

Java Evolution

Bryan Ivan Montiel Ortega

Octubre 2023

Índice

Índice.....	1
¿Qué es Java?.....	3
Características de Java.....	3
Independiente de la arquitectura.....	3
Orientado a objetos.....	3
Interpretado.....	4
Distribuido.....	4
Multihilo.....	4
Robusto y seguro.....	4
Java Platform.....	4
Tecnologías Java.....	5
Java SE.....	5
Java EE.....	5
Java ME.....	5
Java Cards.....	5
Obtaining Java.....	6
Descargar Java SE (Java Standard Edition) de Oracle:.....	6
Descargar OpenJDK:.....	6
Descargar Java IDEs (Entornos de Desarrollo Integrado):.....	6
Sitio Java.com.....	6
Ediciones y Versiones de Java.....	7
Ediciones.....	7
Java SE.....	7
Java EE.....	8
Java ME.....	8
Java FX.....	9
Java Card.....	9
Versiones.....	9
Java versión 1.0.....	9
Java versión 1.1.....	10
Java versión 1.2.....	10
Java versión 1.3.....	10
Java versión 1.4.....	10
Java versión 1.5.....	11
Java versión 1.6.....	11
Java versión 1.7.....	11
Java versión 8.....	12
Java versión 9.....	12

Java versión 10.....	12
Java versión 11.....	13
Java versión 12.....	13
Java SE 18.....	13
Java SE 19.....	13
Java SE 20.....	14
Historia de Java Java History.....	14
EI JDK.....	15
Componentes del JDK.....	15
Compilador Java.....	15
Depurador de Java.....	15
Máquina virtual Java (JVM).....	15
Entorno de ejecución de Java (JRE).....	16
¿Cómo funciona el JDK?.....	16
Instalación del JDK en su sistema.....	16
Versiones y actualizaciones del JDK.....	16
Your First Java Program.....	16
Paso 1. Crear un nuevo archivo.....	16
Paso 2. Escriba la plantilla del programa.....	17
Paso 3. Prepara todo para escribir la instrucción.....	17
Paso 4. Escribe el comando.....	18
Paso 5. Guarde su archivo como un programa.....	18
Paso 6. Configure el kit de desarrollo de Java.....	19
Paso 7. Prepárate para compilar.....	19
Paso 8. ¡Es hora de compilar!.....	19
Paso 9. ¡Pruebe su programa!.....	19
Revisiting the JVM.....	20
Ventajas de utilizar la JVM.....	20
Componentes de la JVM.....	20
Gestión de la memoria en la JVM.....	21
Detalles de implementación de JVM.....	21
API Documentation.....	22
¿Qué es un módulo de Java?.....	22
¿Cómo se importa un paquete en Java?.....	22
¿Cómo se puede ver la referencia de esto en la API?.....	22
Bibliografía.....	23

¿Qué es Java?

Java es un lenguaje de programación orientado a objetos y una plataforma de software ampliamente utilizado que se ejecuta en miles de millones de dispositivos, que incluyen computadoras portátiles, dispositivos móviles, consolas de juegos, dispositivos médicos y muchos otros. Las reglas y la sintaxis de Java se basan en los lenguajes C y C++.

Una de las principales ventajas de desarrollar software con Java es su portabilidad. Una vez que haya escrito el código para un programa Java en una computadora portátil, es muy fácil mover el código a un dispositivo móvil. Cuando el lenguaje fue inventado en 1991 por James Gosling de Sun Microsystems (luego adquirido por Oracle), el objetivo principal era poder "escribir una vez, ejecutar en cualquier lugar".

Java es un lenguaje de programación orientado a objetos de propósito general diseñado para tener menores dependencias de implementación. Es una plataforma informática para el desarrollo de aplicaciones. Java es rápido, seguro y confiable. Por lo tanto, se utiliza ampliamente para desarrollar aplicaciones en Java en computadoras portátiles, centros de datos, consolas de juegos, supercomputadoras científicas, teléfonos celulares, etc.

Características de Java

Java se caracteriza por ser un lenguaje independiente de la arquitectura, orientado a objetos, interpretado, distribuido, multihilo, robusto y seguro.

Independiente de la arquitectura

Gracias a la máquina virtual de Java (Java Virtual Machine JVM) se puede ejecutar el mismo programa en cualquier sistema operativo (Linux, Windows, MacOS, etc.) y en cualquier hardware sin necesidad de hacer modificaciones sobre el mismo. Esto convierte a Java en un lenguaje multiplataforma y 100% portable.

Java es conocido por ser un lenguaje independiente de la arquitectura, lo que significa que el código Java puede ejecutarse en múltiples plataformas sin necesidad de cambios significativos. Esto se debe en gran parte a la máquina virtual Java (JVM), que permite que el código Java sea ejecutado en diferentes sistemas operativos.

Orientado a objetos

Java es un lenguaje orientado a objetos puesto que permite la definición de clases y la instancia de objetos de las mismas.

Además, tiene todas las características comunes de la programación orientada a objetos: polimorfismo, herencia y enlazado dinámico. También permite definiciones abstractas de clases. Sin embargo, Java también admite tipos de datos no primitivos como int, float, boolean, double, long, etc. Por lo tanto, Java no es un lenguaje de programación orientado a objetos puro

Interpretado

No es cierto que Java sea 100% interpretado. Hay una fase inicial en la que el código se compila para generar los ficheros .class de tipo Bytecode.

Los ficheros Bytecode generados tras la compilación son los que la Máquina Virtual de Java (JVM) interpretará durante su ejecución. Si bien esto puede considerarse una forma de interpretación, es importante destacar que Java utiliza una compilación anticipada (Ahead-of-Time Compilation) en algunos casos, lo que puede acelerar la ejecución del código.

Esta fase de 'interpretado' es muy importante puesto que es la que permite ejecutar los programas implementados en Java de forma independiente a la plataforma.

Distribuido

Existen en Java multitud de bibliotecas que permiten a las aplicaciones utilizar protocolos de red como http o ftp. Gracias a esto no es necesario tener una única aplicación corriendo en una máquina. Se pueden implementar múltiples aplicaciones en distintos nodos que se comuniquen a través de la red.

Multihilo

Java es capaz de ejecutar múltiples tareas de forma simultánea, lo que reduce el tiempo de ejecución y mejora el rendimiento del programa.

Aplicaciones que necesitan cargar diferentes tipos de datos (como textos e imágenes) se pueden aprovechar de esta característica para cargar la información de diferente tipo en hilos separados, de forma que se eviten bloqueos por culpa de la carga más lenta de los elementos más pesados.

