

Introducción a la tecnología Java

Ricardo Pérez López

IES Doñana, curso 2020/2021



1. Introducción
2. Compilación vs. interpretación
3. Características de Java
4. Tipado estático vs. dinámico
5. El primer programa Java

1. Introducción

1.1 Historia

1.2 Versiones

1.3 Ediciones

1.4 Características principales

1.1. Historia

Historia

- ▶ Java es un lenguaje creado en 1995 por James Gosling en la empresa Sun Microsystems.
- ▶ La versión 1.0 se publicó en enero de 1996.
- ▶ Desde 2002, la evolución del lenguaje ha sido regulada por el JCP (Java Community Process).



Logo de Java

- ▶ El 13 de noviembre de 2006, Sun publicó gran parte de la tecnología Java como software libre, bajo los términos de la Licencia Pública General GNU (GPL).
- ▶ En 2010, Oracle Corporation compra Sun Microsystems, por lo que Java pasa a formar parte de Oracle.

1.2. Versiones

Versiones

Versión	Publicación
JDK Beta	1995
JDK1.0	23-ene-1996
JDK 1.1	19-feb-1997
J2SE 1.2	8-dic-1998
J2SE 1.3	8-may-2000
J2SE 1.4	6-feb-2002
J2SE 5.0	30-sep-2004
Java SE 6	11-dic-2006
Java SE 7	28-jul-2011

Versión	Publicación
Java SE 8	18-mar-2014
Java SE 9	21-sep-2017
Java SE 10	20-mar-2018
Java SE 11	25-sep-2018
Java SE 12	19-mar-2019
Java SE 13	17-sep-2019
Java SE 14	17-mar-2020
Java SE 15	15-sep-2020

1.3. Ediciones

Ediciones

- ▶ Sun ha definido y soporta cuatro ediciones distintas de Java, centradas en diferentes entornos de aplicaciones y segmentando mucha de sus API.
- ▶ Las plataformas son:
 - **Java Card**: para tarjetas inteligentes.
 - **Java Platform, Micro Edition (Java ME)**: para entornos con recursos limitados.
 - **Java Platform, Standard Edition (Java SE)**: para entornos de estaciones de trabajo.
 - **Java Platform, Enterprise Edition (Java EE)**: para grandes empresas o entornos de Internet.

1.4. Características principales

Características principales

- ▶ Los cinco objetivos principales que se plantearon al diseñar el lenguaje Java son:
 - Debe ser sencillo, orientado a objetos y basado en una sintaxis conocida.
 - Debe ser robusto y seguro.
 - Debe ser portable y de arquitectura neutral, permitiendo la ejecución de un mismo programa en varios sistemas operativos.
 - Debe ejecutarse con gran rendimiento.
 - Debe ser interpretado, multihilo y de enlace dinámico.

2. Compilación vs. interpretación

2.1 Máquinas reales vs. virtuales

2.2 Código objeto (*bytecode*)

2.3 La plataforma Java

2.4 El entorno de ejecución de Java (JRE)

2.5 Las herramientas de desarrollo de Java (JDK)

2.1. Máquinas reales vs. virtuales

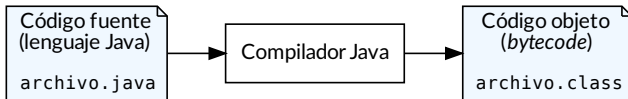
Máquinas reales vs. virtuales

- ▶ Una **máquina abstracta** es una máquina diseñada independientemente de una determinada tecnología de fabricación.
- ▶ Su finalidad principal es la de servir como **modelo de computación teórica**.
- ▶ Una **máquina virtual** es una máquina simulada mediante un software.
- ▶ Las máquinas virtuales pueden ser simulaciones de máquinas reales o abstractas.

2.2. Código objeto (*bytecode*)

Código objeto (*bytecode*)

- ▶ El compilador de Java traduce el código fuente (archivos con extensión `.java`) en código objeto (código binario almacenado en archivos con extensión `.class`) para una máquina virtual llamada **Java Virtual Machine (JVM)**.
- ▶ Al código objeto generado por el compilador de Java se le denomina ***bytecode***.
- ▶ Por tanto, podemos decir que el *bytecode* es el lenguaje máquina al que compila el compilador de Java y el lenguaje que entiende la máquina virtual de Java.



