UD2 Mi primer programa

En este tema, crearemos nuestro primer programa y para ello, deberemos conocer cierta información de antemano.

Contenido

1. Comprendiendo Java

Java es uno de los lenguajes más difíciles de aprender, no debido a su sintaxis o por el hecho de que está orientado a objetos, sino más bien por su amplitud. Dentro de Java existen gran cantidad de bibliotecas de clases (librerías) para hacer cualquier tipo de cosa. Habitualmente es raro comenzar desde cero una aplicación en Java, escribiendo la primera sentencia de código, normalmente nos apoyaremos en algo ya escrito, que implementa las bases de aquello que queremos construir.

Java es muy grande y aunque este hecho parece que represente una dificultad en el aprendizaje, es también una de sus ventajas. Se puede decir que, aunque un desarrollador tenga muchos años de experiencia programando en este lenguaje, es prácticamente imposible que conozca todas las áreas para las que Java se puede utilizar.

Un poco de historia…

Para hablar de la historia de java, primero debemos remontarnos a los años 80s, donde C podía considerarse el lenguaje de programación por excelencia. Era un lenguaje versátil, que podía actuar a bajo nivel y resolvían problemas muy complejos. Era la cima de la programación estructurada, para resolver estos complejos algoritmos, se generaban grandes procedimientos con un código muy complicado de mantener a largo plazo. Por ello empezó a surgir como alternativa la programación orientada a objetos, y con ella nació C++.

Java nace con el objetivo de crear un lenguaje de programación parecido a C++ en estructura y sintaxis, fuertemente orientado a objetos, pero que pueda ser usado bajo cualquier arquitectura y sistema operativo "Write Once, Run Anywhere (escríbelo una vez, ejecútalo en cualquier sitio)".

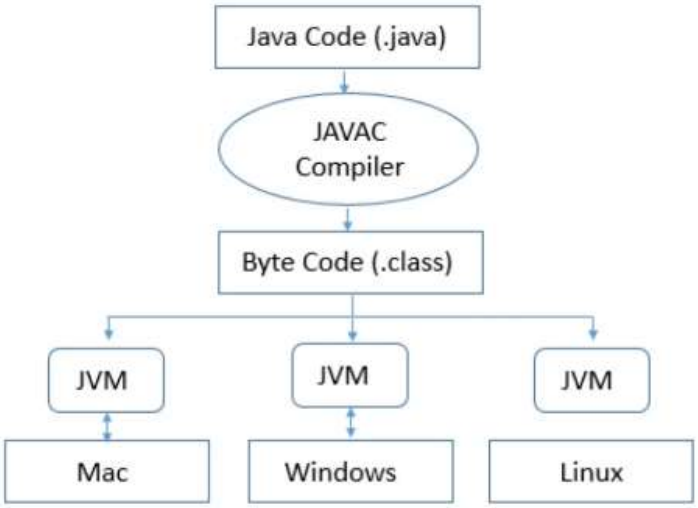
Fue desarrollado en sus inicios por James Gosling, en el año 1991, aunque inicialmente era conocido como Oak o Green. La primera versión del lenguaje es publicada por Sun Microsystems en 1995. Y es en la versión del lenguaje JDK 1.0.2, cuando pasa a llamarse Java,

corría el año 1996. Actualmente Java es propiedad de Oracle, que compró a Sun y es la responsable de su comercialización y la continuidad del desarrollo del lenguaje.

En las primeras versiones de Java 1.1, 1.2 y 1.3 es en la que el lenguaje va tomando forma, con la inclusión de tecnologías como JavaBeans, JDBC para el acceso a base de datos, RMI para las invocaciones en remoto, Collections para la gestión de múltiples estructuras de datos o AWT para el desarrollo gráfico, entre otros.

Java, lenguaje compilado e interpretado

Una de las características más relevante del lenguaje de programación Java es el hecho de que utiliza una máquina virtual para la ejecución de los programas, haciendo ésta de intermediaria entre la máquina real (HW + SO) y los programas desarrollados por los programadores.



