INTRODUCCION JAVA

1. EL LENGUAJE DE PROGRAMACIÓN JAVA

El origen del lenguaje de programación Java se sitúa a principios de los años 90.

Un grupo de ingenieros liderados por **James Gosling** trabajaban en un proyecto para la empresa Sun Microsystems. Este grupo de ingenieros era conocido como el Green Team.

El propósito de este proyecto era desarrollar software específico para programar pequeños dispositivos electrónicos: electrodomésticos y otros aparatos electrónicos de consumo.

Cada dispositivo tenía un software específico incompatible entre ellos.

El objetivo del proyecto era crear funciones que se pudieran aplicar a cualquier dispositivo compatible sin necesidad de modificar y recompilar el código para cada uno de ellos.

El objetivo de Sun era crear un lenguaje independiente del dispositivo.

Los lenguajes de programación existentes (como C ó C++) no servían para ese propósito. En estos lenguajes los programas son dependientes del dispositivo y deben ser modificados y compilados de nuevo para que se puedan ejecutar en dispositivos diferentes.
Por este motivo Gosling y su equipo crearon un nuevo lenguaje que fuera capaz de crear aplicaciones independientes de la arquitectura en la que se ejecutaran.

Inicialmente este nuevo lenguaje se llamó **GreenTalk**.

Posteriormente le cambiaron el nombre y pasó a llamarse **Oak** (Roble)

Mientras Gosling y su equipo trabajaban en este proyecto, ocurrió algo que hizo que la World Wide Web, que hasta ese momento no estaba muy extendida, diera un salto de gigante:

En 1993 se crea el navegador gráfico **Mosaic**.

Este navegador facilitaba la navegación entre los hiperenlaces que componían la web y esto hizo que la web se popularizara.

Ante el auge que estaba experimentando internet, los desarrolladores de Sun pensaron que los logros conseguidos con su proyecto de lenguaje para pequeños electrodomésticos eran perfectamente aplicables a Internet.

Básicamente Internet es una gran red mundial que conecta ordenadores con **distintos sistemas operativos y distintas arquitecturas**.

Sus trabajos se dirigieron entonces hacia el desarrollo de un lenguaje que permitiera crear aplicaciones que se ejecutaran en cualquier ordenador de Internet, sin importar su S.O. ni su arquitectura.

Objetivo: Crear un lenguaje multiplataforma que se pudiera ejecutar en cualquier ordenador independientemente de su arquitectura o sistema operativo.

En **1995** Oak pasa a llamarse **Java**.

