	HTML CSS JavaScript TypeScript
Web servidor	Java C# PHP NodeJS Python

Lenguajes de programación



¿Son todos iguales?

Paciencia

La clave está en comprender la base de un solo lenguaje.

Podrá cambiar el paradigma, la sintáxis, la estructura, etc. Pero vuestra cabeza los aprenderá rápidamente.

Clasificación de lenguajes por tipo

Declarativos	VB, C#, F#
	Java, C++
	C
Imperativos	SQL
Funcionales	Swift, JS, LISP
Lógicos	Prolog

Compilado VS Interpretado

Según el lenguaje de programación será compilado o interpretado

Investigad qué diferencia hay entre ambos.

¿Y transpilado?

Nota para tu YO futuro

Todo lo que a aquí se menciona y parece chino lo entenderé durante la evolución del curso. De hecho, al final de curso sabré realmente cual es la diferencia entre todos esos lenguajes.

La caja negra

Algunas reglas de la programación

1. Entender bien lo que se pide
2. Comprender cómo comienza el programa
3. Contemplar todos los tipos que de entrada que pueda haber en mi caja negra (el usuario puede no saber como usar el programa)
4. ¿Qué pasos tengo que seguir para que con esa entrada le de a mi usuario su resultado deseado?
5. ¿Cómo y cuándo voy a darle su resultado?

Historia de Java

- Creado en 1995
- Fuerte influencia de C y C++
- Creado por Sun Microsystems con la idea de crear aplicaciones que pudieran ejecutarse en cualquier dispositivo.
- Originalmente se llamaba Oak

¿Y por qué Java?

¿Y por qué no?

Es probablemente un lenguaje que uno vez dominas los conceptos básicos es muy fácil pasarse a cualquier otro lenguaje de alto nivel.

JVM, JRE y JDK

Java Virtual Machine

Java Runtime Environment

Java Development Kit

¿Cuál es la diferencia?

Versiones de Java

- 1.0 – 1996
- 1.1 – 1997
- 1.2 o también llamada Java 2 – 1998
- 1.3 o Java 3 – 2000
- 1.4 o Java 4 – 2002
- 1.5 o Java 5 – 2004
- 1.6 o Java 6 – 2006
- 1.7 o Java 7 – 2011
- **1.8 o Java 8 – 2014 – Esta fue gorda**