El lenguaje Java es a la vez compilado e interpretado. Con el compilador se convierte el código fuente que reside en archivos cuya extensión es .java, a un conjunto de instrucciones que recibe el nombre de bytecodes que se guardan en un archivo cuya extensión es .class. Estas instrucciones son independientes del tipo de ordenador. Después, el intérprete ejecuta cada una de estas instrucciones en un ordenador específico (Windows, Macintosh, etc). Solamente es necesario, por tanto, compilar una vez el programa, pero se interpreta cada vez que se ejecuta en un ordenador.

En este punto es relevante hacer notar que, cuando nosotros generamos un código fuente, dejamos en manos del compilador y del intérprete la traducción a código máquina del mismo. Una discusión suele ser, que forma es la más óptima de realizar ciertas sentencias, pero ¿Qué se le envía al procesador en última instancia? La calidad de la traducción queda fuera de nuestro alcance, pero es por ello que, las sucesivas mejoras en los compiladores y en las versiones del propio lenguaje, permiten sucesivamente la transformación en un código máquina más óptimo, lo que implica: menos gasto de memoria, mejor reutilización de ésta, menor tamaño, más velocidad…

Al final es importante ser conscientes de que nuestros programas son transformados a instrucciones, y debemos, si el caso lo requiere, intentar cambiar nuestro código de alto nivel para ayudar al compilador a pasarle a la CPU un código más óptimo para ella.

Entorno de ejecución Java

A continuación, describimos todos los materiales que debes tener para trabajar con Java.

JVM (Java Virtual Machine)

La máquina virtual Java es un software que ejecuta el programa de Java línea a línea. Los desarrolladores configuramos los ajustes de la JVM para administrar recursos de programas cuando se ejecuta la aplicación de Java. Por ejemplo, es posible modificar la configuración de memoria de la JVM y comprobar cuánta memoria interna consumen nuestras aplicaciones de Java durante el tiempo de ejecución.

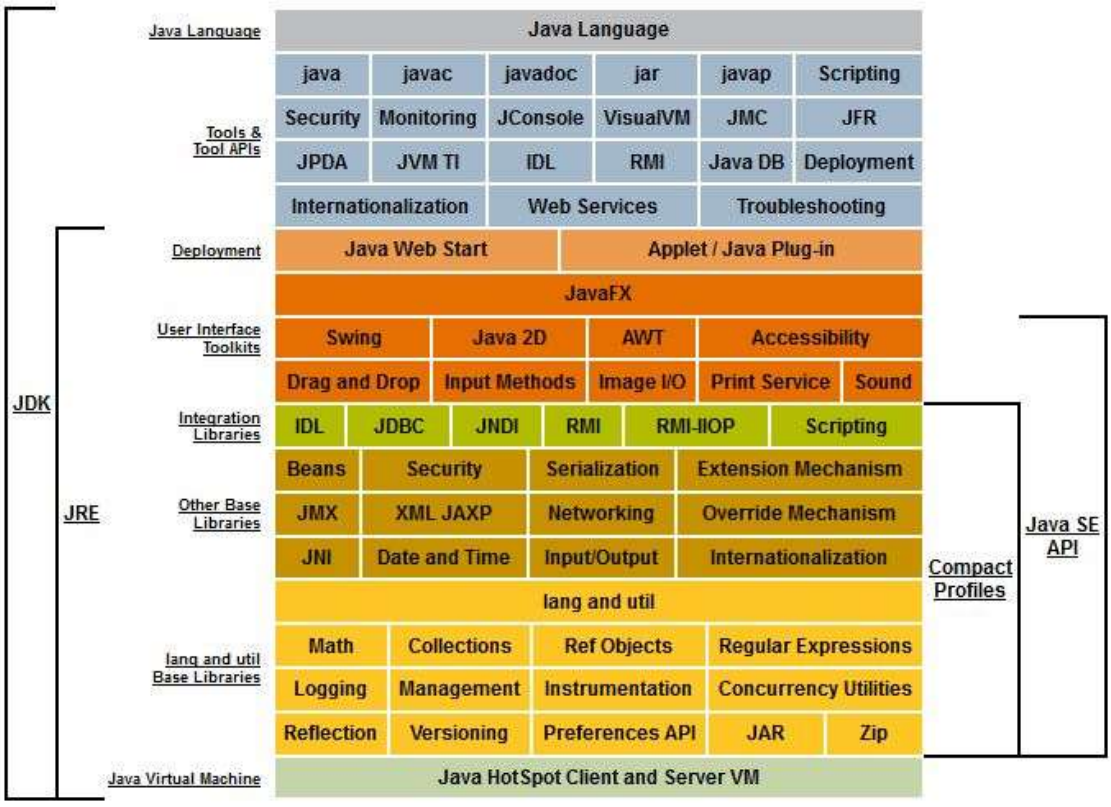
JDK (Java Development Kit) o SDK (Software Development Kit)