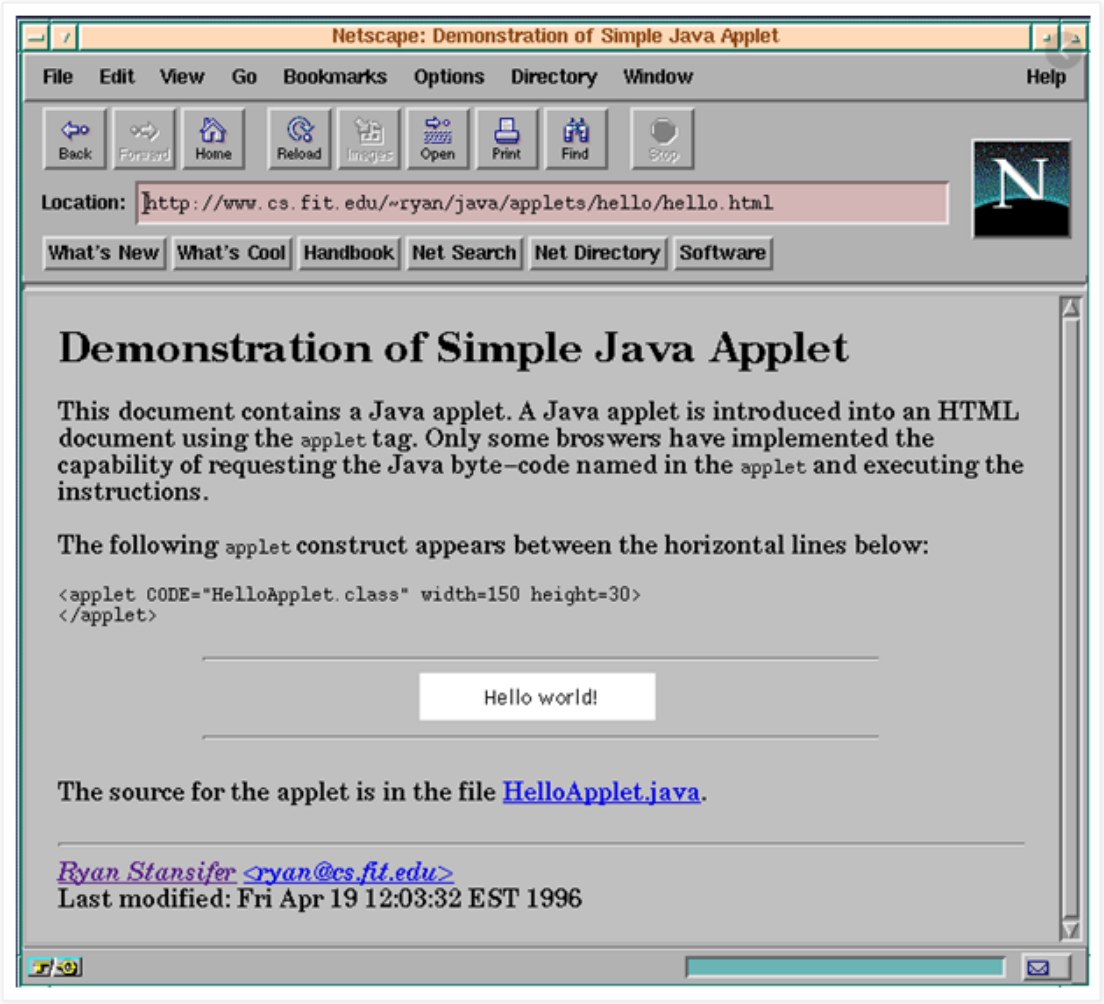
Dice la leyenda que Java debe su nombre a la cantidad de tazas de café que consumieron sus creadores durante el proceso.

Ese mismo año lanzan el navegador **HotJava**.

El navegador soportaba *applets* de Java. Pronto muchos programadores se interesaron por el nuevo lenguaje y empezaron a desarrollar *applets* Java.

HotJava demostró todo el potencial del nuevo lenguaje y ayudó a que se popularizara.

En 1997 Netscape anunció que la versión 2.0 de su navegador **Netscape Navigator** soportaría applets de Java. Este fue el impulso final que hizo que grandes compañías como IBM o SGI se interesaran por Java.



Sun Microsystems creó **JavaSoft**, un grupo empresarial dedicado exclusivamente al desarrollo del lenguaje.

Java logró convertirse en un **lenguaje multiplataforma.**

El eslogan que Sun popularizó sobre el lenguaje Java fue:

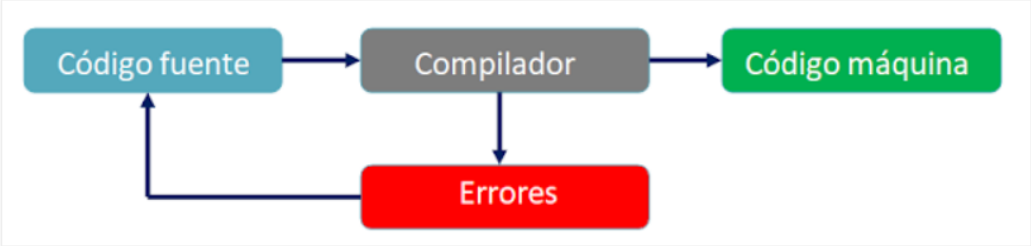
Write Once, Run Anywhere

También conocido con sus iniciales *WORA*. Esto quiere decir que un programa Java se escribe una vez y se puede ejecutar en cualquier plataforma sin tener que ser modificado ni recompilado.