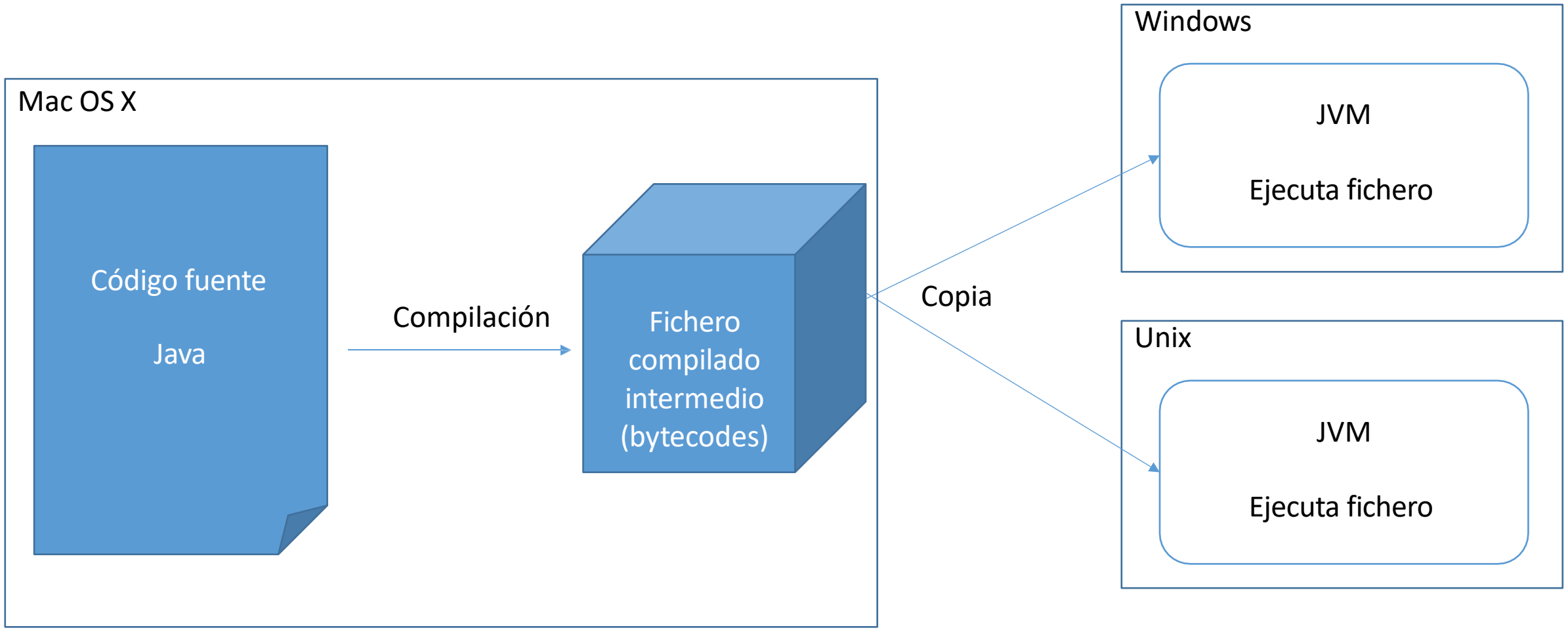
…

Versión actual

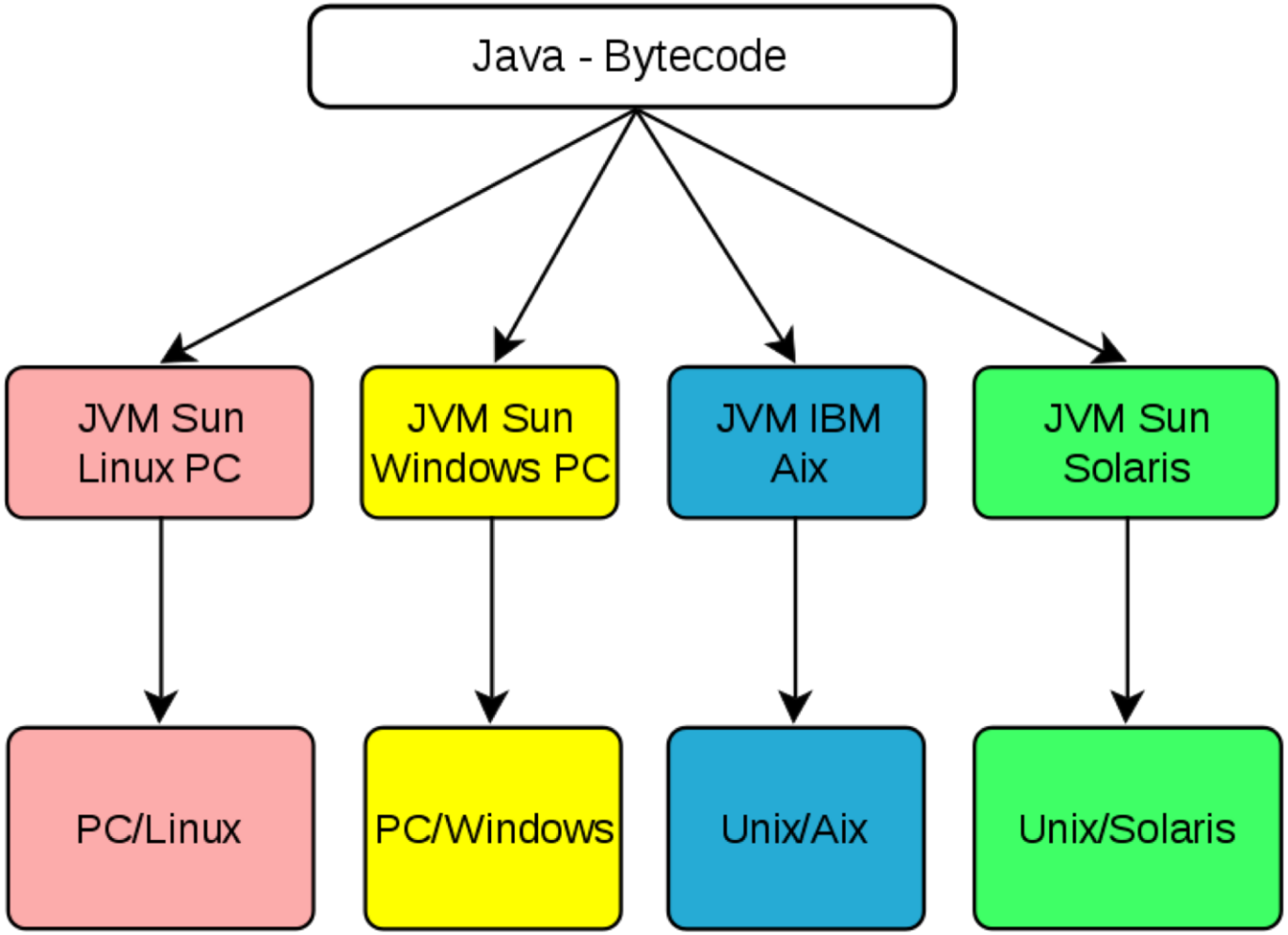
Java 17

En el momento de crear este tema su fecha era el 21de septiembre de 2021.

Compilación en Java (ejemplo)



Bytecode


Java - Bytecode
JVM Sun Linux PC
JVM Sun Windows PC
JVM IBM Aix
JVM Sun Solaris
PC/Linux
PC/Windows
Unix/Aix
Unix/Solaris

¿Qué necesitáis?

Tener configurado un entorno de desarrollo (buscar que es un IDE en internet)

1. Un IDE online
2. Eclipse

Un IDE online, por ejemplo:
https://www.onlinegdb.com/online_java_compiler

Listos para el despegue

Hagamos una prueba todos.

¿Qué necesitáis para Eclipse?

1. Tener instalado/actualizado la última versión de JDK (en este caso Eclipse lo instala por nosotros)
2. Descargar Eclipse de su página oficial (última versión estable)

¿Cómo? San Google

Eclipse IDE for Java EE Developers

Es ese, no otro. Podría ser otro, pero no, es ese. Puede que os gusten los iconos de otras versiones, pero no, hay que descargar ese. Luego no quiero que me digáis: profe, no va…

¿Qué es Eclipse?

Es un IDE (**entorno de desarrollo** integrado) para programar en muchos lenguajes (Java, C, C++, Python, Ruby, etc.).

Entre nosotros:
Banco de trabajo configurable donde más o menos todo es configurable para que el desarrollador lo tenga todo a mano.

Es bastante liviano, plug&play, tiene cosas que no sabréis que son ni cuando llevéis 10 años programando pero algún día diréis: "¡¡¡Para eso era!!!"

Java y la consola

Aunque ahora os de igual, debéis saber que Java compila y ejecuta desde consola (cmd o terminal). Para que funcione correctamente (que suele hacerlo por defecto) las variables de entorno tienen que apuntar al JRE o JDK que queráis.

