Evolución de Java

Autor: José Guillermo Sandoval Huerta

Fecha: 25 octubre 2023

Índice

¿Qué es Java?

Java es un lenguaje de programación de alto nivel, multipropósito y orientado a objetos que se utiliza para crear software compatible con una gran variedad de sistemas operativos. Hoy en día Java se ha convertido en uno de los lenguajes de programación más populares y ampliamente utilizados en el mundo, y su plataforma es ampliamente utilizada en aplicaciones empresariales, desarrollo web, aplicaciones móviles, sistemas embebidos y más. Java se caracteriza por su portabilidad, seguridad y facilidad de uso. Su capacidad de escribir código una vez y ejecutarlo en múltiples plataformas o sistemas ha contribuido a su popularidad y versatilidad [1].

Plataforma Java

La plataforma Java es un conjunto de tecnologías, herramientas y entornos de ejecución que permiten a su empleador desarrollar y ejecutar software en el lenguaje de programación Java. Teniendo varios componentes clave como son [2]:

- **Máquina Virtual de Java (JVM**): La JVM es un componente crítico de la plataforma. Permite la ejecución de programas Java en diferentes sistemas operativos sin necesidad de modificar el código fuente. La JVM traduce el código Java en código de máquina específico de la plataforma en tiempo de ejecución.

- **Paquetes Java:** Los paquetes Java son colecciones de clases e interfaces que proporcionan funcionalidad específica.

- **Biblioteca estándar de Java (API de Java):** Java incluye una amplia biblioteca estándar que proporciona clases y métodos para una variedad de tareas comunes, desde entrada/salida y manipulación de datos hasta operaciones de red y gráficos. Esto facilita el desarrollo de aplicaciones sin necesidad de crear todas las funcionalidades desde cero.

- **Herramientas de desarrollo:** La plataforma Java proporciona una variedad de herramientas para desarrolladores, incluyendo el compilador Java (javac), un depurador (jdb), y entornos de desarrollo integrados (IDEs) como Eclipse, NetBeans y IntelliJ IDEA.

- **Tecnologías y frameworks:** Además de la biblioteca estándar, Java tiene una amplia gama de tecnologías y frameworks que abordan diferentes áreas, como desarrollo web (Java Servlets, JSP), desarrollo empresarial (Java EE, ahora Jakarta EE), desarrollo de aplicaciones de escritorio (JavaFX), acceso a bases de datos (JDBC), y más.

- **Seguridad y administración:** Java se destaca por su enfoque en la seguridad, con características como el control de acceso a clases y la gestión de memoria automática. También ofrece herramientas y tecnologías para la administración de aplicaciones, como JMX (Java Management Extensions).

Obteniendo Java

Para obtener Java en nuestro sistema operativo, debemos descargar e instalar el Java Development Kit (JDK) en caso de querer desarrollar aplicaciones o el Java Runtime

Environment (JRE) en caso de ejecutar aplicaciones Java en nuestro ordenador. Para ello se emplea la siguiente secuencia [3]:

1. **Verificar si Java ya está instalado:**
   En muchas computadoras, Java ya está instalado, sin embargo, para verificarlo, en la línea de comandos (terminal) se ejecuta las siguientes instrucciones:

   - **java -version** - Validar versión de Java.
   - **Javac -version** - Validar compilador de Java.

   Al no tener Java se mostrará un mensaje de error.

2. **Descargar e instalar el JDK o JRE:**
   (1) Visita el sitio web oficial de Oracle [4] o la página de descargas de OpenJDK (una distribución de código abierto) [5] para obtener la última versión del JDK o JRE.
   (2) Descargar el instalador adecuado para el sistema operativo en uso (Windows, macOS o Linux).
   (3) Ejecuta el instalador y seguir las instrucciones para completar la instalación.

3. **Configurar las variables de entorno (opcional, para desarrolladores):**
   Al instalar el JDK hay que configurar las variables de entorno JAVA_HOME y PATH para que el sistema reconozca las ubicaciones de la instalación de Java. Esto es necesario para compilar y ejecutar programas Java desde la línea de comandos.
   (1) Abrir ventana “Panel de Control”.
   (2) Hacer clic en "Configuración avanzada del sistema".
   (3) En la pestaña "Opciones avanzadas", hacer clic en "Variables de entorno".
   (4) En la sección "Variables de usuario", hacer clic en "Nuevo" para agregar una nueva variable de usuario o selecciona una variable existente (como PATH) y hacer clic en "Editar" si ya existe.
   (5) Para agregar JAVA_HOME, se crea una nueva variable llamada "JAVA_HOME" y se establece su valor siendo la ubicación del JDK.
   (6) Hacer clic en "Aceptar" para guardar los cambios.

