

# El lenguaje Java

## Java

- Lenguaje interpretado
- Fuertemente tipado
- Sintaxis similar a C / C++
- Sin punteros: garbage collection
- 100% portable
- Integra librerías estándar para:
  - Interfaces de usuario
  - Objetos distribuidos
  - Threads
  - XML
  - Web, Móviles, TV, muchas más...
- Ejecutable desde navegadores web (Mozilla, MS Explorer, etc.)
- Origen: aumentar html para páginas web más dinámicas
- Creado por Sun Microsystems (actualmente Oracle, adquisición Sun en 2010)

## Versiones de Java

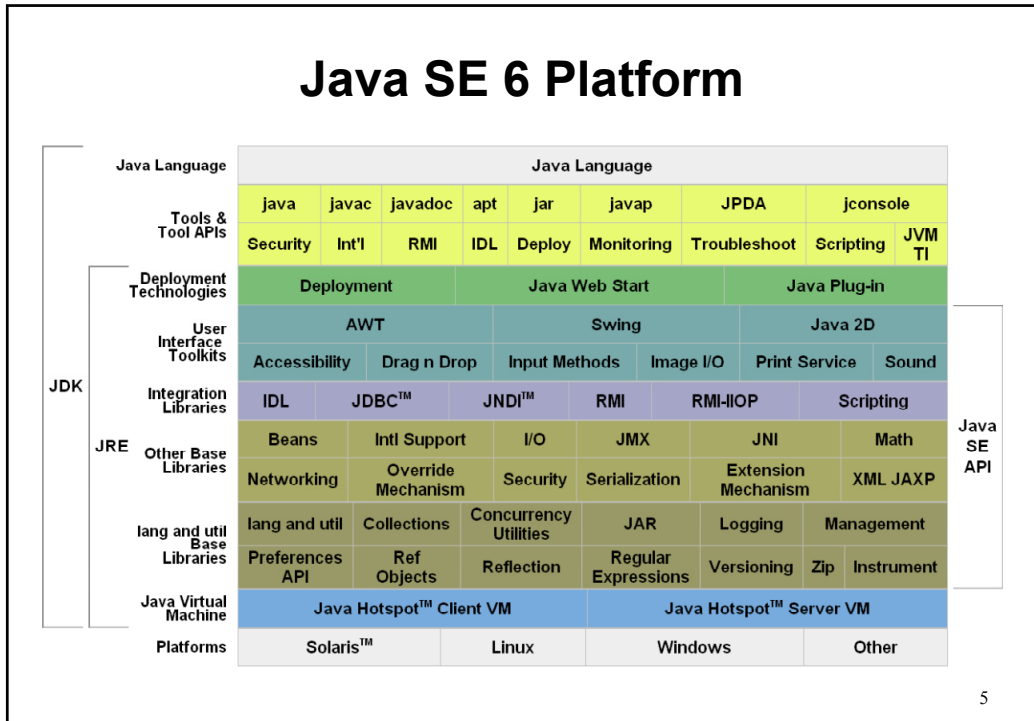
- Java 1.0 – enero 1996
  - Unos cientos de clases
- Java 1.1 – febrero 1997
  - JavaBeans, JDBC, RMI...
- J2SE 1.2 – diciembre 1998
  - Swing, reflexión, Java IDL, Collections...
- J2SE 1.3 – mayo 2000
  - Compatibilidad RMI/CORBA...
- J2SE 1.4 – febrero 2002
  - XML, JWS, mejoras internas...
- J2SE 5.0 (a.k.a. 1.5) – septiembre 2004
  - Generics, Enumerations, enhanced "for"...
- Java SE 6 (a.k.a. 1.6) – diciembre 2006
  - Total de 3.000+ clases, mejoras fuertes de rendimiento...
- Java SE 7... ?

3

## Java Development Kit (JDK)

- Java Runtime Environment (JRE)
  - Java Virtual Machine (JVM)
  - Java API: lenguaje básico + librería estándar de clases
- Compilador a *bytecode*
- Otros compiladores y herramientas específicas: debuggers, applet viewer, generadores de documentación, compiladores RMI, etc.