> javac holamundo.java ➔ Esto compila en consola

> java holamundo ➔ Esto ejecuta en la JVM en consola

Volvamos a la realidad

Notas previas:

- Es case sensitive
- Toda instrucción debe acabar con ; (ya veremos casos especiales)
- Una instrucción puede tener intros, tabs, espacios y seguirá funcionando igualmente
- Los bloques de código se delimitan con { }

Java

A continuación veremos los comandos, instrucciones, operadores y conceptos más básicos de Java.

No perdáis detalle

Comentarios

El código comentado es gloria.

Dentro de todas las instrucciones, si hay aclaraciones que evitan tener que descifrar el código es muy agradecido.

// comentarios de una sola línea

/* comentarios
en
Varias líneas
*/

import

Los imports aparecen SIEMPRE al principio del fichero, si es que hay alguno. Java implícitamente hace el siguiente import por nosotros:

```
import java.lang;
```

El import sirve para utilizar clases (“cosas”) dentro de tu programa. Por ejemplo: para trabajar con fechas.

import

Los imports, gracias a Eclipse, se suelen realizar automáticamente conforme vamos programando. Hay un atajo de teclado que ya os contaré.

Con esta forma:

```
import paquete.subpaquete.[…].subpaquete.clase;
```

import

Hace los imports dentro de este paquete preconfigurados. Normalmente importa todas las clases de ese paquete.

```
import java.util.*; //Importa todo dentro de util
```

y

```
import java.util.Date;  //Importa solo Date
```

Variables

Son contenedores o cajas que sirven para almacenar datos durante la ejecución de un programa.