Robusto y seguro

Tanto la declaración de tipos como la definición de métodos durante la fase de desarrollo ayudan a la detección de errores de forma prematura, como la conversión errónea de datos.

Posteriormente, durante la compilación y la ejecución, Java realiza multitud de comprobaciones para evitar errores.

Java se diseñó con un enfoque en la seguridad y la robustez. Incluye características como la gestión automática de memoria (recolección de basura) para evitar fugas de memoria y el control de excepciones para manejar errores de manera más predecible. También implementa medidas de seguridad, como la verificación de bytecode para prevenir ejecuciones no seguras.

Java Platform

La plataforma Java™ es el entorno para desarrollar y gestionar applets y aplicaciones Java. Consta de tres componentes principales: el lenguaje Java, los paquetes Java y la máquina virtual Java.

La "Java Platform" (Plataforma Java) se refiere a un conjunto de tecnologías, especificaciones y entornos de desarrollo proporcionados por Oracle Corporation (anteriormente Sun Microsystems) y la comunidad de código abierto a través del Proyecto OpenJDK (Open Java Development Kit). La plataforma Java se utiliza para desarrollar y ejecutar aplicaciones en el lenguaje de programación Java.

Tecnologías Java

Java SE

Es la plataforma estándar y objetivo de este tutorial sobre Java en la cual se recogen todas las funcionalidades básicas del lenguaje.

Dentro de estas funcionalidades básicas de Java encontramos: el uso de colecciones, acceso a ficheros con Java IO y NIO y bases de datos con JDBC, librerías para el desarrollo de aplicaciones de escritorio o web como Swing o JavaFX, librerías para la fecha y hora, posibilidad de crear aplicaciones multi-hilo, capacidades para realizar conexiones en red, manejo de contenido XML... incluso incluye la base de datos Java DB para el uso en memoria.

Java EE

Java EE se crea para poder realizar aplicaciones empresariales con Java. De esta forma se dota a Java EE con capacidades de desarrollo de aplicaciones de servidor con tecnologías como Servlets, JSP o EJB. Además ofrece un API de persistencia de objetos con JPA, capacidades de mensajería con Java Message, de correo electrónico con Java Mail o gestión de procesos batch.

Java ME

Java ME es la implementación de Java que nace para la creación de aplicaciones móviles.

Si bien con el paso del tiempo se ha ido enfocando más para el desarrollo de dispositivos IoT (Internet of Things): televisiones, sensores, impresoras, entre otros.

Dentro de Java ME podemos encontrar: Java TV, para el desarrollo de aplicaciones en TV o en dispositivos multimedia. Java Embedded, que nos permite crear diferentes perfiles de desarrollo de "aplicaciones incrustadas", que además no tienen interfaz gráfica.

Java Cards

Es la tecnología de Java que sirve para el desarrollo de aplicaciones que vayan a ir en tarjetas inteligentes, aquellas que llevan un chip y poca capacidad de procesamiento y memoria

Obtaining Java

Hay diferentes maneras de obtener el software Java con diferentes versiones. Estas se expondrán de 3 maneras que se pueden seguir además por último se mencionará qué ocurre desde el sitio Java y las instrucciones que muestra la página en su versión en español.

Descargar Java SE (Java Standard Edition) de Oracle:

Oracle ofrece el JDK (Java Development Kit) y el JRE (Java Runtime Environment) para Java SE en su sitio web oficial. Puedes descargar el JDK y el JRE de Java SE desde la página de descarga de Oracle.

Descargar OpenJDK:

OpenJDK es una implementación de código abierto de Java SE. Es una alternativa gratuita y de código abierto a las distribuciones de Oracle. Puedes encontrar distribuciones de OpenJDK en el sitio web oficial de OpenJDK

Descargar Java IDEs (Entornos de Desarrollo Integrado):

Para descargar un IDE de desarrollo Java, puedes visitar los sitios web de las herramientas populares como Eclipse, IntelliJ IDEA y NetBeans.

Ejemplos de sitios web para descargar IDEs:

- Eclipse: Descargar Eclipse.
- IntelliJ IDEA: Descargar IntelliJ IDEA.
- NetBeans: Descargar NetBeans.

Sitio Java.com

Aunque la mayoría de aplicaciones Java modernas combinan el tiempo de ejecución y la aplicación de Java, todavía existen algunas aplicaciones e incluso sitios web que no funcionan sin instalar Java para escritorio. El sitio web Java.com está pensado para consumidores que todavía necesitan Java en sus aplicaciones de escritorio, sobre todo las aplicaciones que tienen como destino Java 8.

Al descargar Java desde el sitio java.com se obtendrá la versión 8 de Java Runtime Environment (JRE).

JRE incluye Java Virtual Machine (JVM), las clases del núcleo de la plataforma Java y bibliotecas de la plataforma Java de compatibilidad. JRE representa la parte de tiempo de ejecución del software Java, que es todo lo que necesita para ejecutar las aplicaciones de Java WebStart desde un navegador web compatible. Sin embargo, no incorpora herramientas de desarrollo, dado que ya forman parte de Java Development Kit (JDK).

Adicionalmente, existe el Java Plugin, El software Java Plugin es un componente de Java Runtime Environment (JRE). JRE permite que algunas aplicaciones escritas en lenguaje de programación Java se inicien a través de ciertos navegadores. El software Java Plugin no es un programa independiente y no puede instalarse por separado.

Ediciones y Versiones de Java

Ediciones

A través del tiempo, también las ediciones han sido renombradas. Sin embargo, a la fecha, las ediciones de Java vigentes son las siguientes:

- Java SE
- Java EE
- Java ME
- Java FX
- Java Card

Java SE

La Java Platform, Standard Edition permite el desarrollo de aplicaciones de escritorio y servidores. Básicamente, es la edición estándar por la que todo aprendiz inicia. Además, es obligatorio tenerla instalada si quieres utilizar las demás ediciones.