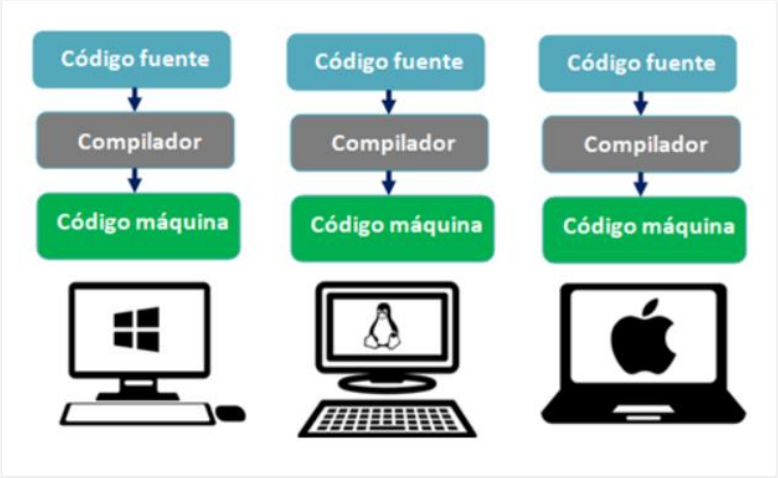
Pero... ¿cómo lo consiguió?

Normalmente un programa escrito en un lenguaje de programación (código fuente) se debe traducir a un lenguaje entendible por la máquina (código máquina o ejecutable).

La traducción la realiza un programa traductor, generalmente un compilador.



Este código máquina o código ejecutable es distinto para cada arquitectura.



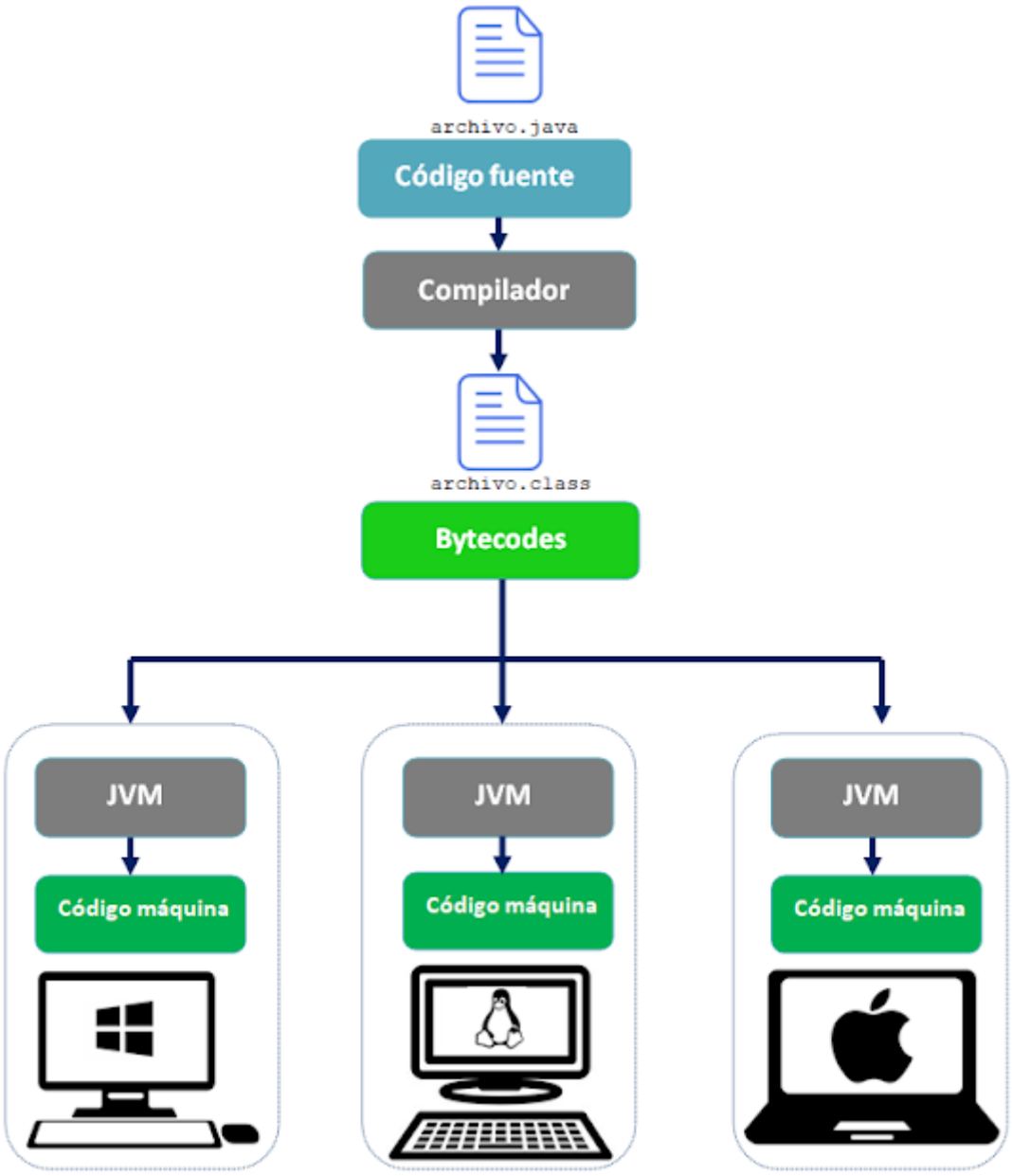
Si cambia el código fuente hay que realizar una compilación para cada tipo de arquitectura donde ese vaya a ejecutar el programa.