2.3. La plataforma Java

2.3.1 La máquina virtual de Java (JVM)

2.3.2 La API de Java

La plataforma Java

- ▶ La **plataforma Java** es el nombre de una arquitectura de desarrollo y ejecución de programas que se compone de un amplio abanico de tecnologías:
 - El lenguaje de programación Java.
 - La biblioteca estándar de Java.
 - La máquina virtual de Java (también llamada **Java Virtual Machine (JVM)**).
 - La implementación de la JVM y la biblioteca estándar (también llamadas **Java Runtime Environment (JRE)**).
 - Las herramientas de desarrollo (también llamadas **Java Development Kit (JDK)**).

La máquina virtual de Java (JVM)

- ▶ La **máquina virtual de Java** (del inglés, *Java Virtual Machine* o **JVM**) es una máquina virtual capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial llamado *bytecode*, el cual es generado por el compilador del lenguaje Java, entre otros.
- ▶ El *bytecode* de Java no es un lenguaje de alto nivel, sino un verdadero **código máquina de bajo nivel**, viable incluso como lenguaje de entrada para un microprocesador físico.
- ▶ La gran ventaja de usar la máquina virtual de Java es la **portabilidad**, de manera que se han creado diferentes implementaciones de la misma máquina virtual para diferentes arquitecturas, y, así, un código objeto (archivo `.class`) puede ser ejecutado en cualquier sistema operativo y arquitectura hardware que disponga de una implementación de la máquina virtual.

- ▶ La regla máxima del diseño de Java es:
«Escríbelo una vez, ejecútalo en cualquier parte.»
«Write once, run anywhere.»

- ▶ La máquina virtual de Java puede estar implementada en software, hardware, una herramienta de desarrollo o un navegador web.
- ▶ Lee y ejecuta código *bytecode* independiente de la plataforma.
- ▶ La JVM proporciona definiciones para un conjunto de instrucciones, un conjunto de registros, un formato para archivos de clases, la pila, un montículo con recolector de basura y un área de memoria.
- ▶ La definición detallada de la JVM está especificada.
- ▶ Por tanto, cualquier implementación debe cumplir con la especificación.

La API de Java

- ▶ Una **API (*Application Programming Interface*)** define un conjunto de funcionalidades recogidas en funciones y/o métodos que ofrece una determinada biblioteca para ser utilizado como una capa de abstracción por otro software o por el programador de un lenguaje de programación.
- ▶ Los sistemas operativos ofrecen servicios para simplificar la tarea de programación.
- ▶ Esos servicios se ofrecen en forma de un conjunto de bibliotecas dinámicas que las aplicaciones pueden llamar cuando lo necesiten.
- ▶ Como la plataforma Java está pensada para ser independiente del sistema operativo subyacente, las aplicaciones no pueden apoyarse en servicios ofrecidos por cada sistema en concreto.
- ▶ Por tanto, lo que hace la Plataforma Java, es ofrecer una biblioteca estándar que contiene mucha de las funciones reutilizables disponibles en los sistemas operativos actuales.

- ▶ Esa biblioteca es accesible desde Java a través de la API de Java.
- ▶ La API de Java especifica, por tanto, el contenido de esa biblioteca, que ofrece sus servicios en forma de **clases** y otros elementos relacionados (como *interfaces*).
- ▶ La documentación del API de la versión 14 de la plataforma Java Standard Edition (Java SE) se encuentra bajo la siguiente dirección:

<https://docs.oracle.com/en/java/javase/14/docs/api/index.html>

- ▶ Allí podemos comprobar que el API de Java SE está dividido en *módulos*, que a su vez se dividen en *paquetes*, que a su vez se dividen en *clases* e *interfaces*.
- ▶ Es fundamental tener siempre a mano la documentación de la API para poder programar con agilidad en este lenguaje.

2.4. El entorno de ejecución de Java (JRE)

2.4.1 El intérprete `java`

2.5. Las herramientas de desarrollo de Java (JDK)

2.5.1 El compilador `javac`

2.5.2 El intérprete interactivo `jshell`

3. Características de Java

4. Tipado estático vs. dinámico

5. El primer programa Java

5.1 El método `main()`

5.2 La clase `System`

5.3 El objeto `out`

5.4 El método `println()`

5.5 Compilación y ejecución en consola y en el IDE

5.1. El método `main()`

5.2. La clase System

5.3. El objeto out

5.4. El método `println()`

5.5. Compilación y ejecución en consola y en el IDE