Una vez completado estos pasos, se tendrá Java instalado en el sistema y estará todo listo para desarrollar o ejecutar aplicaciones Java.

Ediciones y Versiones de Java

Java ha tenido varias ediciones y versiones a lo largo de su historia. Cada edición y versión ha introducido nuevas características, mejoras y correcciones de seguridad. A continuación, se presenta una descripción de las ediciones y algunas de las versiones más destacadas de Java:

Ediciones principales:

- **J2SE (Java 2 Platform, Standard Edition):** Esta fue la primera edición importante de Java que introdujo la plataforma Java 2 en 1998. Incluía numerosas mejoras, como la API Swing para interfaces gráficas de usuario.

- **J2EE (Java 2 Platform, Enterprise Edition):** Esta edición estaba orientada al desarrollo de aplicaciones empresariales y soluciones empresariales. Ofrecía tecnologías como Servlets y JSP (JavaServer Pages).

- **J2ME (Java 2 Platform, Micro Edition):** Estaba dirigida a dispositivos móviles y sistemas integrados. Fue ampliamente utilizado en la creación de aplicaciones para teléfonos móviles y dispositivos embebidos.

- **Java EE (Java Platform, Enterprise Edition):** La plataforma Java Enterprise Edition (anteriormente J2EE) se centró en el desarrollo de aplicaciones empresariales a gran escala y servicios web.

- **Java SE (Java Platform, Standard Edition):** La edición estándar de Java se utiliza para el desarrollo de aplicaciones de escritorio y aplicaciones independientes. A partir de Java 5, se introdujo el nuevo sistema de versiones, con números como Java 5, 6, 7, 8, 9, 10, 11, etc.

- **Java ME (Java Platform, Micro Edition):** La edición Micro Edition se utilizó en dispositivos móviles, pero ha perdido relevancia en favor de las tecnologías móviles modernas. Es una plataforma robusta y versátil que proporciona a los desarrolladores las herramientas y las APIs que necesitan para crear aplicaciones de alta calidad para estos dispositivos.

- **JavaFX:** Una plataforma para la creación de aplicaciones ricas en gráficos y multimedia en Java.

Versiones Destacadas de Java SE:

- **Java SE 5.0 (Java 1.5):** Introdujo importantes características como anotaciones, tipos genéricos y mejoras en la biblioteca estándar.

- **Java SE 6 (Java 1.6):** Incluyó mejoras en el rendimiento, la escalabilidad y la seguridad, así como soporte para scripting con el motor de scripting de Java.

- **Java SE 7 (Java 1.7):** Presentó el proyecto Coin, que incluía mejoras en la sintaxis del lenguaje, y el soporte para el manejo de excepciones multipropósito.

- **Java SE 8 (Java 1.8):** La versión más notable por introducir Lambdas, Streams y la API de Fecha y Hora.

- **Java SE 9 (Java 9):** Introdujo el concepto de módulos y una serie de mejoras en el rendimiento y la seguridad.

- **Java SE 10 (Java 10):** Se centró en mejoras menores y optimizaciones.

- **Java SE 11 (Java 11):** La primera versión con soporte a largo plazo (LTS) después de Java 8. Se eliminó JavaFX del paquete estándar y se introdujo el módulo HTTP cliente.

- **Java SE 12 (Java 12), Java SE 13 (Java 13), etc.:** Estas versiones posteriores han continuado introduciendo mejoras y características menores en el lenguaje y la biblioteca estándar.

Es importante mencionar que a partir de Java 9, Oracle implementó un nuevo ciclo de lanzamiento, con nuevas versiones cada seis meses [6].

Historia de Java

- **Década de 1990 - Los Primeros Días:**
  Java fue concebido y desarrollado inicialmente en Sun Microsystems, Inc. en 1991. James Gosling, Mike Sheridan y Patrick Naughton lideraron el proyecto inicialmente llamado "Green Project" para desarrollar software para dispositivos electrónicos y electrodomésticos.
  El equipo inicial trabajó en un nuevo lenguaje de programación, originalmente llamado "Oak," que se inspiró en C++ pero que debía superar problemas técnicos relacionados con la memoria. "Oak" en honor a un roble fuera de la oficina de Gosling, tuvo que ser renombrado a "Java" debido a problemas legales. La elección del nombre "Java" provino de una cafetería frecuentada por el equipo de desarrollo.