4



## Java vs. C++

<u>Java</u>	<u>C++</u>
Interpretado	Compilado: más eficiente
Totalmente portable	Aspectos no portables
Memoria dinámica Garbage collection No existen punteros	Memoria automática gestionada por el RTS Memoria dinámica gestionada por el programador
Todos los métodos virtuales	Métodos virtuales y no virtuales
No tiene herencia múltiple	Permite herencia múltiple
Información completa sobre las clases en tiempo de ejecución	Muy limitada (dynamic_cast)
Tratamiento genérico de tipos (wrappers)	Punteros a void

6

<b><u>Java</u></b>	<b><u>C++</u></b>
Soporte estándar para interfaces de usuario: Swing	No incluido en el lenguaje: Microsoft MFC, etc.
Soporte estándar para concurrencia	No incluido en el lenguaje
Soporte estándar para objetos distribuidos: RMI	No incluido en el lenguaje: CORBA, ActiveX, etc.
Ejecutable en web browsers (applets)	
Homogeneidad del lenguaje	Variedad de sintaxis y casos excepcionales
Mejoras en diseño y sintaxis	Compatible con C

7

# Elementos del lenguaje

## Ejemplo

```
// package...
// import...

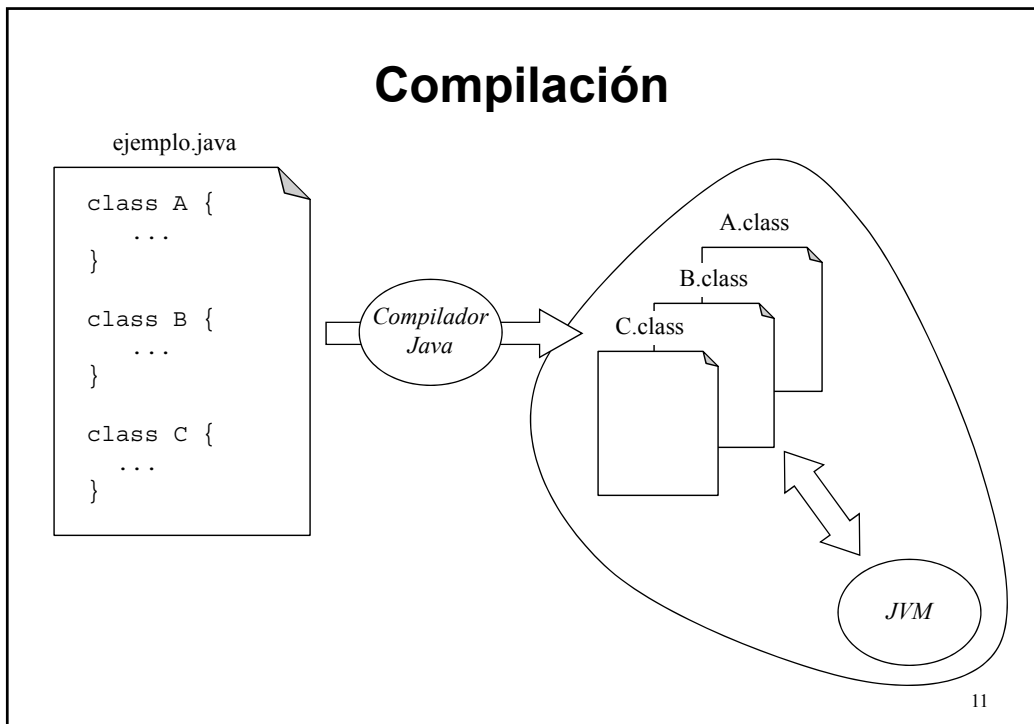
class Persona {
    String nombre;
    int edad;
    void mostrarDatos () {
        System.out.println ("Nombre: " + nombre + "\nEdad: " + edad);
    }
}

class MainClase {
    static void main (String args[]) {
        Persona x = new Persona ();
        x.nombre = "Juan Perez";
        x.edad = 25;
        x.mostrarDatos ();
    }
}
```

- Definición de clases
- Convención identificadores
- Método main
- Declaración y creación de objetos
- Acceso a miembros de objetos
- Strings
- Concatenación de strings
- Packages
- Ambito de las variables
- Variables y métodos de instancia vs. de clase

```
class Persona {
    String nombre;
    int edad;
    void mostrarDatos () {
        System.out.println ("Nombre: " + nombre + "\nEdad: " + edad);
    }
    Persona (String str, int n) {
        nombre = str;
        edad = n;
    }
}

class MainClase {
    static void main (String args[]) {
        Persona x = new Persona ("Juan Perez", 25);
        x.mostrarDatos ();
    }
}
```