Tienen un tipo y un identificador (nombre).

No se recomiendan variables con identificadores con ñ o símbolos especiales como tildes.

Variables

Declaración:

```
tipo identificador;
```

Se les puede dar un valor al crearlas con el operador =

Se pueden crear tantas variables en línea como se desee siempre y cuando sean del mismo tipo.

Variables

Ejemplos:

```
int edad;
char inicialNombre;
int dias = 365;
int mes = 10, anyo = 2017;
float nota = 9.75;
boolean tengoHambre = true;
```

Variables

¡Pregunta!
¿Qué vale "c" al final del programa?

```
int a = 5, b = -7;
int c = a + b;
```

Tipos primitivos

Tipo	Bytes que ocupa	Rango de valores	Valor por defecto
boolean	2	true o false	false
byte	1	-128 a 127	0
short	2	-32768 a 32767	0
int	4	-2147483648 a 2147483649	0
long	8	$-9 \cdot 10^{18}$ a $9 \cdot 10^{18}$	0L
float	4	$-3.4 \cdot 10^{38}$ a $3.4 \cdot 10^{38}$	0.0f
double	8	$-1.79 \cdot 10^{308}$ a $1.79 \cdot 10^{308}$	0.0d
char	2	Caracteres de Unicode	'\u0000'

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Variables

```
int a = 12;
byte b = a;
```

Error en compilación por posible pérdida de precisión

```
int a = 12;
byte b = (byte) a;
```

Esto es un casting o conversión explícita

Variables

```
int a = 9.5;
```

Error en compilación

```
int a = (int) 9.5;
float b = a;

//¿Valor de b?
```

Variables

Caracteres

```
char  letra;
letra = ‘C’;
letra = 67;
```

¡Es lo mismo!

Ejercicio

Probad en un nuevo programa, todos los tipos de variables que existen sin asignarle un valor e imprimid por pantalla esas variables.

Ahora dadles un valor inicial a todas las variables y volved a imprimirlas por pantalla.

Ejercicio

Probad a hacer castings entre distintos tipos a ver qué os sale en cada uno de ellos.

Asignad a un char el valor de un int e imprimid por pantalla su valor.

Caracteres especiales

Caracter	Significado
\b	Retroceso
\t	Tabulador
\n	Nueva línea
\f	Alimentación de página
\r	Retorno de carro
\’	Comilla simple
\”	Comillas dobles
\\	Barra invertida

Ejercicio

Cread un texto en el que mezcléis esos caracteres especiales a lo largo del mismo y observad que efecto tienen los caracteres especiales.

Ciclo de vida y ámbito

1. Una variable se declara
2. Se le da un valor
3. Se utiliza en su ámbito, normalmente el ámbito lo delimitan los { }
4. Cuando se acaba el ámbito la variable es basura y no es accesible

Ámbito de las variables

```
{
   int anyos = 18;
}
anyos = 19; // ERROR, no existe ya.
```

```
int anyos = 18;
{
   anyos = 19; //Sí funciona pues tiene mismo ámbito
}
```

Ámbito de las variables

```
int anyos = 18;
{
   int anyos = 19; //ERROR, ya existe en ese ámbito
}
```

```
int anyos = 18;
{
   anyos = 19; //Sí funciona pues tiene mismo ámbito
}
```

Garbage collector

Se encarga de recoger toda la basura que hay y desecharla para liberar espacio en memoria.

AVISO

Las cadenas de texto o String no son tipos primitivos de Java.

Son **OBJETOS** de la **CLASE** String.

Repito, un String no es un tipo de dato primitivo en Java.

Operadores

Son los agentes de cambio de las variables.