- **1995 - El Lanzamiento de Java:**
  En mayo de 1995, Sun Microsystems lanzó oficialmente Java con la premisa de ser "write once, run anywhere" (escribe una vez, ejecuta en cualquier lugar). Java 1.0 incluía la Máquina Virtual de Java (JVM) y una serie de bibliotecas estándar.
  La primera versión pública fue Java 1.0 en 1996.

- **1996 - Auge de Java en la Web:**
  Java se hizo popular en la web gracias a su capacidad para ejecutar applets, pequeñas aplicaciones en el navegador. Estos applets permitían una interactividad avanzada en las páginas web, pero esta forma de ejecución quedó obsoleta con el tiempo debido a preocupaciones de seguridad y a la evolución de las tecnologías web.

- **1998 - Java 2:**
  Java 2 (también conocido como J2SE) se lanzó en 1998, introduciendo importantes mejoras, como las interfaces de programación de aplicaciones (API) Swing para interfaces gráficas de usuario (GUI) y otras características significativas.

- **2000 - Plataforma Java 2, Empresa (J2EE):**
  Java 2 Enterprise Edition (J2EE) se presentó, centrado en el desarrollo de aplicaciones empresariales y soluciones empresariales, incluyendo tecnologías como Servlets y JSP (JavaServer Pages).

- **2010 - Adquisición por Oracle:**
  En 2010, Oracle Corporation adquirió Sun Microsystems, convirtiéndose en el nuevo propietario de Java.

- **2017 - Cambio en el Modelo de Lanzamiento:**
  A partir de Java 9, Oracle implementó un nuevo ciclo de lanzamiento con versiones cada seis meses, lo que permitió a los desarrolladores acceder a nuevas características con más frecuencia.

- **2023 - Versión LTS (Java 21):**
  Java 21 se convirtió en una versión LTS (Soporte a Largo Plazo), lo que garantiza actualizaciones de seguridad y soporte a largo plazo para los usuarios.

La historia de Java se caracteriza por su evolución continua, su enfoque en la portabilidad y su adopción en una amplia gama de aplicaciones, desde el desarrollo web y empresarial hasta aplicaciones móviles (como Android) y sistemas integrados. Java ha dejado una marca significativa en la informática y sigue siendo uno de los lenguajes de programación más populares y utilizados en todo el mundo [7].

JDK y JRE

El JDK (Java Development Kit) y el JRE (Java Runtime Environment) son dos componentes relacionados con la plataforma Java, pero tienen diferentes propósitos y se utilizan en contextos distintos, donde el JDK es para desarrolladores y contiene las herramientas necesarias para crear software en Java, mientras que el JRE es para usuarios finales que solo necesitan ejecutar aplicaciones Java en sus sistemas. [8]:

JDK (Java Development Kit):

- **Propósito:** El JDK está diseñado para desarrolladores que desean crear aplicaciones en Java. Proporciona las herramientas necesarias para escribir, compilar, depurar y empaquetar programas en Java. Es esencial para el desarrollo de software en Java.
- **Componentes clave:** El JDK incluye el compilador de Java (javac), la Máquina Virtual de Java (JVM) para pruebas, bibliotecas y herramientas de desarrollo como jar para crear archivos JAR.

JRE (Java Runtime Environment):

- **Propósito:** El JRE es para usuarios que desean ejecutar aplicaciones Java en sus sistemas. No es necesario si solo deseas desarrollar software en Java.
- **Componentes clave:** El JRE incluye la Máquina Virtual de Java (JVM) y las bibliotecas necesarias para ejecutar aplicaciones Java. No incluye herramientas de desarrollo como el compilador (javac).

Primer programa Java

La mayoría de las veces para iniciar a programar con cualquier tipo de lenguaje de programación es crear un programa llamado “Hola Mundo” (en inglés “Hello World”). Este programa simple se utiliza para familiarizar a los programadores con la sintaxis básica del lenguaje y para verificar que el entorno de desarrollo está configurado correctamente.

Para el caso del lenguaje Java es por medio de la siguiente secuencia:

1. Asegurarse de tener Java instalado dentro del sistema operativo.

2. En un editor de código o IDE crear y guardar un archivo con extensión ".java" que contenga el siguiente fragmento de código:

```
public class MiPrimerPrograma {
    public static void main(String[] args) {
        System.out.println("Hola, Mundo");
    }
}
```

Tabla 1. Clase “MiPrimerPrograma”.

