Jayaro Alvarez Eduardo
1TSM1
21-03-2021

1. **¿Qué es Java?**

Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems.

El lenguaje de programación Java fue desarrollado originalmente por James Gosling, de Sun Microsystems (constituida en 1983 y posteriormente adquirida el 27 de enero de 2010 por la compañía Oracle), y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

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, en Menlo Park, para desarrollarlo.

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

2. **¿Qué significan las siglas de Java?**

No está claro si es un acrónimo o no, aunque algunas fuentes señalan que podría tratarse de las iniciales de sus diseñadores: **J**ames Gosling, **A**rthur **V**an Hoff y **A**ndy Bechtolsheim. Otros abogan por el siguiente acrónimo, ***Just Another Vague Acronym*** ("sólo 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, 0xCAFEBABE. A pesar de todas estas teorías, el nombre fue sacado al parecer de una lista aleatoria de palabras.

Jayaro Alvarez Eduardo
1TSM1
21-03-2021

3. **Principales características del lenguaje.**

Es simple:

Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ es un lenguaje que adolece de falta de seguridad, pero C y C++ son lenguajes más difundidos, por ello Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje.

Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (reciclador de memoria dinámica). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y como es un thread de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria.

Es Orientado a Objetos:

Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje. Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Las plantillas de objetos son llamadas, como en C++, clases y sus copias, instancias. Estas instancias, como en C++, necesitan ser construidas y destruidas en espacios de memoria.

Es Distribuido:

Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.

La verdad es que Java en sí no es distribuido, sino que proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando.

Es 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 de desarrollo de aplicaciones en Java.

Además, para asegurar el funcionamiento de la aplicación, realiza una verificación de los byte-codes, que son el resultado de la compilación de un programa Java. Es un código de máquina virtual que es interpretado por el intérprete Java. No es el código máquina directamente entendible por el hardware, pero ya ha pasado todas las fases del compilador: análisis de instrucciones, orden de operadores, etc., y ya tiene generada la pila de ejecución de órdenes.

Es de Arquitectura Neutral:

Para establecer Java como parte integral de la red, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado.

El código fuente Java se "compila" a un código de bytes de alto nivel independiente de la máquina. Este código (byte-codes) está diseñado para ejecutarse en una máquina hipotética que es implementada por un sistema run-time, que sí es dependiente de la máquina.

Es Seguro:

Las aplicaciones de Java resultan extremadamente seguras, ya que no acceden a zonas delicadas de memoria o de sistema, con lo cual evitan la interacción de ciertos virus. Java no posee una semántica específica para modificar la pila de programa, la memoria libre o utilizar objetos y métodos de un programa sin los privilegios del kernel del sistema operativo. Además, para evitar modificaciones por parte de los crackers de la red, implementa un método ultraseguro de autentificación por clave pública. El Cargador de Clases puede verificar una firma digital antes de realizar una instancia de un objeto. Por tanto, ningún objeto se crea y almacena en memoria, sin que se validen los privilegios de acceso. Es decir, la seguridad se integra en el momento de compilación, con el nivel de detalle y de privilegio que sea necesario.

Dada, pues la concepción del lenguaje y si todos los elementos se mantienen dentro del estándar marcado por Sun, no hay peligro. Java imposibilita, también, abrir ningún fichero de la máquina local (siempre que se realizan operaciones con archivos, éstas trabajan sobre el disco duro de la máquina de donde partió el applet), no permite ejecutar ninguna aplicación nativa de una plataforma e

impide que se utilicen otros ordenadores como puente, es decir, nadie puede utilizar nuestra máquina para hacer peticiones o realizar operaciones con otra. Además, los intérpretes que incorporan los navegadores de la Web son aún más restrictivos. Bajo estas condiciones (y dentro de la filosofía de que el único ordenador seguro es el que está apagado, desenchufado, dentro de una cámara acorazada en un bunker y rodeado por mil soldados de los cuerpos especiales del ejército), se puede considerar que Java es un lenguaje seguro y que los applets están libres de virus.

Es Portable:

Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Los enteros son siempre enteros y además, enteros de 32 bits en complemento a 2. Además, Java construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que las ventanas puedan ser implantadas en entornos Unix, Windows o Mac.