El kit de desarrollo de Java es una colección de herramientas de software que se pueden utilizar para desarrollar aplicaciones de Java como, por ejemplo: un compilador, un depurador y otras herramientas que se encuentran habitualmente en cualquier entorno de desarrollo de software . Podemos configurar el JDK en el entorno de desarrollo con sólo descargarlo e instalarlo, previamente es necesario seleccionar la versión de software del JDK que coincida con la versión de Java que deseamos utilizar.

JRE (Java Runtime Environment)

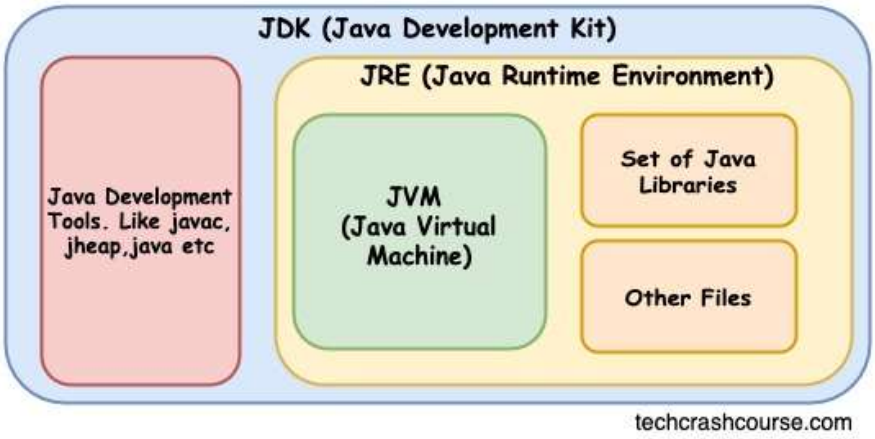
El entorno de ejecución de Java combina el código de Java creado por los desarrolladores, con el código adicional incorporado que se denominan bibliotecas. Además, incluye la JVM, la cual depende del sistema operativo, de forma que el JRE genera una única copia de código ejecutable para la máquina. Por lo tanto, el JRE utiliza estos componentes de software para ejecutar el código intermedio en cualquier dispositivo.

Oracle nos proporciona el siguiente diagrama conceptual sobre los productos de la Plataforma Java de Oracle:



FUENTE: https://www.oracle.com/java/technologies/platform-glance.html

En la imagen siguiente podemos verlo de forma simplificada:



El uso de Java es gratuito, pero, desde que salió la versión Java 11, Oracle modificó las condiciones de uso. Hasta este momento podías descargar y programar con el Kit de Desarrollo de Java oficial de Oracle (JDK) y luego poner tu aplicación en producción o distribuirla sin tener que pagar nada al gigante del software. Sin embargo, a partir de Java 11 y del JDK 11, aunque puedes seguir desarrollando con él, tendrás que pagar una licencia a Oracle si quieres utilizarlo para poner las aplicaciones en producción. El coste es de 2,5 dólares al mes por cada usuario de escritorio, y de 25 dólares por procesador en el caso de aplicaciones de servidor. Aquí tienes la lista de precios oficial.