Los productos que pertenecen a esta plataforma son:

- Java SE Runtime Environment o JRE: es el entorno de ejecución del lenguaje Java, también se puede ver como un conjunto de utilidades que permiten la ejecución de programas Java. JRE incluye la JVM, Java Virtual Machine o Máquina Virtual de Java, Java Plug-in que permite que las applets puedan correr en los navegadores más populares, y Java Web Start que ejecuta aplicaciones independientes a través de una red.
- Java Development Kit o JDK: es el Kit de Desarrollo de Java, el cual proporciona las herramientas necesarias para la creación de programas en el lenguaje Java.
- Java SE API: la API es la Application Programming Interface o Interfaz de programación de Aplicaciones. Ésta indica la forma en la que los programadores pueden utilizar las bibliotecas de clases compiladas de la Java SE. La API se compone de tecnologías centrales, de escritorio o cliente, y otras tecnologías:
 - Las tecnologías centrales proporcionan la funcionalidad esencial para construir aplicaciones poderosas en áreas como bases de datos, seguridad, y comunicaciones.
 - Los componentes de escritorio añaden un conjunto de características para ayudar al desarrollo de aplicaciones que proporcionan una mejor experiencia al usuario.

Java EE

La Java Platform Enterprise Edition permite desarrollar y ejecutar aplicaciones para el entorno empresarial. Ofrece una plataforma rica para el desarrollo de software para las empresas con cerca de 20 implementaciones a elegir. Java EE consiste de 28 especificaciones y un ambiente de ejecución.

Al igual que Java SE, EE se basa en especificaciones o APIs, como Enterprise Java Beans (EJB), Java Persistence API (JPA), y Java Message Service (JMS), entre otras. Estas especificaciones son ejecutadas por diversos proveedores, como pueden ser GlassFish, Oracle Weblogic o Apache TomEE.

Las aplicaciones JEE se construyen a partir de componentes.

Los componentes Java EE son los siguientes:

- Componentes de cliente: los clientes de aplicación y los applets
- Componentes Web: los componentes de la tecnología Java Servlet y Java Server Pages (JSP).
- Componentes empresariales: los componentes Enterprise JavaBeans o EJB.

En esencia una aplicación JEE es una aplicación que se escribe en lenguaje Java, pero en la que se revisa que los componentes EE cumplan con estas especificaciones y se implementen y administren en el servidor EE,

Java ME

La Java Platform Micro Edition o Java ME proporciona un ambiente robusto y flexible para aplicaciones que se ejecutan en dispositivos móviles. [Java ME – Oracle]

Los dispositivos móviles para los que fue diseñada esta edición de Java incluyen productos de consumo como teléfonos celulares, PDAs, o tablets. Pero además, a otros dispositivos especializados como sensores y microcontroladores, entre otros más que se utilizan en el Internet de las cosas.

Por ello, Java ME incluye interfaces de usuario flexibles, seguridad robusta, protocolos integrados de redes e incluso soporte para aplicaciones que se ejecutan off-line. Además tiene la ventaja de crear aplicaciones portables entre muchos dispositivos, además de ofrecer acceso a las capacidades propias de cada uno de ellos.

La arquitectura de Java ME se compone de 3 capas:

- La máquina virtual
- Una configuración: que es un conjunto mínimo de bibliotecas de clase que proporcionan funcionalidad básica a un rango de dispositivos específicos. Existen dos configuraciones: CLDC (Connected Limited Device Configuration) y la CDC (Connected Device Configuration)
- Y un perfil, que es una extensión de la configuración dirigida a satisfacer las necesidades específicas de una familia de dispositivo

Java FX

JavaFX no es una edición de Java, sino una familia de tecnologías para la creación de RIAs (Rich Internet Applications) o Aplicación de Internet Enriquecida.

La familia de tecnologías que incluye la denominación JavaFX son Java FX Script y Java FX Mobile, entre otros productos.

Básicamente Sun Microsystems buscó competir con Flash y Silverlight cuando FX fue liberado en diciembre del 2008.

Actualmente JavaFX se ha mudado al proyecto OpenJFX, como un proyecto open source para ser usado con el JDK.

Java Card

La edición Java Card se diseñó para ejecutar applets en smart cards y otros dispositivos embebidos.

Como en otras ediciones de Java, la base es una máquina virtual. Además, el kit de desarrollo incluye Java Card JRE que permite simular la ejecución de las applets como se ejecutarían en una smart card.

Sun Microsystems Inc. anunció la especificación de Java Card el 29 de octubre de 1996, y en tan sólo poco más de 10 años se convirtió en el estándar más usado para la programación de tarjetas inteligentes.

Versiones

Desde J2SE 1.4, la evolución del lenguaje ha sido regulada por el JCP (Java Community Process), que usa Java Specification Requests (JSRs) para proponer y especificar cambios en la plataforma Java. El lenguaje en sí mismo está especificado en la Java Language Specification (JLS), o Especificación del Lenguaje Java.

Java versión 1.0

Fecha de lanzamiento: 23 de enero de 1996

Principales Características

- Contiene las clases principales, la máquina virtual y el API gráfico de AWT.
- Fue una gran innovación para el mundo de la tecnología, puesto que a partir de él, los expertos proporcionaron un lenguaje independiente de la plataforma y un entorno de ejecución ligero y gratuito para las plataformas con mayor fama.
- Gracias a esta versión, los principales navegadores web añadieron, poco después, la posibilidad de ejecutar applets Java que estuvieran incrustadas en los sitios web.

Java versión 1.1

Fecha de lanzamiento: 19 de febrero de 1997

Principales características:

- Se basó en incorporar varias clases que faltaban, tales como: Readers/Writers, Calendars y Bundles.
- La mayor aportación en la versión 1.1, sin duda, fue el hecho de añadir el estándar de JavaBeans y el API de JDBC (Java Database Connectivity) para la conexión a base de datos. Lo cual, se estima como un hecho de relevancia en la informática.
- También muestra una reestructuración intensiva del modelo de eventos AWT (Abstract Windowing Toolkit).
- Incluye clases internas o “inner classes”.

Java versión 1.2

Fecha de lanzamiento: 08 de diciembre de 1998

Principales características:

- Exhibió la llegada del framework de Collections y el API de Swing. Lo que permite desarrollar interfaces de ventanas mucho más complejas.
- Contiene Java IDL, que es una implementación de IDL o “Interfaz para Descripción de Lenguaje” diseñada para la interoperabilidad con CORBA.
- Por primera vez, la máquina virtual de Sun fue equipada con un compilador “Just in Time” (JIT).
- Otras particularidades: Java Plug-in, colecciones o “collections”, la palabra reservada strictfp, etc.

Java versión 1.3

Fecha de lanzamiento: 08 de mayo del 2000

Principales características:

- Se añade soporte JNDI o “Java Naming and Directory Interface” en el paquete de librerías principales. Lo cual, anteriormente, solo estaba disponible como una extensión.
- En esta versión se da la inclusión de la máquina virtual de HotSpot JVM con compilación JIT. La cual, fue lanzada en abril de 1999 para la JVM de J2SE 1.2.
- Aquí RMI fue cambiado para que se basará en CORBA.
- Contiene JavaSound y Java Platform Debugger Architecture (JPDA).