Es Interpretado:

El intérprete Java (sistema run-time) puede ejecutar directamente el código objeto. Enlazar (linkar) un programa, normalmente, consume menos recursos que compilarlo, por lo que los desarrolladores con Java pasarán más tiempo desarrollando y menos esperando por el ordenador. Java es más lento que otros lenguajes de programación, como C++, ya que debe ser interpretado y no ejecutado como sucede en cualquier programa tradicional.

La verdad es que Java para conseguir ser un lenguaje independiente del sistema operativo y del procesador que incorpore la máquina utilizada, es tanto interpretado como compilado. Y esto no es ningún contrasentido, el código fuente escrito con cualquier editor se compila generando el byte-code. Este código intermedio es de muy bajo nivel, pero sin alcanzar las instrucciones máquina propias de cada plataforma y no tiene nada que ver con el p-code de Visual Basic. El byte-code corresponde al 80% de las instrucciones de la aplicación. Ese mismo código es el que se puede ejecutar sobre cualquier plataforma. Para ello hace falta el run-time, que sí es completamente dependiente de la máquina y del sistema operativo, que interpreta dinámicamente el byte-code y añade el 20% de instrucciones que faltaban para su ejecución. Con este sistema es fácil crear aplicaciones multiplataforma, pero para ejecutarlas es necesario que exista el run-time correspondiente al sistema operativo utilizado.

Es MultiThreaded:

Al ser multithreaded (multiproceso), Java permite muchas actividades simultáneas en un programa. Los threads (a veces llamados, procesos ligeros),

son básicamente pequeños procesos o piezas independientes de un gran proceso. Al estar los threads contruidos en el lenguaje, son más fáciles de usar y más robustos que sus homólogos en C o C++.

El beneficio de ser miltithreaded consiste en un mejor rendimiento interactivo y mejor comportamiento en tiempo real. Aunque el comportamiento en tiempo real está limitado a las capacidades del sistema operativo subyacente (Unix, Windows, etc.), aún supera a los entornos de flujo único de programa (single-threaded) tanto en facilidad de desarrollo como en rendimiento.

Cualquiera que haya utilizado la tecnología de navegación concurrente, sabe lo frustrante que puede ser esperar por una gran imagen que se está trayendo. En Java, las imágenes se pueden ir trayendo en un thread independiente, permitiendo que el usuario pueda acceder a la información en la página sin tener que esperar por el navegador.

Es Dinámico:

Java se beneficia todo lo posible de la tecnología orientada a objetos. Java no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución. Las librería nuevas o actualizadas no paralizarán las aplicaciones actuales (siempre que mantengan el API anterior).

Java también simplifica el uso de protocolos nuevos o actualizados. Si su sistema ejecuta una aplicación Java sobre la red y encuentra una pieza de la aplicación que no sabe manejar, tal como se ha explicado en párrafos anteriores, Java es capaz de traer automáticamente cualquiera de esas piezas que el sistema necesita para funcionar.

Java, para evitar que los módulos de byte-codes o los objetos o nuevas clases, haya que estar trayéndolos de la red cada vez que se necesiten, implementa las opciones de persistencia, para que no se eliminen cuando de limpie la caché de la máquina.

4. ¿Qué es el JDK de Java?

Java™ Development Kit (JDK) es un software para los desarrolladores de Java. Incluye el intérprete Java, clases Java y herramientas de desarrollo Java (JDT): compilador, depurador, desensamblador, visor de applets, generador de archivos de apéndice y generador de documentación.

El JDK le permite escribir aplicaciones que se desarrollan una sola vez y se ejecutan en cualquier lugar de cualquier máquina virtual Java. Las aplicaciones Java desarrolladas con el JDK en un sistema se pueden usar en otro sistema sin tener que cambiar ni recompilar el código. Los archivos de clase Java son portables a cualquier máquina virtual Java estándar.

OpenJDK: es la versión libre de la plataforma de desarrollo Java bajo concepto de lenguaje orientado a objetos. Es el resultado de esfuerzos constantemente realizados por la empresa denominada Sun Microsystems. Esta implementación se encuentra catalogada dentro de la licencia GPL de GNU con una excepción de enlaces, por lo que algunos de los componentes de los folders de clases y sitios web de Java se ultiman de los términos de la licencia para poder ser considerados dentro de la versión estipulada como GNU.