## Aspectos léxicos

- Espacios en blanco
- Sentencias separadas por ';' ;'
- Comentarios
  - // hasta final de línea
  - /\* \*/ más de una línea
- Sensible al caso
- Convenios de nomenclatura

12

## Valores y variables

### Tipos básicos

byte 1 byte  
char 2 bytes (sin signo, caracteres Unicode, incluyen los ASCII)  
short 2 bytes  
int 4 bytes  
long 8 bytes  
float 4 bytes  
double 8 bytes  
boolean true ó false, no compatible con tipos numéricos

### Variables y literales

Declaración, utilización, ámbito léxico, etc. similar a C

13

## Compatibilidad entre tipos numéricos

```
byte b = 42;  
char c = 'a'; // también válido: c = 97;  
short s = 1024;  
int i = 50000;  
long l = 120000;  
float f = 5.67f;  
double d = .1234;  
double result = (f*b) + (i/c) - (d*s);  
System.out.println ((f*b) + " + " + (i/c) + " - " + (d*s));  
System.out.println ("result = " + result);
```

▪ Ensanchar: automático	▪ Estrechar: cast explícito	▪ char: cast explícito
f = c;	c = (char) i;	s = (short) c;
d = s;	b = (byte) d;	c = (char) s;
f = l;	f = (float) d;	c = (char) b;

14

## Operadores

- 46 operadores
- Numéricos
  - + , - , \* , / , %
  - += , -= , \*= , /= , %= , -- , ++
- Lógicos
  - & , | , ^ , ! , && , ||
- Operadores de bits
  - & , | , ^ , ~ , >> , <<
- Relacionales
  - Cualquier tipo: == , !=
  - Tipos numéricos: > , < , >= , <=
- Condicional
  - If-then-else: (condición)? acción1 : acción2

15

## Precedencia de operadores

[]	.	(params)	expr++	expr--	
++expr	--expr	+expr	-expr	~	!
new	(type) expr				
*	/	%			
+	-				
<<	>>	>>>			
<	>	<=	>=	instanceof	
==	!=				
&					
^					
&&					
? :					
=	*=	/=	%=	&=	^=
	=	<<=	>>=	>>>=	

16

## Arrays

- Declaración  
`int a[];`
- Reserva de memoria  
`a = new int[3];`  
`int b[] = {34, 23, 9, 1, 18};`
- Asignación de valores  
`a[1] = 419;`  
`b = a; // Válido: los arrays no son constantes`
- Errores  
`a[6] = 7; // Fuera de rango`  
`a = {51, 29, 7}; // Sólo válido en inicialización`  
`int c[5]; // La dimensión sólo al hacer new`  
`char str[] = "hola"; // Los strings no son arrays`
- Los arrays son objetos de la clase `java.lang.Array`, tienen una variable `length`

17

## Arrays multidimensionales

```
float a[][] = new float [4][3];
float m[][] = new float [4][];
m[0] = new float [2];
m[1] = new float [5];
m[2] = new float [m[1].length];
float x[] = {4.5, 8/3, m[2][1]};
m[3] = x; // O cualquier expresión que devuelva un float[]
a[0] = m[1]; // Las variables de array no son constantes
a[2, 1] = 7.3 // Error de sintaxis: a[2][1]
```

18

## Condicionales

```
if (condición) acción1 [else acción2]  
  
switch (expresión) {  
  case valor1:  
    ...  
    break;  
  ...  
  case valorn:  
    ...  
    break;  
  default:  
    ...  
}
```

byte, char, short ó int

tienen que ser literales

19

## Iteraciones

```
while (condición) {  
  ...  
}  
  
do {  
  ...  
} while (condición)  
  
for (inicialización; condición; iteración) {  
  ...  
}  
  
for (tipo variable : colección/array) {  
  ...  
}
```

20

## break

```
boolean t = true;
a: {
b:   {
c:     {
        System.out.println ("Antes de break");
        if (t) break b;
        System.out.println ("Esto no se ejecuta");
      }
      System.out.println ("Esto no se ejecuta");
    }
  System.out.println ("Después de b");
}
```

21

## continue

```
for (int i = 0; i < 10; i++) {
  System.out.print( i + " ");
  if (i % 2 == 0) continue;
  System.out.println ();
}

outer: for (int i = 0; i < 10; i++) {
  for (int j = 0; j < 10; j++) {
    if (j > i) {
      System.out.println ();
      continue outer;
    }
    System.out.print (" " + (i * j));
  }
}
```

22