Aritméticos
Lógicos
A nivel de bit
Asignación
Ternario

Operadores aritméticos

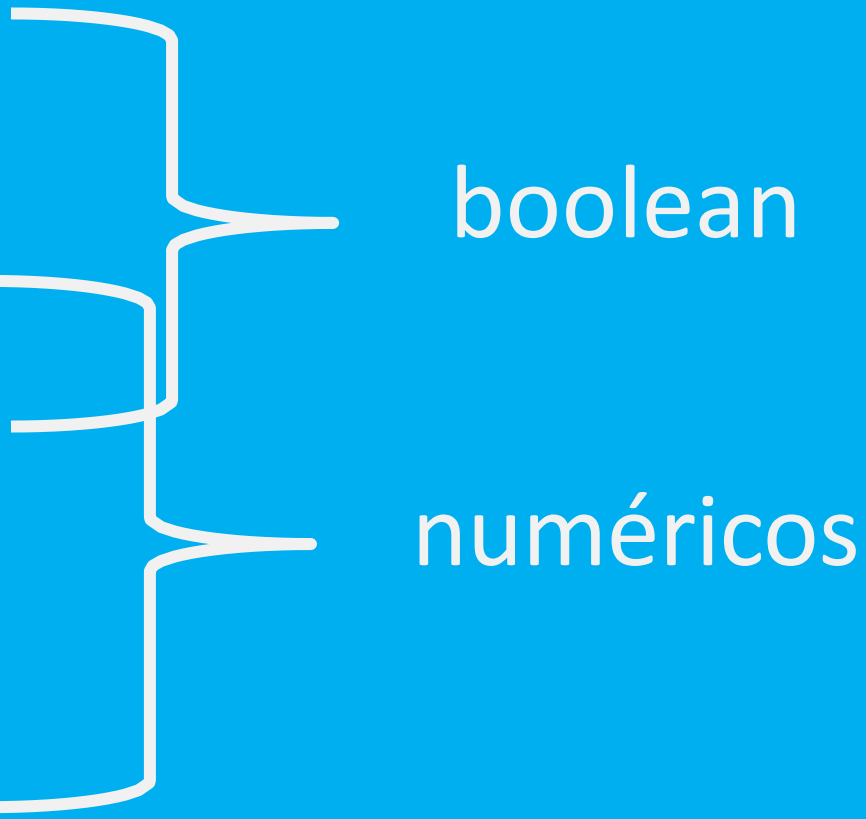
+ Suma
- Resta
* Multiplicación
/ División
% Módulo (Resto de la división)

Ejercicio:

```
int a = 6, b = 3;
    c = a + b;
    c = a – b;
    c = a * b;
    c = a / b;
    c = a % b;
```

Operadores lógicos

&&	Y lógico
||	O lógico
!	NO lógico
==	Igual que
!=	Distinto de
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que



Tablas de la verdad

&&

a	b	Resultado
true	true	true
true	false	false
false	true	false
false	false	false

||

a	b	Resultado
true	true	true
true	false	true
false	true	true
false	false	false

!

a	Resultado
true	false
false	true

Ejercicio

Probad los operadores lógicos de las tablas de verdad en un programa Java con dos variables booleanas y comprobad que no me he inventado las tablas de verdad.

Ejemplos lógicos

```
boolean mayorDeEdad, menorDeEdad;
int edad = 21;

mayorDeEdad = edad >= 18;
//mayorDeEdad será true

menorDeEdad = !mayorDeEdad;
//menorDeEdad será false
```

Ejemplos lógicos

```
boolean carnetConducir = true;
int edad = 20;

boolean puedeConducir = (edad>=18) &&
carnetConducir;

//Si la edad es de al menos 18 años y carnetConducir
es
//true, puedeConducir es true
```

Ejemplos lógicos

```
boolean nieva = true, llueve = false, graniza
= false; boolean malTiempo = nieva || llueve
||  graniza;

//¿Hace mal tiempo?
```

Operadores a nivel de bit

No los vamos a usar en un 99%. Son más para lenguajes de bajo nivel como C. Además, requiere conocimientos profundos de lógica y sistemas digitales para dominarlos.

Tenemos:

& | ! ^ >> << <<< >>>

Operadores de asignación

a = b;

Operadores de asignación

El más típico es el =

Pero hay otros interesantes:

`+=` `-=` `*=` `/=` `%=` `&=` `|=` `^=`

Ejemplo:

```
int a = 2;
a += 3; // es lo mismo que: a = a + 3;
```

Operadores de asignación

Se pueden concatenar asignaciones:

```
a = b = c = 3;
```

Aunque no es recomendable.