(Opción editor de código)

3. Abrir la línea de comandos (terminal).

4. Navega al directorio que contiene el archivo Java, por medio del comando **cd**.

5. Compila el programa utilizando el comando **javac** seguido del nombre del archivo Java:

    - **javac MiPrimerPrograma.java**

    Para generar archivo ".class".

6. Ejecuta el programa compilado con el comando java seguido del nombre de la clase principal (sin la extensión ".class"):

    - **java MiPrimerPrograma**

7. Ver en la salida de la consola "Hola, Mundo".

(Opción IDE)

3. Dar clic en botón de compilación y ejecución.

4. Ver en la terminal del IDE "Hola, Mundo".

Revisando la JVM

La Máquina Virtual de Java (JVM, por sus siglas en inglés) es un componente esencial en la ejecución de programas escritos en Java. Esta se encarga de ejecutar el bytecode de Java, lo que permite que las aplicaciones Java sean independientes de la plataforma, ya que el bytecode se puede ejecutar en diferentes sistemas operativos y arquitecturas.

A continuación, se mencionan algunas características de la JVM:

- **Independencia de Plataforma:** La JVM permite que las aplicaciones escritas en Java se ejecuten en cualquier sistema operativo o arquitectura que tenga una JVM compatible. Esto es posible gracias a la capacidad de la JVM para interpretar el bytecode de Java.

- **Gestión de Memoria:** La JVM se encarga de la administración de la memoria, incluyendo la asignación y liberación de memoria para objetos en el heap y la recolección de basura (garbage collection) para recuperar memoria no utilizada.

- **Seguridad:** La JVM incluye mecanismos de seguridad para proteger el sistema huésped de código malicioso. Las aplicaciones Java se ejecutan en un entorno aislado llamado "sandbox", lo que limita su acceso a recursos del sistema.

- **Compilación Just-In-Time (JIT):** La JVM utiliza la compilación JIT para traducir el bytecode en tiempo de ejecución en código de máquina nativo. Esto mejora el rendimiento en comparación con la interpretación pura del bytecode.

- **Administración de Hilos:** La JVM admite la creación y administración eficiente de múltiples hilos de ejecución, lo que facilita la programación paralela y la concurrencia en las aplicaciones Java.

- **Clase Loader:** La JVM tiene un Class Loader que carga las clases necesarias en tiempo de ejecución. Esto permite la modularización y la carga dinámica de clases, lo que es fundamental en la plataforma Java.

- **Administración de Excepciones:** La JVM gestiona excepciones y errores, lo que facilita la detección y el manejo de problemas en las aplicaciones Java.

- **Recolección de Basura (Garbage Collection):** La JVM automatiza la gestión de la memoria y la eliminación de objetos no utilizados a través de la recolección de basura. Esto reduce la carga de trabajo del desarrollador en la gestión manual de la memoria.

- **Rendimiento y Optimización:** La JVM incluye técnicas de optimización, como la compilación Just-In-Time y la optimización de tiempo de ejecución, para mejorar el rendimiento de las aplicaciones Java.

- **Herramientas de Diagnóstico:** La JVM proporciona herramientas de diagnóstico y monitoreo, como JVisualVM, JConsole y Java Flight Recorder, que ayudan a los desarrolladores a evaluar el rendimiento y solucionar problemas en sus aplicaciones.

- **Extensibilidad:** La JVM es extensible y permite que se agreguen bibliotecas y componentes personalizados para satisfacer las necesidades específicas de las aplicaciones.

- **Gestión de Clases Dinámicas:** La JVM permite la carga de clases de forma dinámica en tiempo de ejecución, lo que es útil para crear aplicaciones con características de complementos o módulos.

En resumen, la JVM es una parte fundamental de la plataforma Java y proporciona características esenciales como la independencia de plataforma, la gestión de memoria, la seguridad, la gestión de hilos y la administración de excepciones. Estas características hacen que Java sea una plataforma versátil y poderosa para el desarrollo de aplicaciones en una amplia variedad de entornos y dispositivos [9].

Documentación API

La documentación de la API de Java es una referencia esencial para los desarrolladores que trabajan con el lenguaje de programación Java. Proporciona detalles sobre todas las clases, métodos y componentes disponibles en el lenguaje Java y en las bibliotecas

estándar, lo que facilita la comprensión y el uso de estas funcionalidades. Entre las fuentes de documentación de la API de Java están:

- **Oracle Java SE API Documentation (Oficial):**
  El sitio web oficial de Oracle proporciona documentación completa y detallada de la API de Java para cada versión de Java SE (Standard Edition) [10].