5. **¿Qué es NetBeans?**

La plataforma NetBeans es un amplio marco de Java en el que puede basar grandes aplicaciones de escritorio. NetBeans IDE en sí es una de las cientos de aplicaciones basadas en la plataforma NetBeans. La plataforma NetBeans contiene API´s que simplifican el manejo de ventanas, acciones, archivos y muchas otras cosas típicas de las aplicaciones.

Cada característica distinta en una aplicación de la plataforma NetBeans puede ser proporcionada por un módulo NetBeans distinto, que es comparable a un complemento. Un módulo de NetBeans es un grupo de clases de Java que proporciona una aplicación con una característica específica.

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

Sun Microsystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos. Actualmente Sun Microsystems es administrado por Oracle Corporation.

Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente.

6. **Principales características**.

Suele dar soporte a casi todas las novedades en el lenguaje Java. Cualquier preview del lenguaje es rápidamente soportada por Netbeans.

Buen editor de código, multilenguaje, con el habitual coloreado y sugerencias de código, acceso a clases pinchando en el código, control de versiones, localización de ubicación de la clase actual, comprobaciones sintácticas y

semánticas, plantillas de código, coding tips, herramientas de refactorización, entre otras.

Simplifica la gestión de grandes proyectos con el uso de diferentes vistas, asistentes de ayuda, y estructurando la visualización de manera ordenada, lo que ayuda en el trabajo diario. Una vez que nos metemos en una clase java, por poner un ejemplo, se nos mostrarán distintas ventanas con el código, su localización en el proyecto, una lista de los métodos y propiedades (ordenadas alfabéticamente), también hay una vista que nos presenta las jerarquías que tiene nuestra clase y otras muchas opciones. Por supuesto personalizable según el gusto de cada usuario.

Herramientas para depurado de errores: el debugger que incluye el IDE es bastante útil para encontrar dónde fallan las cosas. Podemos definir puntos de ruptura en la línea de código que nos interese, monitorizar en tiempo real los valores de propiedades y variables, se nos permite ir paso a paso, ejecutar un método de un tirón, o entrar dentro, en fin, las opciones típicas, pero que tan útiles son en el trabajo diario. Incluso podemos usar el debugger en caliente, conectándonos a él cuándo ya tenemos un proceso ejecutándose.

Optimización de código: por su parte el Profiler nos ayuda a optimizar nuestras aplicaciones e intentar hacer que se ejecuten más rápido y con el mínimo uso de memoria. Podemos igualmente configurarlo a nuestro gusto, aunque por defecto, nos ofrece opciones bastante útiles. Lo importante es que podemos ver el comportamiento de nuestra aplicación y obtener indicadores e información de cómo y cuantos recursos consume, cuantos objetos se crean, también podemos obtener capturas del estado del sistema en diferentes momentos (Snapshots) y compararlos entre sí.

Acceso a base de datos: desde el propio Netbeans podemos conectarnos a distintos sistemas gestores de bases de datos, como pueden ser Oracle, MySql y demás, y ver las tablas, realizar consultas y modificaciones, y todo ello integrado en el propio IDE.

Se integra con diversos servidores de aplicaciones, de tal manera que podemos gestionarlos desde el propio IDE: inicio, parada, arranque en modo debug, despliegues.

Jayaro Alvarez Eduardo
1TSM1
21-03-2021

Fuentes de consulta

https://es.wikipedia.org/wiki/Java_(lenguaje_de_programación)

http://www.itlp.edu.mx/web/java/Tutorial%20de%20Java/Intro/carac.html

https://www.ibm.com/support/knowledgecenter/es/ssw_ibm_i_73/rzaha/sunjdk.htm

https://es.scribd.com/doc/125144805/Que-es-el-JDK

https://netbeans.apache.org/kb/docs/platform/index.html

https://es.wikipedia.org/wiki/NetBeans#NetBeans

https://www.genbeta.com/desarrollo/netbeans-1