Operadores de asignación

Incrementos y decrementos

```
int x = 2;
x++;            // x = x + 1;
x--;            // x = x - 1;
```

Operadores de asignación

Pre-incremento y post-incremento

++x	x++
Primero incremento luego asigno	Primero asigno luego incremento

Operadores de asignación

Pre-incremento y post-incremento

```
int a = 2;
int b = a++;
System.out.println(a); // 3
System.out.println(b); // 2

a = 2;
b = ++a;
System.out.println(a); // 3
System.out.println(b); // 3
```

Operadores de asignación

Operador ternario (? :)

```
variable = expresionLogica ? valorSiVerdadero : valorSiFalso;
```

Ejemplo:

```
int paga= (edad>18) ? 6000 : 3000;
```

Precedencia de operadores

Level	Operator	Description	Associativity
16	[] . ()	access array element access object member parentheses	left to right
15	++ --	unary post-increment unary post-decrement	not associative
14	++ -- + - ! ~	unary pre-increment unary pre-decrement unary plus unary minus unary logical NOT unary bitwise NOT	right to left
13	() new	cast object creation	right to left
12	* / %	multiplicative	left to right
11	+ - +	additive string concatenation	left to right
10	<< >> >>>	shift	left to right
9	< <= > >= instanceof	relational	not associative
8	== !=	equality	left to right

Level	Operator	Description	Associativity
7	&	bitwise AND	left to right
6	^	bitwise XOR	left to right
5	|	bitwise OR	left to right
4	&&	logical AND	left to right
3	||	logical OR	left to right
2	?:	ternary	right to left
1	= += -= *= /= %= &= ^= |=	assignment	right to left

¿Es importante saberlo?

```
boolean a, b, c;

a = b && c || a;
a = b && (c || a);
```

```
int x, y, z;

x = y * x / z;
x = y * (x / z);
```

Más fácil, ¿cuánto vale x?

```
int x;
x = 9 + 3 * 4 / 2;

¿12? ¿15? ¿24?
```

Truco de orden de precedencia

Usad los paréntesis para aritmética o para lógica.

No fallan.

Ejercicio

Calculad y después probad en un programa Java a ver si estabais en lo cierto:

```
System.out.println( 3 + 3 * 2 );
System.out.println( 3 * 3 - 2 );
System.out.println( 3 * 3 / 2 );
System.out.println( 1 * 1 + 1 * 1);
System.out.println( 1 + 1 / 1 - 1);
System.out.println( 3 * 3 / 2 + 2);
System.out.println( x++ + x++ * --x );
```

Constantes

La palabra final hace que una vez asignado un valor a una variable sea imposible volver a cambiarlo.

Ejemplo:

```
final int MAX = 20;
```

Salida por pantalla

Usaremos:

```
System.out.println();
```

o

```
System.out.print();
```

Lectura por teclado

AVISO

Leer por teclado en Java es algo traicionero para principiantes. Es más complejo de lo que puede parecer.

Lectura por teclado

```
Scanner sc = new Scanner(System.in);
// entrada de una cadena
String name = sc.nextLine();
// entrada de un carácter
char gender = sc.next().charAt(0);
// Entrada de datos numéricos
int age = sc.nextInt();
long mobileNo = sc.nextLong();
double average = sc.nextDouble();
```

Lectura por teclado

Ojo, aún no hemos aprendido a validar, pero cuando lo hagamos será obligatorio.

Clase Math

Para operadores matemáticos complejos (potencia, raíz cuadrada, redondeo, trigonometría, valor absoluto, constantes) usaremos la clase Math.

```
float resultado = Math.pow(2,3); //2³
```

Clase Math. Ejercicio.

Buscad en la documentación de esa clase qué métodos podéis utilizar.

Números aleatorios

Hay muchas formas de generar números aleatorios, de hecho es muy difícil crear un buen generador de números aleatorios. Se paga mucho por ellos.

Números aleatorios

```
Math.random();
```

Devuelve un double entre 0 y 1
[0,1) = [0.00, 0.9999...]
(0 incluido pero no el 1)

Números aleatorios. Ejercicio.

```
Math.random()*9+1;
(int) Math.floor(Math.random()*10+1);
(int) Math.floor(Math.random()*21+10);
```

Enhorabuena, ya sois programadores juniors

Tema 1
Fundamentos de Java

Prof. José de la Torre López
(adaptación Antonio Hernández)

1º Desarrollo de Aplicaciones Multiplataforma
Módulo formativo - Programación