- **JavaDoc (Generación de Documentación de Código Fuente):**
  JavaDoc es una herramienta que permite generar documentación a partir del código fuente de las clases Java. Se puede incluir comentarios especiales en el código que JavaDoc procesará y generará documentación en formato HTML. Esta documentación personalizada es útil para proyectos internos.

- **Java Platform, Standard Edition (Java SE) Documentation (Oracle):**
  Además de la documentación de la API, el sitio web de Oracle ofrece una variedad de recursos y tutoriales para aprender sobre Java SE [11].

- **OpenJDK API Documentation:**
  OpenJDK es la implementación de código abierto de Java SE. Puedes encontrar documentación de la API en el sitio web de OpenJDK, que es similar a la proporcionada por Oracle [12].

- **Libros de Referencia sobre Java:**
  Además de la documentación en línea, hay libros de referencia excelentes que cubren la API de Java en detalle.

- **Búsqueda en Línea:**
  Además de las fuentes oficiales, se puede buscar en línea para encontrar explicaciones y ejemplos adicionales de cómo utilizar componentes específicos de la API de Java. Plataformas como Stack Overflow, blogs técnicos y tutoriales en línea son recursos útiles.

- **IDEs (Entornos de Desarrollo Integrados):**
  Muchos IDEs, como Eclipse, IntelliJ IDEA y NetBeans, proporcionan herramientas integradas que facilitan la búsqueda y la navegación de la documentación de la API de Java.

Al utilizar la documentación de la API de Java, se podrá aprovechar al máximo las bibliotecas y las clases disponibles en Java, lo que facilitará el desarrollo de aplicaciones y la solución de problemas.

Referencias

[1] «Java (lenguaje de programación),» Wikipedia, 4 Octubre 2023 . [En línea]. Available: https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n). [Último acceso: 25 Octubre 2023].

[2] «Plataforma Java,» Wikipedia, 24 Junio 2023 . [En línea]. Available: https://es.wikipedia.org/wiki/Plataforma_Java. [Último acceso: 25 Octubre 2023].

[3] «¿Cómo puedo instalar Java?,» Java, [En línea]. Available: https://www.java.com/es/download/help/download_options_es.html. [Último acceso: 25 Octubre 2023].

[4] «Oracle,» Oracle Corporation, 18 Agosto 2018. [En línea]. Available: https://www.oracle.com/mx/java/technologies/downloads/. [Último acceso: 25 Octubre 2023].

[5] «JDK Builds from Oracle,» Oracle Corporation, [En línea]. Available: https://jdk.java.net. [Último acceso: 25 Octubre 2023].

[6] G. Nieva, «Versiones y ediciones de Java.,» d.CodingGames();, 19 Febrero 2019. [En línea]. Available: https://dcodingames.com/versiones-y-ediciones-de-java/. [Último acceso: 25 Octubre 2023].

[7] «Historia de Java, un camino lleno de curiosidades,» Netec, 5 Noviembre 2018. [En línea]. Available: https://www.netec.com/post/historia-y-curiosidades-de-java. [Último acceso: 25 Octubre 2023].

[8] «Difference between JDK, JRE, and JVM,» DigitalOcean, 22 Agosto 2022. [En línea]. Available: https://www.digitalocean.com/community/tutorials/difference-jdk-vs-jre-vs-jvm. [Último acceso: 25 Octubre 2023].

[9] J. M. Alarcón, «¿Qué es la máquina virtual de Java o Java Virtual Machine?,» Campus MVP.es, 23 Octubre 2017. [En línea]. Available: https://www.campusmvp.es/recursos/post/que-es-la-maquina-virtual-de-java-o-java-virtual-machine.aspx. [Último acceso: 25 Octubre 2023].

[10] «Java Platform, Standard Edition Documentation,» Oracle, [En línea]. Available: https://docs.oracle.com/en/java/javase/. [Último acceso: 25 Octubre 2023].

[11] «Java SE Documentation,» Oracle, [En línea]. Available: https://www.oracle.com/java/technologies/. [Último acceso: 25 Octubre 2023].

[12] M. Reinhold, «OpenJDK API Documentation,» OpenJDK, 25 Octubre 2023. [En línea]. Available: https://openjdk.org/jeps/0. [Último acceso: 25 Octubre 223].