Para eliminar la dependencia de la máquina:

- En Java un programa no se traduce directamente a código ejecutable.
- Un programa Java se compila y se obtiene un código intermedio llamado **bytecode**.
- El bytecode lo interpreta la **Máquina Virtual de Java** (**JVM** Java Virtual Machine) y obtiene el código ejecutable.

La JVM forma parte del **JRE** (Java Runtime Environment) o Entorno de ejecución de Java.

La máquina virtual de Java se distribuye gratuitamente para practicamemnte todos los sistemas operativos.
Un archivo .class (bytecode) se puede ejecutar en cualquier ordenador que tenga instalada la máquina virtual java.



El programa Java se compila una única vez y el fichero bytecode que se obtiene se procesa por la máquina virtual de Java instalada en cualquier plataforma (Windows, Linux, MacOS, etc).

De esa forma Java logra ser un lenguaje que no depende de una arquitectura específica.

En 2010 **Oracle** compró Sun Microsystems por 7.400 millones de dólares.

2. CARACTERÍSTICAS DEL LENGUAJE JAVA

SENCILLO

Elimina la complejidad de los lenguajes como C y da paso al contexto de los lenguajes modernos orientados a objetos. Aunque la sintaxis de Java es muy similar a C++, elimina algunas de las características más conflictivas de este lenguaje, entre ellas:

- No hay punteros.
- No hay sobrecarga operadores.
- No permite la herencia múltiple.
- No hay necesidad de liberar memoria manualmente. La gestión de memoria dinámica se hace automáticamente (recolector de basura).

ORIENTADO A OBJETOS

Es un lenguaje orientado a objetos puro. En Java todo, a excepción de los tipos fundamentales de variables (int, char, double...) es un objeto.

MULTIPLATAFORMA

Para eliminar la dependencia de la máquina, en Java un programa no se traduce directamente a código ejecutable.

Un programa Java (.java) se compila y se obtiene un código llamado bytecode (.class).

El bytecode lo interpreta la **Máquina Virtual de Java** (JVM).

También se conoce como **JRE** (**Java Runtime Environment**, entorno de ejecución de Java).

El JRE o la máquina virtual de Java se distribuye gratuitamente para prácticamente todos los sistemas operativos, lo que significa que un archivo .class se puede ejecutar en cualquier ordenador o máquina que incorpore el JRE.

Los bytecodes se interpretan por diferentes computadoras de igual manera, por lo que únicamente hay que implementar una máquina virtual para cada plataforma.

El programador compila una única vez el programa Java, y el fichero de bytecode que obtiene se ejecuta igual por la máquina virtual de Java de cualquier plataforma (Windows, Linux, MacOS, etc). De esa forma Java logra ser un lenguaje que no depende de una arquitectura de ordenador específica.

La ejecución mediante la máquina virtual hace que la ejecución de los programas Java sea más lenta que la de los programas escritos en C++.