Esto no afecta a versiones anteriores del JDK, por lo que si se usa Java 8, 9 o 10 sigue siendo gratuito.

Además de la nueva licencia y el coste asociado, existe otra novedad: desde Java 9 Oracle se ha comprometido a sacar una nueva versión "mayor" de Java (y del JDK) cada 6 meses, con dos actualizaciones garantizadas entre ellas: una al mes del lanzamiento (la x.0.1) y otra al cabo de tres meses (la x.0.2). Después tendrás que actualizarte a la versión "mayor" siguiente, que sale a los 6 meses. Esto es un problema para muchas organizaciones puesto que un cambio de versión muchas veces supone problemas e inestabilidad. Es por eso que una gran mayoría de empresas sigue todavía con Java 8 a pesar de las nuevas versiones que han aparecido.

Esto no afecta a versiones anteriores del JDK, por lo que si usas Java 8, 9 o 10 sigue siendo gratuito.

Pero ahora bien, en la parte superior, cuando hacíamos referencia a la JDK, también hemos escrito SDK ¿Por qué? La versión oficial, la de pago es la que lleva el nombre JDK, pero por suerte tenemos disponibles versiones con licencias GPL, los kit de desarrollo de software de Java.

La compilación OpenJDK es una implementación open source gratuita de Java Platform, Standard Edition (Java SE) que surgió de una iniciativa de Sun Microsystems en 2006, la cual consistía en poner a disposición de todos la implementación de Java mediante el proyecto OpenJDK.

El OpenJDK es la opción por defecto para seguir programando en Java y publicando aplicaciones sin pasar por caja, siendo muy similar técnicamente hablando al JDK. Se diferencia de él principalmente en aspectos de actualizaciones y soporte. Aunque en cuanto a actualizaciones sigue la línea del JDK, con respecto al soporte es donde la cosa cambia.

¿Dónde descargar el OpenJDK? adoptium.net, Amazon Corretto, Azul Zulu y Red Hat.

2. Manipulación básica de datos

El propósito principal de un ordenador, es procesar información. La información son datos. Por lo tanto, el propósito principal de un ordenador es procesar datos de todo tipo, los datos serán necesarios para la ejecución de todos los programas.

"Los datos son, así, la información (valores o referentes) que recibe el computador a través de distintos medios, y que es manipulada mediante el procesamiento de los algoritmos de programación. Su contenido puede ser prácticamente cualquiera: estadísticas, números, descriptores, que por separado no tienen relevancia para los usuarios del sistema, pero que en conjunto pueden ser interpretados para obtener una información completa y específica."
Fuente: https://concepto.de/dato-en-informatica/#ixzz8Dxbudpzy.

Debemos tener en cuenta, que, aunque hablemos del término datos en plural, en los programas, cada dato a tratar es un elemento individual e independiente, sobre el cual operarán los algoritmos. Pensemos en un programa que realiza una suma: necesitaremos dos números que se utilizarán como datos de entrada, y se generará un resultado final, que será otro dato.

2.1. Representación de la información

Dicho esto, ¿Cómo va a realizarse la representación de la información en el computador?

Para que un ordenador pueda tratar o transmitir la información, debe estar codificada en un formato adecuado: a esto le llamamos representación de la información.

Podemos considerar dos niveles de representación de la información: Externa o para el ser humano e interna o para el computador.

En el primer nivel, la representación de la información, para el ser humano, se produce inicialmente oral y se puede representar y registrar de tres formas: Numérica, Alfanumérica y Simbólica.

En el segundo nivel, la representación interna en un computador. Recordemos que, por su naturaleza digital, todos los datos que se manejan dentro de un ordenador se codifican como una secuencia binaria, es decir, de dos posibles valores, 0's y 1's, por lo tanto, la representación de cualquier información tendrá que realizarse a través de una secuencia de este tipo.