Java versión 1.4

Fecha de lanzamiento: 06 de febrero de 2002

Principales características:

- En cuanto a las nuevas APIs, se produce un salto verdaderamente relevante. Ya que añade API I/O para la lectura y escritura de imágenes en formatos como JPEG o PNG. Al igual que Logging API (Specified in JSR 47.).
- También se incorpora un potente soporte de XML y Expresiones Regulares.
- Cuenta con seguridad integrada y extensiones criptográficas, tales como: JCE, JSSE, JAAS. Además de Java Web Start incluido.

Java versión 1.5

Fecha de lanzamiento: 30 de septiembre de 2004

Principales características:

- Gracias al uso de anotaciones en esta versión, es posible etiquetar las clases o los métodos con datos adicionales. Los cuales, puedan ser procesados por utilidades de proceso de metadatos, posteriormente.
- Se evidencia una relevante ampliación en cuanto al soporte de APIs que están orientadas a programación concurrente.
- La palabra reservada “enum” crea una typesafe; lista ordenada de valores (como Dia.LUNES, Dia.MARTES, etc.). Lo cual, solo se podía llevar a cabo por constantes enteras o clases construidas de forma manual, anteriormente.
- Muestra un “bucle for” mejorado. Puesto que, la sintaxis se ha extendido con una especial para iterar sobre cada miembro de un array o sobre cualquier clase que implemente Iterable, como la clase estándar Collection, por ejemplo.

Java versión 1.6

Fecha de lanzamiento: 11 de diciembre de 2006

- Principales características:
- Añade el motor Rhino de Mozilla, que se trata de una implementación del también reconocido lenguaje de programación JavaScript en el Java.
- Cuenta con excelentes mejoras en su rendimiento y también en la interfaz gráfica.
- Incluye un cliente completo de servicios web y, asimismo, tiene soporte para las últimas especificaciones para servicios web. Tales como: JAX-WS 2.0, JAXB 2.0, STAX y JAXP.
- Entre sus mejores ventajas, añade un novedoso marco de trabajo y APIs que permiten combinar Java con lenguajes dinámicos (PHP, Python, Ruby y JavaScript).

Java versión 1.7

Fecha de lanzamiento: Entre 2006 y 2008 se encontraba en las primeras etapas de planificación. Su lanzamiento oficial fue en julio del 2011.

Principales características:

- Tiene soporte para XML dentro de su propio lenguaje. Al igual que para closures.
- Presenta introducción de anotaciones estándar para detectar fallos en el software.
- Maneja un nuevo concepto de superpaquete.
- También añade: Nueva API para el manejo de Días y Fechas, la cual reemplazará las antiguas clases Date y Calendar, Java Module System, Java Kernel y la posibilidad de operar con clases BigDecimal usando operandos.

Java versión 8

Fecha de lanzamiento: Marzo del 2014

Principales características:

- De forma completa, incorporó la librería JavaFX dentro de la JDK de Java.
- Incluye una notable mejora en torno a la seguridad, concurrencia y la integración de JavaScript. Esto último, debido al Proyecto Nashorn para disponer de un engine JavaScript.
- Dentro de su esquema a un enfoque más funcional, también aparecen las transformaciones MapReduce.

Java versión 9

Fecha de lanzamiento: 21 de septiembre del 2017

Principales características:

- Se puede encontrar el “Project Jigsaw” dentro de esta versión de Java. Lo que, establece la modularización de la JDK.
- Ofrece un completo soporte para http 2.0.
- Se destaca también por contar con “Java Shell”; a partir del cual, será posible trabajar e interactuar al mismo tiempo al estilo “Read-eval-print loop” o RELP.

Java versión 10

Fecha de lanzamiento: 20 de marzo del 2018

Principales características:

- Se añade, de manera experimental, el compilador JIT Graal implementado en Java en la plataforma Linux.
- La presente versión incluye varios certificados raíz al keystore añadido para permitir que las conexiones TLS funcionen por defecto.
- Al instaurar la funcionalidad añadida en javac, se elimina la funcionalidad javah. De forma que, esta última fue mejorada y sustituida por javac directamente.

Java versión 11

Fecha de lanzamiento: 25 de septiembre del 2018

Principales características:

- Proporciona una versión LTS en la que las grandes empresas confiarán como base para todos sus desarrollos.
- Soporta Unicode 10 con 16018 nuevos caracteres soportados, 128 nuevos emojis y 19 símbolos nuevos para el estándar en televisiones 4K.
- Se evidencia la eliminación de módulos Java EE y CORBA. En vista de que, estos fueron desaconsejados en versiones anteriores y así, la lista de paquetes incluye:
 - xml.ws (JAX-WS, plus the related technologies SAAJ and Web Services Metadata)
 - xml.bind (JAXB)
 - activation (JAF)
 - xml.ws.annotation (Common Annotations)
 - corba (CORBA)
 - transaction (JTA)
 - se.ee (Aggregator module for the six modules above)
 - xml.ws (Tools for JAX-WS)
 - xml.bind (Tools for JAXB).

Java versión 12

Fecha de lanzamiento: 19 de marzo del 2019

Principales características:

- Expresiones Switch (JEP 325); la cual se introdujo en fase preview y extiende dicha sentencia para ser utilizada como una expresión. Logrando así, simplificar la escritura de código diaria.
- Optimiza el recolector de basura G1 para devolver, de modo automático, un conjunto de memoria de Java al sistema operativo cuando está inactivo.
- Incluye una API para modelar descripciones nominales de archivos de clase clave y artefactos de tiempos de ejecución. A partir de la API de constantes en la JVM.
- Mejora el proceso de compilación del JDK, al momento de producir un archivo CDS haciendo uso de la lista de clases predeterminada en plataformas de 64-bit.

Java SE 18

Lanzada el 22 de marzo de 2022.

Java SE 19

Lanzada el 20 de septiembre de 2022.

Historia de Java Java History

Java se creó como una herramienta de programación para ser usada en un proyecto de set-top-box en una pequeña operación denominada the Green Project en Sun Microsystems en 1991. El equipo (green team), compuesto por trece personas y dirigido por James Gosling, trabajó durante 18 meses en Sand Hill Road, Menlo Park (California), para desarrollarlo.

El lenguaje se denominó inicialmente Oak (por un roble que había fuera de la oficina de Gosling), luego pasó a llamarse red tras descubrir que Oak era ya una marca comercial registrada para adaptadores de tarjetas gráficas, y finalmente se le renombró java.