Para aumentar la velocidad de ejecución se pueden emplear:

- **Compiladores de código nativo**: el programa java se compila y se obtiene directamente código máquina ejecutable. Perdemos la portabilidad.
- **Compiladores JIT (Just-In-Time)**: A medida que se van interpretando los bytecodes se guarda el código máquina generado en un buffer. De esta forma no es necesario volver a traducir de nuevo el código ya traducido con lo que ganamos en velocidad.

ROBUSTO

Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución.

La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error.

Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

También implementa los *arrays* auténticos, en vez de listas enlazadas de punteros, con comprobación de límites, para evitar la posibilidad de sobreescribir o corromper memoria resultado de punteros que señalan a zonas equivocadas. Estas características reducen drásticamente el tiempo empleado en el desarrollo de aplicaciones Java.

SEGURO

Al contrario de lo que sucede con C/C++, no se puede acceder a la memoria mediante punteros.
Además Java incorpora medidas que evitan que se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no se puede hacer con los recursos críticos de una computadora.

MULTITAREA

Soporta múltiples *threads*, hilos o tareas. Esto quiere decir que puede ejecutar diferentes líneas de código al mismo tiempo tanto si la máquina es multiprocesador como si no lo es.

DINAMICO

En Java no es necesario cargar completamente el programa en memoria sino que las clases compiladas pueden ser cargadas bajo demanda en tiempo de ejecución (dynamic binding).
Esto proceso permite la carga de código bajo demanda, lo que es especialmente importante en los applets.

5. LA PLATAFORMA JAVA

Una plataforma es el ambiente de software o hardware en el que corre un programa.

La plataforma Java consta de dos componentes:

- La máquina virtual de Java

- La API de Java (Application Programming Interface, interfaz de programación de aplicaciones)

La API de Java es una colección de bibliotecas con clases estándar.

Estas clases se pueden incluir en los programas Java, sin temor a fallos de portabilidad. Además, están bien documentadas (mediante páginas Web), y organizadas en paquetes y en un gran árbol de herencia.

A este **conjunto de paquetes** se le conoce como la **API** de Java.

Los paquetes básicos de la API de Java son:

PAQUETES DE UTILIDADES

- **java.lang**: Fundamental para el lenguaje. Incluye clases como String o StringBuffer.
- **java.io**: Para la entrada y salida a través de flujos de datos, y ficheros del sistema.
- **java.util**: Contiene colecciones de datos y clases, el modelo de eventos, facilidades horarias, generación aleatoria de números, y otras clases de utilidad.
- **java.math**: Clases para realizar aritmética con la precisión que se desee.
- **java.text**: Clases e interfaces para manejo de texto, fechas, números y mensajes de una manera independiente a los lenguajes naturales.
- **java.security**: Clases e interfaces para seguridad en Java: Encriptación RSA...

6. El JDK

Para poder escribir un programa Java es necesario tener instalado el Kit de Desarrollo de Java o JDK (**Java Development Kit**), también llamado Java SDK (Software Development Kit).

El JDK contiene el software necesario para que los programadores compilen, depuren y ejecuten programas y applets escritos en Java.

Tanto el software como la documentación son gratuitos. Se puede descargar en:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

En esta página también se puede descargar el **JRE**.

El entorno JDK no es el más adecuado para el desarrollo de aplicaciones Java, debido a funcionar única y exclusivamente **mediante comandos de consola**:

javac Es el comando compilador de Java.

Su sintaxis es:

javac ejemplo.java

La entrada de este comando ha de ser necesariamente un fichero que contenga código escrito en lenguaje Java y con extensión .Java. El comando nos creará un fichero .class por cada clase que contenga el fichero Java.

Los ficheros .class contienen código bytecode, el código que es interpretado por la máquina virtual.

java Es el intérprete de Java.

Permite ejecutar aplicaciones que previamente hayan sido compiladas y transformadas en ficheros .class.

Su sintaxis es:

java ejemplo

No es necesario aquí suministrar la extensión del fichero, ya que siempre ha de ser un fichero .class.

appletviewer Se trata de un comando que verifica el comportamiento de un applet. La entrada del comando ha de ser una página web que contenga una referencia al applet que deseamos probar.

Su sintaxis es:

appletviewer mipagina.html

El comando ignora todo el contenido de la página web que no sean applets y se limita a ejecutarlos.

javadoc Permite generar documentación en formato html sobre el contenido de ficheros con extensión .Java.