Cada lenguaje de programación incorpora sus tipos de datos propios, por lo tanto, para la representación de la información que deseemos manipular, tendremos que escoger la que ofrece el lenguaje, que mas se acerca al tipo de dato a tratar.

Ahora bien, no todos los datos que necesitemos tratar, van a tener una relación directa con un tipo de dato del lenguaje, esto sólo va a ocurrir con aquellos tipos de datos simples, que son los llamados, tipos primitivos. A partir de estos tipos primitivos, es posible construir otros más complejos.

2.2. Tipos de datos primitivos

Aunque, como hemos dicho, cada lenguaje tiene sus propios tipos de datos, hay cuatro tipos que son básicos y soportados por todos ellos ya que están directamente relacionados con secuencias binarias dentro del ordenador: Números enteros, reales, caracteres y booleanos.

2.2.1. Entero

Representa un valor numérico, sin decimales y que puede ser con, o sin signo (positivo o negativo).

Ejemplos: 8, 0, 18953626, -485699, -1...

Este tipo de datos se identifica en java con la palabra clave **int**.

2.2.2. Real

Representa un valor numérico, con decimales y que puede ser con, o sin signo (positivo o negativo).

Ejemplo: 3.5, -6.09783458, 25698.36845, -0.25, 0.0, 5.0…

Este tipo de datos se identifica en java con la palabra clave **double**.

Viendo este tipo de datos, podríamos pensar que es mejor utilizar siempre este, ya que es capaz de representar los decimales, pero también los números enteros. Bien, pues aunque así es, realmente para el ordenador es mucho más sencillo y rápido operar con números enteros que con reales. Los reales, también necesitan más memoria para almacenarse. Por lo tanto, antes datos que de antemano sabemos que nunca van a permitir decimales, es mejor usar el tipo entero.

Para saber más sobre representación de entero y reales, se recomienda buscar información sobre como se representan en el computador y sobre el "complemento a dos".

2.2.3. Carácter

Representa una unidad de texto del alfabeto, números, e incluso algunos signos.

Ejemplos: 'w', 'h', 'H', '.', '<'…

Este tipo de datos se identifica en java con la palabra clave **char**.

Para escribir este tipo de datos siempre se hace entre comillas simples y entre ellas únicamente podremos escribir un carácter.

Es importante diferenciar un tipo de dato entero, de un carácter: no es lo mismo 9, que '9'. El primero es un número mediante el cual podremos operar, el segundo es una representación de texto.

Sistemas de representación de los caracteres

Para representar un carácter internamente se utilizan tablas de códigos establecidos, como, por ejemplo; ASCII i UNICODE.

ASCII era el más utilizado al principio. Representaba 128 caracteres y más tarde, al necesitar añadir los caracteres especiales, surgió ASCII extendido, con 256.

Pero el código ASCII representa alfabetos occidentales, por lo que con los caracteres que contenía, no cubría alfabetos como el asiático, el cirílico,… así que se creó la codificación UNICODE, que mediante sus 65.536 caracteres, permite incluso agregar emoticonos.

2.2.4. Booleano

Representa un tipo de valor lógico verdadero o falso.

Ejemplo: true (verdadero) o false (falso). Un ejemplo aplicado podría ser la representación de la información de si un alumno es menor de edad, o por ejemplo, de si quedan alumnos en una lista por revisar.

Este tipo de datos se identifica en java con la palabra clave **boolean**.

2.2.5. Otras formas de representar algunos tipos primitivos

ENTEROS			
Nombre	**Bits**	**Bytes**	**Rango**
BYTE	8	1	–128 to 127
SHORT	16	2	–32,768 to 32,767
INT	32	4	–2,147,483,648 a 2,147,483,647
LONG	64	8	–9,223,372,036,854,775,808 a 9,223,372,036,854,775,807

REALES			
Nombre	**Bits**	**Bytes**	**Rango**
FLOAT	32	4	1.4e–045 a 3.4e+038
DOUBLE	68	8	4.9e–324 a 1.8e+308