Es frecuentada por algunos de los miembros del equipo. Pero no está claro si es un acrónimo o no, aunque algunas fuentes señalan que podría tratarse de las iniciales de sus diseñadores: James Gosling, Arthur Van Hoff y Andy Bechtolsheim. Otros abogan por el siguiente acrónimo, Just Another Vague Acronym ("simplemente otro acrónimo ambiguo más"). La hipótesis que más fuerza tiene es la de que Java debe su nombre a un tipo de café disponible en la cafetería cercana; de ahí que el icono de Java sea una taza de café caliente. Un pequeño signo que da fuerza a esta teoría es que los cuatro primeros bytes (el número mágico) de los archivos.class que genera el compilador, son en hexadecimal, 0xCAFEBAE. A pesar de todas estas teorías, el nombre fue sacado al parecer de una lista aleatoria de palabras.

Los objetivos de Gosling eran implementar una máquina virtual y un lenguaje con una estructura y sintaxis similar a C++. Entre junio y julio de 1994, tras una sesión maratónica de tres días entre John Gage, James Gosling, Patrick Naughton, Wayne Rosing y Eric Schmidt, el equipo reorientó la plataforma hacia la Web. Sintieron que la llegada del navegador web Mosaic propiciaría que Internet se convirtiese en un medio interactivo, como el que pensaban era la televisión por cable. Naughton creó entonces un prototipo de navegador, WebRunner, que más tarde sería conocido como HotJava.

En 1994, se les hizo una demostración de HotJava y la plataforma Java a los ejecutivos de Sun. Java 1.0a pudo descargarse por primera vez en 1994, pero hubo que esperar al 23 de mayo de 1995, durante las conferencias de SunWorld, a que vieran la luz pública Java y HotJava, el navegador Web. El acontecimiento fue anunciado por John Gage, el director científico de Sun Microsystems. El acto estuvo acompañado por una pequeña sorpresa adicional, el anuncio por parte de Marc Andreessen, vicepresidente ejecutivo de Netscape, de que Java sería soportado en sus navegadores. El 9 de enero del año siguiente, 1996, Sun fundó el grupo empresarial JavaSoft para que se encargará del desarrollo tecnológico. Dos semanas más tarde la primera versión de Java fue publicada.

La promesa inicial de Gosling era Write Once, Run Anywhere (Escríbelo una vez, ejecútalo en cualquier lugar), proporcionando un lenguaje independiente de la plataforma y un entorno de ejecución (la JVM)

ligero y gratuito para las plataformas más populares, de forma que los binarios (bytecode) de las aplicaciones Java pudiesen ejecutarse en cualquier plataforma.

El entorno de ejecución era relativamente seguro, y los principales navegadores web pronto incorporaron la posibilidad de ejecutar applets Java incrustados en las páginas web.

Java ha experimentado numerosos cambios desde la versión primigenia, JDK 1.0, así como un enorme incremento en el número de clases y paquetes que componen la biblioteca estándar.

El JDK

El Java Development Kit (JDK) es un kit de desarrollo de software que incluye todas las herramientas necesarias para crear aplicaciones Java. El JDK incluye un compilador, un depurador y otras utilidades importantes que facilitan a los desarrolladores escribir, probar e implantar aplicaciones Java.

El JDK puede obtenerse gratuitamente en el sitio web de Oracle Corporation y está disponible para los sistemas operativos Windows, Mac y Linux. El JDK es una herramienta esencial para los desarrolladores de Java, y se utiliza para crear una amplia gama de aplicaciones, desde aplicaciones móviles a sistemas empresariales.

Componentes del JDK

El JDK está formado por varios componentes, cada uno de los cuales desempeña un papel fundamental en el desarrollo de aplicaciones Java. Estos son algunos de los componentes clave del JDK:

Compilador Java

El compilador Java es una herramienta que convierte el código fuente Java en bytecode, que puede ser ejecutado por la JVM. El compilador comprueba la sintaxis del código y genera un mensaje de error si el código contiene algún error de sintaxis.

Depurador de Java

El depurador de Java es una herramienta que permite a los desarrolladores depurar su código Java. Permite a los desarrolladores recorrer su código línea por línea, establecer puntos de interrupción e inspeccionar variables para encontrar errores en su código.

Máquina virtual Java (JVM)

La JVM es una máquina virtual que ejecuta el bytecode de Java. Es responsable de traducir el bytecode en código máquina que puede ser ejecutado por el hardware subyacente. La JVM está disponible para los principales sistemas operativos y es lo que hace que Java sea un lenguaje independiente de la plataforma.

Entorno de ejecución de Java (JRE)

El JRE es un subconjunto del JDK que incluye todo lo necesario para ejecutar aplicaciones Java. El JRE incluye la JVM y la biblioteca de clases Java, pero no incluye el compilador Java ni otras herramientas de desarrollo.

¿Cómo funciona el JDK?

El JDK se utiliza para crear aplicaciones Java proporcionando a los desarrolladores todas las herramientas que necesitan para escribir, probar y desplegar su código. Los desarrolladores escriben su código utilizando un editor de texto o un entorno de desarrollo integrado (IDE) y, a continuación, utilizan el compilador de Java para compilar el código en bytecode.

Una vez compilado el código, puede ser ejecutado por la JVM, que traduce el bytecode a código máquina que puede ser ejecutado por el hardware subyacente. El depurador de Java puede utilizarse para encontrar y corregir errores en el código, y la biblioteca de clases Java proporciona a los desarrolladores una amplia gama de API que pueden utilizar para crear sus aplicaciones.

Instalación del JDK en su sistema

Instalar el JDK en su sistema es un proceso sencillo. Simplemente visite el sitio web de Oracle Corporation y descargue la versión del JDK adecuada para su sistema operativo. Una vez completada la descarga, ejecute el instalador y siga las instrucciones que aparecen en pantalla.

Una vez instalado el JDK, tendrás que configurar la variable de entorno `JAVA_HOME` para que apunte al directorio de instalación del JDK. Esto permitirá que otras herramientas, como el compilador y el depurador de Java, encuentren el JDK y utilicen sus herramientas.

Versiones y actualizaciones del JDK

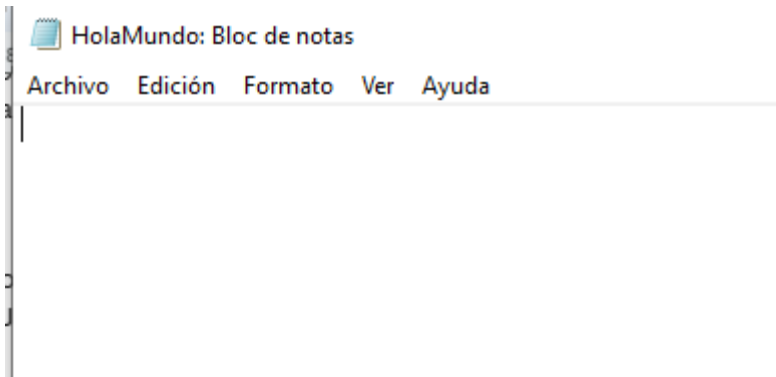
El JDK se actualiza periódicamente para corregir errores y añadir nuevas funciones. Es esencial mantener el JDK actualizado para garantizar que las aplicaciones Java sean seguras y fiables. Cuando se publique una nueva versión del JDK, podrá descargarla desde el sitio web de Oracle Corporation.

Your First Java Program

Generalmente el primer programa es el conocido 'Hola Mundo', a continuación se mostrarán los pasos para hacer el primer 'Hola Mundo' dentro de Java.

Paso 1. Crear un nuevo archivo

Utilice el explorador de archivos y diríjase al directorio deseado. Puede ser cualquier ubicación en la PC. Y se debe crear un archivo tipo documento `.txt`.



Paso 2. Escriba la plantilla del programa

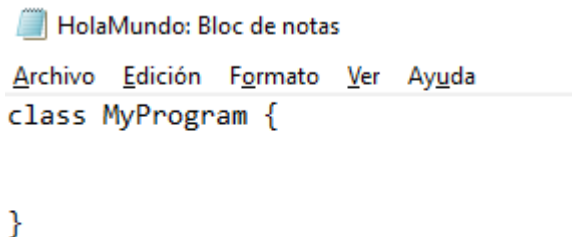
Para empezar se debe introducir el siguiente texto:

```
class MyProgram {
```

Debe escribir el corchete: este símbolo le dice al sistema dónde comienza su comando. Es el punto donde la computadora comenzará a leer su programa e iniciará las tareas dadas. Ahora, agregue dos líneas debajo y el corchete de cierre.

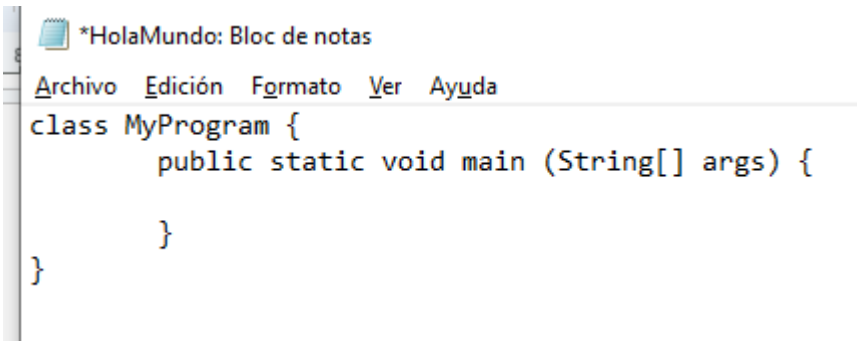
```
class MyProgram {
```

```
}
```



Paso 3. Prepara todo para escribir la instrucción

Lo siguiente que debe hacer es ingresar al método 'principal' del programa. El objetivo es agregar la instrucción en la línea debajo del corchete de apertura. Escribe lo siguiente:



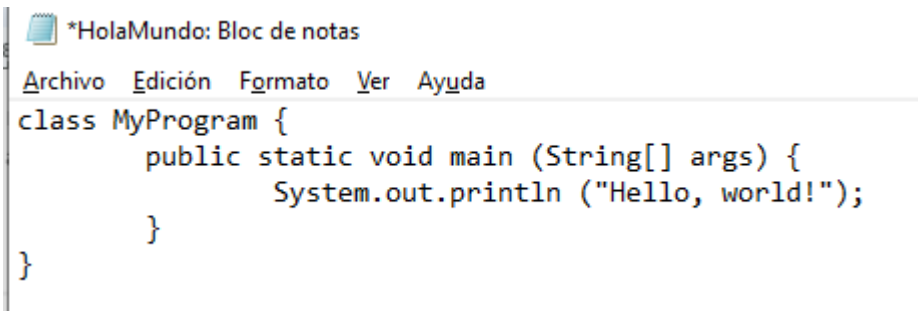
```
class MyProgram {  
    public static void main (String[] args) {  
  
    }  
}
```

Se recomienda indentar por lo que es posible utilizar un tab para lograrlo.

Paso 4. Escribe el comando

Si ejecuta el programa en este punto, no hará nada. Una forma de modificar eso es agregando las instrucciones deseadas. Imagina que queremos las palabras "¡Hola, mundo!" para aparecer en nuestra línea de comandos. Aquí está el comando que usaríamos:

```
System.out.println ("Hello, world!");
```

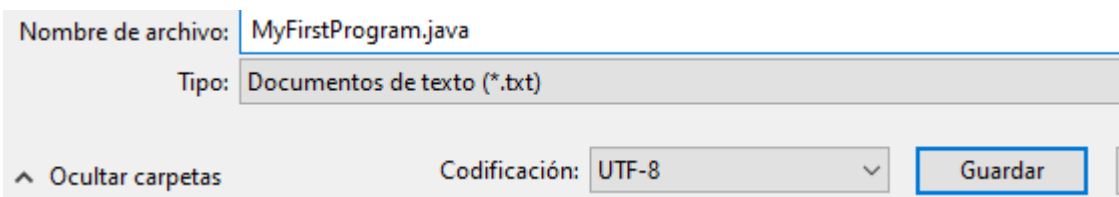


```
class MyProgram {  
    public static void main (String[] args) {  
        System.out.println ("Hello, world!");  
    }  
}
```

Al dar este comando, está accediendo al sistema y pidiéndole que "imprima" (escriba) el texto en la salida deseada.

Paso 5. Guarde su archivo como un programa

¿Recuerdas que mencionamos que habría algunos cambios de nombre? Ahora es el momento para eso. Desea ir con la opción "Guardar como...". En este punto, debe modificar el formato del archivo. Presione la opción "Guardar como tipo..." en la ventana emergente para guardar. En lugar de "*.txt", elige "Todos los archivos". Guarde su programa como "MyFirstProgram.java".

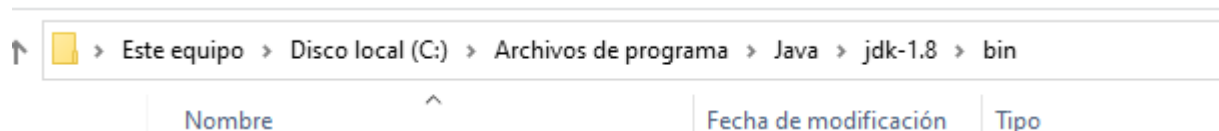


Paso 6. Configure el kit de desarrollo de Java

Java Development Kit (JDK) es esencial para ejecutar programas. Afortunadamente, JDK se puede descargar gratis y no toma mucho tiempo. Puede descargar la última versión desde el sitio web oficial de Oracle . Asegúrese de elegir uno adecuado para su sistema operativo y configuración de PC. Una vez que se complete la descarga, continúe e instale JDK. Es un proceso sin esfuerzo que solo toma un par de pasos.

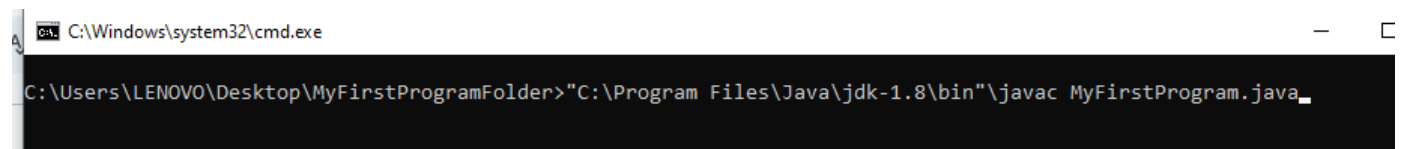
Paso 7. Prepárate para compilar

Nuestra próxima tarea es compilar el programa. Desea dirigirse a la sección Archivos de programa de su disco duro local. Aquí, debe identificar la carpeta "Java" y luego ingresar a la carpeta "jdkx.xx" (la "X" marca la versión del JDK). Ingrese al directorio "bin" y resalte su ruta en la barra de arriba. El objetivo es copiar la ruta completa (C:\Program Files\Java...").



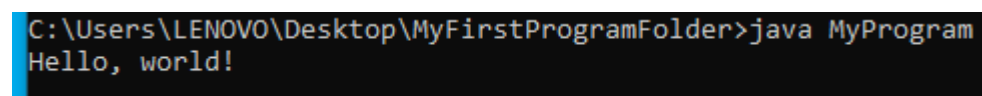
Paso 8. ¡Es hora de compilar!

La compilación es un proceso que convierte código Java (o cualquier código) de texto escrito en un archivo que su computadora puede ejecutar. Cambie a la ventana de comandos y pegue la ruta que copió en el último paso. A continuación, escriba 'javac MyFirstProgram.java'. Una vez que haya terminado, presione Entrar.



Paso 9. ¡Pruebe su programa!

Permanezca en el símbolo del sistema y presione el botón "Flecha arriba". Notará que aparece el comando de compilación. Presione eliminar para mantener solo la carpeta de herramientas JDK. Finalmente, escriba "\java MyFirstProgram" y presione enter para ver si su programa funciona correctamente. Si lo hace, notará el texto "¡Hola, mundo!" que aparece en la pantalla. ¡Felicitaciones, ha completado su primer programa Java!



Revisiting the JVM

La JVM es un componente fundamental del lenguaje de programación Java y permite ejecutar código Java en cualquier plataforma que admita la JVM. La JVM está diseñada para ser independiente de la plataforma, lo que significa que puede ejecutarse en cualquier arquitectura informática y sistema operativo que la admita. Esto hace de Java un lenguaje ideal para crear aplicaciones multiplataforma que puedan ejecutarse en cualquier sistema, desde ordenadores de sobremesa a dispositivos móviles.

La JVM funciona interpretando el código Java y traduciéndolo a código máquina que la máquina anfitriona puede ejecutar. Cuando se ejecuta un programa Java, la JVM carga primero en memoria las clases y bibliotecas necesarias. A continuación, la JVM verifica el bytecode para asegurarse de que cumple los requisitos de seguridad y corrección. Si se verifica el bytecode, la JVM comienza a interpretar el código, ejecutándolo instrucción a instrucción.

Durante el proceso de interpretación, la JVM utiliza un compilador justo a tiempo (JIT) para optimizar el código para su ejecución. El compilador JIT compila los puntos calientes del código, que son las secciones del código que se ejecutan con frecuencia. El código optimizado se almacena en memoria, lo que permite a la JVM ejecutarlo más rápidamente la próxima vez que se llame.

Ventajas de utilizar la JVM

La JVM ofrece varias ventajas que la convierten en una opción atractiva para los desarrolladores. En primer lugar, proporciona un alto nivel de independencia de plataforma, permitiendo que el código Java se ejecute en cualquier plataforma que soporte la JVM. Esto facilita a los desarrolladores la creación de aplicaciones multiplataforma, ya que no tienen que preocuparse del código específico de cada plataforma.

En segundo lugar, la JVM proporciona un alto nivel de seguridad, ya que verifica el bytecode antes de ejecutarlo. Esto impide que se ejecute código malicioso, reduciendo el riesgo de fallos de seguridad.

Por último, la JVM proporciona un rendimiento excelente, ya que utiliza un compilador JIT para optimizar el código para su ejecución. Esto permite que las aplicaciones Java se ejecuten con rapidez y eficacia, lo que las hace adecuadas para una amplia gama de aplicaciones, desde aplicaciones de escritorio hasta aplicaciones de servidor de alto rendimiento.

Componentes de la JVM

La JVM consta de varios componentes que trabajan juntos para ejecutar código Java. Estos componentes incluyen el cargador de clases, las áreas de datos en tiempo de ejecución, el motor de ejecución y la interfaz nativa.

El cargador de clases se encarga de cargar en memoria las clases y bibliotecas necesarias. Existen tres tipos de cargadores de clases: el cargador de clases bootstrap, el cargador de clases de extensión y el cargador de clases de aplicación. El cargador de clases de arranque carga las clases del núcleo de Java,

mientras que el cargador de clases de extensión carga las clases de los directorios de extensión de Java. El cargador de clases de aplicación carga clases del classpath de la aplicación.

Las áreas de datos en tiempo de ejecución consisten en el área de métodos, el montón, la pila Java y la pila de métodos nativos. El área de métodos almacena los datos de clases e interfaces, mientras que el montón almacena las instancias de objetos. La pila Java almacena las tramas de cada método, mientras que la pila de métodos nativos almacena las tramas de los métodos nativos.

El motor de ejecución se encarga de ejecutar el código de bytes. Interpreta el código de bytes y utiliza el compilador JIT para optimizar el código para su ejecución.

Por último, la interfaz nativa proporciona un puente entre el código Java y el sistema operativo anfitrión. Permite que el código Java acceda a los recursos del sistema, como archivos y sockets de red.

Gestión de la memoria en la JVM

La gestión de memoria es un aspecto esencial de la JVM, y es responsable de asignar y desasignar memoria para los objetos Java. La JVM utiliza un recolector de basura para liberar automáticamente la memoria que ya no está en uso.

El recolector de basura identifica los objetos que ya no se utilizan y recupera la memoria que ocupan. Para ello, rastrea el grafo de objetos, empezando por los objetos raíz, e identifica los objetos que ya no son accesibles. A continuación, el recolector de basura libera la memoria ocupada por estos objetos, dejándola disponible para usos futuros.

La JVM proporciona varios algoritmos de recolección de basura, incluyendo el algoritmo de marcado y barrido, el algoritmo de copia y el algoritmo generacional. Cada algoritmo tiene sus puntos fuertes y débiles, y los desarrolladores pueden elegir el algoritmo que mejor se adapte a las necesidades de su aplicación.

Detalles de implementación de JVM

La especificación JVM define un conjunto de reglas que una JVM debe seguir para garantizar la compatibilidad con el lenguaje Java. Sin embargo, diferentes implementaciones de JVM pueden tener diferentes detalles de implementación, como el algoritmo de recolección de basura utilizado o las técnicas de optimización empleadas.

Por ejemplo, la JVM HotSpot de Oracle es una de las implementaciones de JVM más populares y utiliza un algoritmo de recolección de basura generacional. También emplea varias técnicas de optimización, como inlining y loop unrolling, para mejorar el rendimiento.

Otras implementaciones de JVM son J9 JVM de IBM, que se utiliza en el servidor de aplicaciones WebSphere de IBM, y OpenJDK JVM, que es una implementación de código abierto de la JVM.

API Documentation

La API de Java es el conjunto de módulos y clases que proporciona Java para que podamos programar aplicaciones. Es como una biblioteca que contiene todo lo que necesitamos para crear programas.

¿Qué es un módulo de Java?

Un módulo es una unidad de organización y distribución que contiene un conjunto de paquetes y otros módulos.

¿Cómo se importa un paquete en Java?

Para importar un paquete en Java, primero debes conocer su nombre. Luego, por norma general, en la parte superior del código, escribes la siguiente línea:

```
import nombre_del_paquete;
```

Por ejemplo, si queremos importar la clase `Math` que nos permite realizar operaciones matemáticas, escribiríamos:

```
import java.lang.Math;
```

¿Cómo se puede ver la referencia de esto en la API?

Utiliza el buscador que tiene integrado, o busca en los diferentes módulos. Por ejemplo, `"Math"` está en `java.base > java.lang > Math`.

`"Math"` es una clase que se encuentra dentro del paquete `"java.lang"`. Este paquete, a su vez, se encuentra dentro del módulo `"java.base"`. Por lo tanto, la ruta completa sería `"java.base > java.lang > Math"`.

- `"java.lang.Math"` es una clase del paquete `"java.lang"`.
- `"java.lang"` es un paquete.
- `"java.base"` es un módulo que entre otras cosas, tiene el paquete `"java.lang"`.

Al escribir `"import java.lang.Math;"`, estamos importando la clase `"Math"` del paquete `"java.lang"` dentro del módulo `"java"`. De esta manera, podemos utilizar la clase `"Math"` en nuestro código sin tener que escribir el nombre completo del paquete y el módulo cada vez que queramos acceder a ella.

Bibliografía

1. Albornoz, F. (2019, octubre 17). ¿Cuáles y cuántas versiones de Java hay hasta la fecha? Lista 2023. Internet Paso a Paso. <https://internetpasoapaso.com/versiones-java/>
2. Fácil, P. (2023, febrero 13). La API de Java y la importación de módulos, paquetes y clases. Programación Fácil Blog; Programación Fácil. <https://programacionfacil.org/blog/la-api-de-java-y-la-importacion-de-modulos-paquetes-y-clases/>
3. González, S. M. (2023, abril 6). ¿Qué es Java? Definición, características y usos de Java. Tech Krowd. <https://techkrowd.com/programacion/java/que-es-java/>
4. Hartman, J. (2023, octubre 2). What is Java? Definition, meaning & features of Java platforms. Guru99. <https://www.guru99.com/java-platform.html>
5. IBM Documentation. (2021, marzo 8). Ibm.com. <https://www.ibm.com/docs/es/i/7.1?topic=java-platform>
6. Moraguez, E. R. (2023, marzo 28). ¿Qué es el JDK (Java Development Kit): ¿Cómo funciona y para qué sirve? LovTechnology. <https://lovtechnology.com/que-es-el-jdk-java-development-kit-como-funciona-y-para-que-sirve/>
7. Nieva, G. (2019, febrero 20). Versiones y ediciones de Java. dCodinGames; Gaby Nieva. <https://dcodingames.com/versiones-y-ediciones-de-java/>
8. ¿Que es Java? (s/f). Ibm.com. Recuperado el 27 de octubre de 2023, de <https://www.ibm.com/mx-es/topics/java>
9. Segura, J. B., & Gálvez, J. M. B. (2021, marzo 1). Historia del lenguaje de programación Java. Portal Académico del CCH. <https://portalacademico.cch.unam.mx/cibernetica1/algoritmos-y-codificacion/historia-java>
10. Squirrels, J. (2023, julio 21). Cómo comenzar a codificar en Java y escribir su primer programa hoy. CodeGym. <https://codegym.cc/es/groups/posts/es.171.como-comenzar-a-codificar-en-java-y-escribir-su-primero-programa-hoy>
11. Wikipedia contributors. (s/f-a). Java (lenguaje de programación). Wikipedia, The Free Encyclopedia. [https://es.wikipedia.org/w/index.php?title=Java_\(lenguaje_de_programaci%C3%B3n\)&oldid=154301802](https://es.wikipedia.org/w/index.php?title=Java_(lenguaje_de_programaci%C3%B3n)&oldid=154301802)
12. Wikipedia contributors. (s/f-b). Plataforma Java. Wikipedia, The Free Encyclopedia. https://es.wikipedia.org/w/index.php?title=Plataforma_Java&oldid=152061618